JAVA-XSC:

Uma Biblioteca JAVA para Computações Intervalares

JOSÉ ENÉAS MONTENEGRO DUTRA

Orientador: Prof. Dr. BENJAMÍN RENÉ CALLEJAS BEDREGAL

Dissertação apresentada ao Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte para obtenção do título de Mestre em Sistemas e Computação.

NATAL

Rio Grande do Norte - Brasil

Fevereiro, 2000

JAVA-XSC:

Uma Biblioteca JAVA para Computações Intervalares

Dissertação de Mestrado

José Enéas Montenegro Dutra

Orientador Prof. BENJAMÍN RENÉ CALLEJAS BEDREGAL

Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Mestrado em Sistemas e Computação
Campus Universitário s/n, Lagoa Nova, Natal-RN
59072-970, Fax: (084)215-3780
eneas@ufrnet.ufrn.br

NATAL
Rio Grande do Norte - Brasil
Fevereiro, 2000

Banca Examinadora

....

Prof. Dr. Benedito Melo Acióly

....

Prof. Dr. Benjamín René Callejas Bedregal

....

Prof. Dr. Dario José Aloise

....

Prof^a. Dra. Márcia de Paiva Bastos Gottgtroy

Dedico esta dissertação a meus pais
José de Freitas Dutra e *Riograndina Montenegro Dutra*.

Agradecimentos

Após essa longa e difícil jornada, resta-nos agradecer a todos aqueles que diretamente ou indiretamente, contribuíram para que chegássemos até aqui.

Na impossibilidade de citar todos os nomes que fizeram a travessia conosco nesta construção, permitimo-nos destacar os seguintes nomes, na certeza de que somos gratos a todos que, direta ou indiretamente, concorreram para o sucesso deste trabalho.

Assim, apresentamos a profundidade dos reconhecimentos plenos:

A Deus, pelo dom precioso da vida e que tua presença ilumine sempre o nosso futuro.

Aos meus pais, *José de Freitas Dutra e Riograndina Montenegro Dutra*, que tiveram muita paciência e compreensão nos momentos mais dificeis da minha vida.

Aos meus irmãos *Eurico José Montenegro Dutra, Eduardo Francisco Montenegro Dutra*, pelo eterno companheirismo e amizade.

Ao professor orientador, *Dr. Benjamín René Callejas Bedregal*, que tem se dedicado grandiosamente à pesquisa, ao magistério, aos seus alunos, venho agradecer por sua tão competente orientação. A minha admiração e agradecimento pelo apoio constante, pela confiança, amizade e, principalmente, pelo incentivo dedicado.

Aos professores do DIMAp, aos professores do Departamento pelas significativas contribuições que enriqueceram este estudo. Em especial, aos *Profs Regivan Hugo Nunes Santiago*, *Dario José Aloise*, *Nilda Maria Teixeira de Cerqueira* e a nossa Coordenadora *Prof*^{a}. *Anamaria Martins Moreira*.

Aos professores do Ceres-Caicó-UFRN, que permitiram e me incentivaram para a conclusão deste mestrado, especialmente à ex-Diretora do CERES *Prof*^{a}. *Maria da Lúcia Costa Bezerra*; à Diretora atual *Prof*^{a}. *Maria das Graças do Lago Borge*; Chefe de Departamento de Formação Básica, *Prof. Álvaro Freire Damasceno* e Vice-Chefe *Carlos José Wanderley Ferreira*; aos Coordenadores, do Curso de Matemática, *Prof. Luis Gonzaga Vieira Filho* e de Contabilidade *Prof. Celso Luiz Souza de Oliveira*.

Aos ex-Diretores do Núcleo de Processamento de Dados da UFRN, em especial aos *Profs. Pedro Fernandes Maia* e *Marcos Antonio Alves*, e ao atual superintendente de Informática, *Prof. Galileu Batista*, que incentivaram e permitiram o meu afastamento para a conclusão deste curso.

Aos funcionários do DIMAp, em especial a *José Francisco*, *Ana Cláudia* e *José Aguiar* que sempre contribuíram comigo, aos quais sou eternamente grato.

Aos alunos, pelas contribuições dadas a este trabalho em especial a *Rodrigo* do LabLIC-DIMAp, e *Ming Chyi* da University of Houston, aluno orientado pelo Prof. ChenYi Hu, que tem feito pesquisas sobre bibliotecas intervalares.

Aos colegas, pela amizade e apoio ao longo do curso. Em especial a *Mônica, Cláudia e Aarão.*

Aos amigos, que me apoiaram e incentivaram nos momentos mais difíceis, os meus sinceros agradecimentos.

Resumo

Esta dissertação estende a linguagem de programação JAVA com bibliotecas intervalares para facilitar o desenvolvimento de computações científicas. Chamaremos esta extensão de JAVA-XSC, onde XSC é um acrônimo de "Language **Ex**tensions for **S**cientific **C**omputation" ou linguagem estendida para a computação científica. A linguagem JAVA-XSC conterá uma biblioteca em JAVA com as rotinas elementares e as funções padrão úteis da matemática intervalar, de modo a permitir que alunos, pesquisadores e usuários da área de matemática computacional a possam utilizar no desenvolvimento de suas atividades. A biblioteca será uma classe cujos objetos serão intervalos e as funções intervalares serão implementadas como métodos desta classe uma vez que a linguagem é orientada a objetos.

Palavras-chaves: Biblioteca, Computações científicas, Intervalos, JAVA, JAVA-XSC.

Abstract

This dissertation extends the JAVA Language of Programming with libraries in order to facilitate the development of scientific computations. We will call this extension JAVA-XSC where XSC is the acronym for "Language E**x**tensions for **S**cientific **C**omputation" or language which is extended towards computation scientific. The JAVA-XSC language will have a library in JAVA with elementary routines and useful standard functions of interval mathematics, so as to enable students, researchers and users of the area of computational mathematics to use it in the development of their activities. The library will be a class in which objects shall be intervals and the funtions will be implemented as methods once the language is oriented to objects.

key-words: Library, Scientific computations, Intervals, JAVA, JAVA-XSC.

Conteúdo

Lista de Símbolos

$\mathbb{N}$	Conjunto de números naturais
$\mathbb{Z}$	Conjunto de números inteiros
$\mathbb{R}$	Conjunto de números reais
$\mathbb{I}(\mathbb{F})$	Conjunto dos intervalos de ponto flutuante
$\mathbb{I}(\mathbb{R})$	Conjunto de todos os intervalos reais
$\mathbb{I}(\mathbb{R}^+)$	Conjunto de todos os intervalos reais positivos
$\mathbb{F}$	Conjunto dos números de ponto flutuante
$A = [a_1; a_2]$	Intervalo real de extremos a_1 e a_2
$F(X)$	Função intervalar
$\mathcal{I}(f; X)$	Imagem intervalar da função f em X
X^{-1}	Inverso do intervalo X
$F'(X)$	Derivada intervalar f
$(X_n)_n$	Seqüência intervalar
$maxX$	Maior elemento do conjunto X
$minX$	Menor elemento do conjunto X
$f : \mathbb{R} \to \mathbb{R}$	Função real
$Dom(f)$	Domínio da função f
$Cod(f)$	Contra-domínio da função f
$Img(f)$	Imagem da função f
$Graf(f)$	Representação gráfica da função f
$f(x)$	Valor da função no ponto x
$f'(x)$	Valor da derivada da função f no ponto x
$f''(X)$	Valor da derivada segunda da função f no ponto x
$\triangle$	Arredondamento para cima
$\bigtriangledown$	Arredondamento para baixo
$[](x)$	Arredondamento genérico
$\lfloor\rfloor$	Arredondamento para menor

$\ll$	Relação essencialmente abaixo
$\subseteq$	Subconjunto, contido ou igual
$\supseteq$	Subconjunto, contém ou igual
$\sqsubseteq$	Relação de ordem do domínio contínuo $\mathbb{I}(\mathbb{R})$
$\in$	Pertence
$\notin$	Não pertence
$\cup$	União
$\overline{A \cup B}$	União convexa de A e B
$\cap$	Interseção
$\exists$	Existe
$\therefore$	Portanto
$\uparrow$	Indefinido
π	Número real pi
$N(x)$	Operador newtoniano
x_k	Elemento da seqüência
x_*	Limite de uma seqüência, raiz de uma equação
$dist$	Distância entre dois intervalos
$diam$	Diâmetro de um intervalo
med	Ponto médio de um intervalo
x^2	Quadrado de x
$\sqrt{\ }$	Raiz quadrada
e^x	Função exponencial real de x
Exp	Função exponencial intervalar
$Exp2$	Função exponencial intervalar com base 2
$Exp10$	Função exponencial intervalar com base 10
x^n	Função potência real elevada a n
X^n	Função potência intervalar elevada a n
ln	Função logaritmo real
Ln	Função logaritmo intervalar
$log2$	Função logaritmo de base 2
$Log2$	Função logaritmo de base 2 intervalar
$log10$	Função logaritmo de base 10
$Log10$	Função logaritmo de base 10 intervalar
sen	Função seno real
Sen	Função seno intervalar
cos	Função co-seno real
Cos	Função co-seno intervalar
tan	Função tangente real
Tan	Função tangente intervalar
cot	Função co-tangente real
Cot	Função co-tangente intervalar

sec	Função secante real
Sec	Função secante intervalar
csc	Função co-secante real
Csc	Função co-secante intervalar
$arcsen$	Função arco seno real
$Arcsen$	Função arco seno intervalar
$arccos$	Função arco co-seno real
$Arccos$	Função arco co-seno intervalar
$arctan$	Função arco tangente real
$Arctan$	Função arco tangente intervalar
$arccot$	Função arco co-tangente real
$Arccot$	Função arco co-tangente intervalar
$arcsec$	Função arco secante real
$Arcsec$	Função arco secante intervalar
$senh$	Função seno hiperbólico real
$Senh$	Função seno hiperbólico intervalar
$cosh$	Função co-seno hiperbólico real
$Cosh$	Função co-seno hiperbólico intervalar
$tanh$	Função tangente hiperbólica real
$Tanh$	Função tangente hiperbólica intervalar
$coth$	Função co-tangente hiperbólica real
$Coth$	Função co-tangente hiperbólica intervalar
$sech$	Função secante hiperbólica real
$Sech$	Fuunção secante hiperbólica intervalar
$csch$	Função co-secante hiperbólica real
$Csch$	Função co-secante hiperbólica intervalar
$senhI$	Função seno hiperbólico inverso real
$SenhI$	Função seno hiperbólico inverso intervalar
$coshI$	Função co-seno hiperbólico inverso real
$CoshI$	Função co-seno hiperbólico inverso intervalar
$tanhI$	Função tangente hiperbólica inversa real
$TanhI$	Função tangente hiperbólica inversa intervalar
$cothI$	Função co-tangente hiperbólica inversa real
$CothI$	Função co-tangente hiperbólica inversa intervalar
$sechI$	Função secante hiperbólica inversa real
$SechI$	Função secante hiperbólica inversa intervalar
$cschI$	Função co-secante hiperbólica inversa real
$CschI$	Função co-secante hiperbólica inversa intervalar
$IEEE$	Institute of Electral and Eletronics Engineers
XSC	Language Extensions for Scientific Computation

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

As pesquisas no campo da aritmética computacional estão sendo desenvolvidas desde os anos sessenta, com o objetivo de controlar os erros computacionais e para que os computadores suportem uma aritmética muito mais poderosa e precisa que a aritmética empregada na maioria das linguagens modernas de programação.

A aritmética intervalar proposta por Moore, em 1966, possibilitou um grande desenvolvimento destas pesquisas. Esta aritmética trata com dados na forma de intervalos numéricos e tem como objetivo automatizar a análise de erro computacional. Serve para controlar o erro de arredondamento e para representar dados inexatos, aproximações e erros de truncamento de procedimentos. Atualmente, vem sendo empregada na elaboração de algoritmos numéricos autovalidáveis e com controle de erro computacional.

O uso da aritmética intervalar permite uma alta exatidão aritmética que é uma qualidade necessária nos ambientes que propiciam a resolução de problemas da Computação Científica e das Engenharias. Ela proporciona que o resultado calculado nas operações aritméticas, mesmo de um produto escalar, diferencie de apenas um arredondamento do valor exato. A minimização dos arredondamentos resulta em qualidade no resultado que também é uma das características necessárias à resolução de problemas com verificação do resultado.

A necessidade de termos linguagens de programação para estas aritméticas computacionais com esse suporte à computação científica fez que surgissem, na cooperação de institutos de pesquisas universitários e empresas (como por exemplo, a IBM), as linguagens com extensões científicas, conhecidas como XSC, que é o acrônimo de Language E**x**tensions **S**cientific **C**omputation.

As linguagens XSC provêem características indispensáveis para o desenvolvimento de modernos softwares numéricos e aplicações científicas tais como: controle de arredondamento, tipos de dados com exatidão após a virgula; bibliotecas com as principais rotinas matemáticas para a resolução de problemas; arrays dinâmicos;

conceito de operador (operadores definidos pelo usuário); tipos de dados não existentes nas linguagens comuns, como dado complex (complexo), interval (intervalo); além de várias outras características.

Nos últimos anos, a visão sobre a computação de rede ganhou uma crescente aceitação à medida que cada vez mais soluções computacionais se voltam para internet e intranet. A linguagem de programação JAVA para aplicações na Rede parece ter gerado maior força do que a própria Internet.

Trata-se de uma linguagem de programação orientada a objetos, independente de plataforma, que está revolucionando a World Wide Web e está tendo um efeito extraordinário na rede, no desenvolvimento de aplicações. É a primeira linguagem que vai ao encontro do desafio de distribuir aplicações dinâmicas, através de internet e intranets corporativas, e que permite aos criadores de páginas Web, desenvolvê-las com interatividade e em tempo real. Com o uso da linguagem JAVA, um usuário pode pedir aplicações através da internet ou intranet e processá-las numa máquina local. O emissor não precisa conhecer o seu ambiente de hardware/software, pois as aplicações processam indistintamente em qualquer plataforma.

A linguagem JAVA, por suas características e qualidades vistas acima, associada com uma extensão científica, poderá tornar-se imbatível nas aplicações computacionais e científicas na solução de problemas físicos, químicos e das engenharias que necessitam de alta exatidão, tendo como vantagem o uso na internet.

Na busca da atualização científica das linguagens XSC e para acompanhar o desenvolvimento desta nova ordem computacional, a presente dissertação propõe uma biblioteca com operações e funções intervalares na linguagem JAVA, a qual é descrita neste relatório, juntamente com os softwares e testes desenvolvidos. Este trabalho constitui os primeiros passos para uma futura linguagem JAVA-XSC.

Esta dissertação consta de 8 capítulos, incluídos o primeiro e o último como a introdução e a conclusão, respectivamente. Os outros como seguem.

Capítulo 2- *Matemática Intervalar.* Neste capítulo faremos uma introdução à matemática intervalar, seu histórico, conceitos, operações e funções intervalares básicas.

Capítulo 3- *Funções Intervalares.* Neste capítulo faremos uma breve revisão sobre as funções definidas em conjuntos genéricos e suas respectivas propriedades. Depois definiremos o que são funções intervalares. Descreveremos como definir funções intervalares básicas a partir de funções reais, através de imagem e avaliações intervalares. Serão também relatadas as funções trigonométricas, hiperbólicas e as suas respectivas inversas. As funções definidas nos capítulos anteriores serão desenvolvidas na linguagem proposta JAVA-XSC que será tratada no capítulo 6.

Capítulo 4- *Linguagens XSC*. Neste capítulo descreveremos as linguagens XSC, que são as linguagens apropriadas para computações cíentificas. A maioria das modernas linguagens de computação de alto nível realiza operações aritméticas simples para que o programador seja capaz de escrever expressões simples, fórmulas ou funções na notação usual da matemática.

Mas, desde os anos sessenta, tem havido pesquisas para possibilitar que os computadores suportem uma matemática mais poderosa que a aritmética de ponto flutuante. Para solucionar este problema, várias empresas de software, em cooperação com institutos de pesquisas, participaram do desenvolvimento adicional nas linguagens de programação para que estas suportassem os requisitos da alta qualidade aritmética.

Serão descritas neste capítulo algumas características, funções e operações de determinadas linguagens, tais como PASCAL-XSC, C-XSC e INTLIB, além de se apresentar um resumo de suas evoluções.

Capítulo 5-*A Linguagem JAVA*. Neste capítulo faremos um estudo desta linguagem, considerada uma nova ordem da computação, uma revolução, pois os programas de uma máquina não irão mais determinar as funções que podem ser executadas. As aplicações dependerão da rede. O computador torna-se um periférico para a Internet e para Web.

A universalidade desta linguagem de programação, única no seu gênero, foi confirmada pelo grande número de acordos de licenças, concluídos entre numerosas empresas de informática e a Sun Microsystems. Em perfeita simbiose com a filosofia da Sun, que preconiza uma tecnologia de rede aberta, o acordo de licenças JAVA permite às diversas empresas escolherem livremente suas plataformas, materiais e logísticas. E acabará com o "gargalo" da rede pois atualmente muitas aplicações estão amarradas ao sistema operacional que por sua vez está amarrado ao hardware do computador.

Portanto, conforme o estudo da linguagem JAVA neste capítulo, verificamos a necessidade fundamental dessa linguagem suportar computações científicas. Foi justamente o desejo de ter uma linguagem JAVA-XSC, que nos fez desenvolver, nesta Dissertação, uma biblioteca de matemática intervalar em JAVA. E que descreveremos no próximo capítulo.

Capítulo 6- *JAVA-XSC*. Neste capítulo desenvolveremos propriamente a proposta da dissertação. Definiremos as classes e métodos que serão criados e os testes executados, incluindo os anexos com classes e métodos para computações científicas desenvolvidos na linguagem JAVA.

Capítulo 7- *Algoritmo Intervalar: Método de Newton.* Neste capítulo utilizaremos e testaremos, com a biblioteca desenvolvida, o algoritmo clássico conhecido como Método de Newton Intervalar, em suas várias versões.

Capítulo 2

Matemática Intervalar

2.1 Introdução

Neste capítulo, introduzir-se-á Matemática Intervalar procurando-se ressaltar o seu uso, os aspectos principais e a sua importância na solução da problemática da computação científica.

A matemática intervalar busca resolver problemas que se concentram fundamentalmente em dois aspectos:

(a) na criação de um modelo computacional que reflita fidedignamente o controle e análise dos erros que ocorrem no processo computacional e,

(b) na escolha de técnicas de programação adequadas para desenvolvimento de softwares científicos buscando minimizar os erros nos resultados.

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. No entanto, a matemática intervalar busca dar suporte a estes problemas.

2.2 Uso de aproximações

Como sabemos, para computarmos qualquer objeto, é necessário representá-lo num dispositivo computacional o qual é em essência finito. Infelizmente, muitos objetos fundamentais para resolvermos problemas do dia-a-dia, não são finitamente representáveis em máquinas. Por exemplo, a solução de complexas equações reais é essencial para resolvermos muitos problemas, entretanto um número irracional não é finitamente representável.

A solução de problemas desse tipo utiliza as conhecidas aproximações que induzem a erros. Por exemplo, no caso dos números irracionais, são números racionais que estão próximos de tais objetos infinitos. Por exemplo, o número racional 3.14 aproxima o número irracional $\pi = 3.1415...$ Esta abordagem remete ao problema das limitações dessas aproximações, suscitando questões tais como se elas possuem as mesmas propriedades algébricas que os objetos que aproximam.

No caso dos números racionais, do ponto de vista algébrico, eles são perfeitos como aproximações de números reais, pois assim como os números reais, eles também constituem um corpo, possibilitando a substituição de equações de coeficientes irracionais por equações de coeficientes racionais. Por exemplo, para a equação $\pi x+\sqrt{2} = e$ podemos encontrar uma aproximação da solução $\frac{e-\sqrt{2}}{\pi}$, simplesmente trocando os coeficientes da equação por aproximações racionais, tomando-se $3.14x + 1.2 = 2.3$, e resolvendo a equação racional em vez da irracional, o que nos leva a uma aproximação da solução irracional, a saber $\frac{2.3-1.2}{3.14}$.

Entretanto, esta abordagem nos conduz ao bem conhecido **erro de aproximação** que é a distância entre o irracional e sua aproximação racional. Esse ainda não é o principal problema dessa abordagem. O problema maior reside no fato de que esse erro de aproximação não obedece à qualquer lei durante as computações, o que pode levar a distorções em uma computação. O controle desse erro de aproximação durante as computações é feito por uma computação em paralelo, requerendo esforços computacionais extras.

2.3 Erros de computação

Existem três fontes de erros em computação numérica [MLF87]:

1. **A propagação de erros nos dados e parâmetros iniciais.** Ao se tentar representar um fenômeno do mundo físico por meio de um modelo matemático, raramente se tem uma descrição correta deste fenômeno. Normalmente, são necessárias várias simplificações do mundo físico para que se tenha um modelo matemático com o qual se possa trabalhar, tomando-se apenas algumas grandezas, como por exemplo: tempo, temperatura, distância, intensidade, etc.. Estas por sua vez são obtidas de instrumentos que têm precisão limitada de modo que a incerteza destes parâmetros iniciais levará conseqüentemente a incertezas dos resultados.

   A questão de como a incerteza dos dados contribui para a incerteza da resposta pode ser ignorada ou pode ser feita uma análise profunda com simulações, ou com auxílio da experiência de pesquisador. A análise é freqüentemente difícil ou impossível, simulações são dispendiosas, especialmente em muitas dimensões, e a experiência do pesquisador deve ser considerada com cautela. Este tipo de erro é o mais sério porque não é possível torná-lo arbitrariamente pequeno via computação tradicional.

2. **Erro de arredondamento.** Para a resolução de modelos matemáticos, muitas vezes torna-se necessária a utilização de instrumentos de cálculo como computadores digitais. Sabemos que estes instrumentos de cálculo trabalham com números arredondados, ou seja, representam os números em uma forma finita de dígitos, de acordo com o seu sistema interno de representação e que tais limitações geram erros durante as computações numéricas.

3. **Erro de truncamento.** São erros provenientes da utilização de processos que deveriam ser infinitos, ou muitos grandes, para a determinação de um valor e que, por razões práticas, são truncados. Estes processos infinitos são muito utilizados na avaliação de funções matemáticas, tais como exponenciação, logaritmos, funções trigonométricas e várias outras que uma máquina pode ter.

A limitação dos dados de entrada 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 ordinária. Assim, espera-se que as técnicas intervalares forneçam garantias e que possam ser aplicadas quase que automaticamente. Uma resposta intervalar possui a garantia de sua incerteza.

2.4 Origem da aritmética intervalar

Foi por causa destes erros vistos acima que surgiram os primeiros trabalhos buscando a sua resolução. Sunaga [Sun58] fez o primeiro trabalho sobre intervalos munido de uma aritmética. Sunaga e Moore [Moo66], aparentemente, desenvolveram a mesma teoria, ao mesmo tempo, em diferentes lugares. Todavia, sempre nos referimos a Moore quando falamos sobre matemática intervalar. Os trabalhos de Moore e Sunaga permitiram que uma idéia simples se transformasse numa poderosa ferramenta para a análise de erros inerentes à computação científica. Assim, os algoritmos trabalham sobre intervalos em vez de números racionais, tendo como resultados da computação, intervalos que contém as soluções reais desejadas, cuja amplitude dá uma medida da sua qualidade.

As respostas aos problema dos erros das aproximações surgiram nos anos 50 com a aritmética intervalar proposta por Moore, que introduziu operações aritméticas de maneira a controlar esse erro de aproximação, de modo que o resultado de uma operação com intervalos é novamente um intervalo. Esta abordagem encara um intervalo fechado, $[a_1; a_2]$, como uma aproximação de todos os números reais pertencentes a ele. Como a distância entre os extremos do intervalo e o número real aproximado não excede $\frac{a_2-a_1}{2}$, o erro de aproximação no máximo é $\frac{a_2-a_1}{2}$.

Por exemplo, dados dois números reais $r_1, r_2 \in \mathbb{R}$ e duas aproximações $[a_1; a_2]$ e $[b_1; b_2]$, respectivamente, uma operação aritmética $\diamond$ é tal que $r_1 \diamond r_2 \in [a_1; a_2] \diamond [b_1; b_2]$. Portanto, o controle dos erros de aproximação é feito diretamente pela aritmética, dispensando a necessidade das meta-computações. Essa alternativa, proposta por Moore, foi chamada aritmética intervalar, fornecendo uma saída para problemas como erros de arredondamento, erros de truncamento, etc. Desse modo, em vez de representarmos um número real r por um ponto-flutuante, fazêmo-lo através de um intervalo $[a_1; a_2]$, tal que $r \in [a_1; a_2]$.

2.5 Técnicas intervalares

Atualmente tem sido muito divulgada a necessidade do uso de técnicas intervalares como uma alternativa para alcançar limites garantidos para os resultados de computações científicas, através do controle rigoroso e automático do erro do resultado. A análise intervalar está interessada em técnicas que podem ser programadas por computador, contendo em sua computação uma análise rigorosa, completa e automática destes erros.

A análise intervalar, que é uma teoria matemática com origem na década de 60, tem por objetivo responder à questão da exatidão e da eficiência que aparece na prática da Computação Científica. Apresentam-se esquemas computacionais que tratam tanto do problema da propagação do erro dos 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 da aritmética de máquinas ordinárias.

Técnicas intervalares manipulam dados e parâmetros iniciais com intervalos, com o indicativo do erro máximo presente nestes valores, antes que os mesmos sejam introduzidos no computador. 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. 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 simulações que pode substituir a execução repetida e dispendiosa.

Entretanto, obter uma resposta intervalar não garante que ela inclua algo de interesse. Atingir uma inclusão significativa requer uma fundamentação matemática cuidadosa de todos os estágios do desenvolvimento do algoritmo e suas implementações. Os algoritmos a serem desenvolvidos devem ser algoritmos intervalares e não versões intervalares de algoritmos pontuais.

2.6 Definições básicas da aritmética de Moore

Na Teoria dos Intervalos de Moore [Moo66], um **intervalo real**, $[a_1; a_2]$, onde $a_1, a_2 \in \mathbb{R}$, é definido como o subconjunto dos números reais $\mathbb{R}$ dado por $[a_1; a_2] = \{x \in \mathbb{R} \mid a_1 \leqslant x \leqslant a_2\}$, comumente chamado intervalo real fechado ou simplesmente intervalo. O conjunto de todos os intervalos reais é denotado por $\mathbb{I}(\mathbb{R})$, ou seja, $\mathbb{I}(\mathbb{R}) = \{[a_1; a_2] \mid a_1, a_2 \in \mathbb{R}, a_1 \leqslant a_2\}$. Os pontos de um intervalo real serão denotados por letras latinas minúsculas, tais como $x, y, z, ...$ Observe que a idéia principal é considerar um intervalo como uma aproximação dos reais que ele contém, embora Moore não tenha considerado a noção de aproximação no sentido de uma interpretação computacional.

Um intervalo pode ser visto como um número **representado** por um par ordenado $[a_1; a_2]$ do mesmo modo como os números racionais $\frac{a_1}{a_2}$ são representados por um par de números inteiros com $a_2 \neq 0$, ou um número complexo é representado por um par de números reais (a_1, a_2). Portanto, os intervalos passam a ter dois pontos de vista, quer como um conjunto de números reais, quer como um tipo de número. Isso nos permite construir algoritmos para operações como intersecção, união, soma, multiplicação, etc., baseados somente nos extremos dos intervalos.

2.7 Representações numérica em computadores

A representação dos números reais é uma questão importante desde os primórdios da história dos computadores. Os números reais eram inicialmente representados no sistema numérico de ponto-fixo. A passagem para a representação em ponto-flutuante iniciou-se na década de 50 e caracterizou uma evolução significativa na área da computação científica, principalmente pela melhora de exatidão nos resultados das operações.

Essa evolução prosseguiu de uma forma não muito padronizada, com o uso de representação de números em ponto-flutuante com cada vez mais dígitos na mantissa. O tamanho da mantissa era dependente da máquina e, ainda, poderia ser variado, resultando nos formatos conhecidos como: precisão simples, precisão dupla e precisão estendida.

Quando se quer falar de um sistema de ponto-flutuante, ou quando se quer caracterizá-lo, é necessário especificar: a base numérica do sistema; o número de dígitos da mantissa; o intervalo de abrangência do expoente da base e como é feita a representação destes números; como é feito o tratamento de underflow e overflow; como são efetuadas as operações aritméticas e os tipos de arredondamentos disponíveis e utilizados nas operações, pois estes dados influenciarão a análise e quantificação dos erros que os cálculos efetuados neste sistema apresentarão.

2.8 Matemática intervalar e aritmética de máquina

A matemática intervalar trata com dados na forma de intervalos numéricos e surgiu com o objetivo de automatizar a análise do erro computacional, trazendo uma nova abordagem que permite um controle de erros com limites confiáveis, além de provar a existência ou não da solução de diversas equações.

A aritmética intervalar de máquina [Lis97] ao invés de aproximar os reais para números de ponto-flutuante mais próximos (por alguma das regras de aproximação ou dos tipos de arredondamento), representa os reais por intervalos de números em ponto-flutuante, ou seja, um real x é representado por um intervalo $X = [x_1; x_2]$, onde os limites inferior(x_1) e superior (x_2) são números de máquina, tais que $x_1 \leqslant x \leqslant x_2$. A partir da aritmética intervalar de máquina são implementados os métodos intervalares para a resolução de problemas numéricos.

Assim, um intervalo de máquina ou intervalo de ponto-flutuante é um intervalo real em que seus extremos são números de máquina. Então, o conjunto dos intervalos de máquina $\mathbb{F}$ sobre F é denotado por:

$$\mathbb{F} = \{X = [x_1, x_2] \in \mathbb{I}(\mathbb{R}) \mid x_1, x_2 \in F\}.$$

Uma caracteristística muito importante desta definição é que todos os números entre x_1 e x_2, incluindo estes, pertencem ao intervalo X. Isto é, X não contém somente os números de máquina entre seus extremos mas também, todos os números reais deste intervalo. Assim, intervalos trazem continuidade para dentro do computador.

2.8.1 Arredondamentos direcionados

Há vários tipos de arredondamento. O tipo mais importante de arredondamentos são os arrendondamentos direcionados ($\triangle x, \bigtriangledown x$), fundamentais para a implementação da aritmética intervalar de máquina.

Funções de mapeamento [KuM81] são necessárias para representar os números reais em números de máquina. Seja $\mathbb{R}$ o espaço dos números reais e $\mathbb{F}$ o conjunto dos números representáveis no computador. Sendo [] a função arredondamento, a mesma teria as seguintes propriedades:

$[](x) = x, \text{ se } x \in \mathbb{F}$

$x \leqslant y \Longrightarrow [](x) \leqslant [](y)$, para todo $x, y \in \mathbb{R}$.

Arredondamento para cima

Arredondamento para cima, $\triangle x$, é a função que aproxima o número real x para o menor número de máquina, maior que o número real x. A fórmula a seguir define este tipo de arredondamento.

$\triangle x = min\{y \in \mathbb{F} \mid y \geqslant x\}$, para todo $x \in \mathbb{R}$.

Arredondamento para baixo

Arredondamento para baixo, $\bigtriangledown x$, é a função que aproxima o número real x para o maior número de máquina, menor que o número real x. A fórmula a seguir define este tipo de arredondamento.

$\bigtriangledown x = max\{y \in \mathbb{F} \mid y \leqslant x\}$, para todo $x \in \mathbb{R}$.

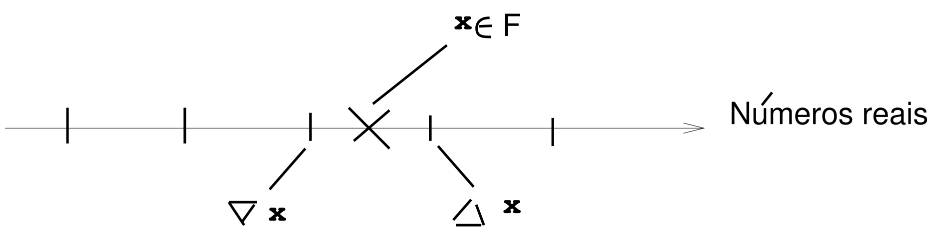

Figura 2.1: Representação dos arredondamentos direcionados

2.8.2 Padronização

Inicialmente a questão aritmética era um encargo do fabricante do computador. Cada fabricante tinha seu produto com uma aritmética computacional própria. Os softwares não eram portáveis e nem compatíveis. Isso resultava em resultados incompatíveis os quais muitas vezes produziam erros.

A representação de um número em ponto-flutuante proporcionou muitas vantagens, mas também introduziu algumas desvantagens, como a geração do problema de controle de erros nas computações numéricas que muitas vezes proporcionaram resultados totalmente errados com a aparência de serem corretos. Assim, mesmo o procedimento estando correto, o resultado perde o significado devido à inexatidão da representação numérica e de arredondamentos aplicados nas avaliações das operações e expressões aritméticas.

Outra desvantagem é a definição de métodos que avaliem ou estabeleçam a qualidade de resultados gerados por operações em ponto-flutuante. Isto abrange o conceito de validação do resultado.

Um outro problema sério foi decorrente da variedade de sistemas de ponto flutuante disponíveis no mercado, ou seja, máquinas distintas proporcionavam resultados distintos. Em algumas máquinas o resultado de uma expressão ou método era significativo e, em outras, não. Este fenômeno está relacionado com o problema denominado "análise sensitiva".

Devido a esses problemas, em 1980 o Subcomitê de Padrões da Sociedade Computacional IEEE (Institute of Electral and Eletronics Engineers) propôs um padrão de aritmética de ponto flutuante não apenas para representação, mas também para a forma como as operações são efetuadas, os tipos de arredondamentos, o tratamento de exceções como underflow, overflow e divisão por zero. O padrão IEEE-754 define as regras para a aritmética binária de ponto-flutuante e para os casos especiais que ocorrem durante a resolução de determinados cálculos ou problemas. É o que veremos a seguir.

2.8.3 Padrão IEEE-754

O padrão IEEE-754 foi definido como um conjunto de regras comercialmente possíveis para novos sistemas executarem aritmética binária de ponto-flutuante.

Os tópicos mais relevantes na formulação desse padrão foram:

a) A facilidade de movimentação de programas já existentes, entre diversos computadores com esse padrão, apresentando igual resultado.

b) Tornar mais fácil e seguro para os profissionais produzirem programas elaborados na área matemática, mesmo não tendo muita experiência.

c) Providenciar suporte direto para diagnosticar, em tempo de execução, as anomalias; tratar uniformemente as exceções; e ter uma aritmética intervalar, com um custo razoável.

d) Prover o desenvolvimento de funções elementares padrão como seno e exponencial; de aritmética de alta precisão e de funções que juntem a computação algébrica simbólica com a numérica.

e) Habilitar refinamentos e extensões, ao invés de simplesmente preveni-los.

A implementação de um sistema de ponto-flutuante, conforme o padrão IEEE-754, pode ser realizado por hardware, software ou pela combinação de hardware e software. Componentes de hardware, que requerem suporte de software para se adequarem, não podem ser considerados dentro do modelo, sem esse software.

O padrão IEEE especifica o formato:

a) De número em ponto-flutuante estendido e simples;

b) Das operações de adição, subtração, multiplicação, divisão, raiz quadrada, resto e operações de comparação;

c) Da conversão entre inteiros e formatos de ponto-flutuante;

d) Da conversão entre diferentes formatos de ponto-flutuante;

e) Da conversão entre números em ponto-flutuante simples e strings decimais;

f) Das exceções em ponto-flutuante e suas manipulações, incluindo "não-número (NaNs)".

Entretanto, o padrão IEEE não especifica os formatos:

a) De cadeias decimais (números representados na base 10) e inteiros;

b) De uma interpretação dos campos de sinal e mantissa dos não-números (NaNs);

c) De uma conversão entre os formatos estendidos de números em binário e decimal.

2.8.4 Aritmética de alta precisão

A padronização foi um passo inicial mas não é suficiente para se terem cálculos exatos e confiáveis como exige a computação científica. Em virtude disso, Kulisch e Miranker [KuM81] propuseram uma aritmética de alta exatidão composta das padronizações do IEEE-754. Nesta aritmética, os valores aproximados só diferem do valor exato por um arredondamento. Se for acrescentada à aritmética de alta exatidão à matemática, e com algoritmos baseados em teoremas da matemática intervalar, pode-se transferir para o próprio computador a tarefa de verificar o resultado. Assim, tem-se o resultado verificado automaticamente.

2.9 Operações e propriedades da aritmética de Moore

Conforme o trabalho de Moore [Moo79], uma operação aritmética binária $* \in \{+, -, \times, \div\}$ entre intervalos $A = [a_1; a_2]$ e $B = [b_1; b_2]$ é definida como $A * B = \{a * b \mid a \in A \text{ e } b \in B\}$. Pode-se observar que $\mathbb{I}(\mathbb{R})$ constitui uma estrutura algébrica que generaliza a estrutura dos reais. A seguir descreveremos operações e propriedades [ODC97].

2.9.1 Igualdade

Dizemos que dois intervalos $[a_1; a_2]$ e $[b_1; b_2]$ são iguais, denotando-se por $[a_1; a_2] = [b_1; b_2]$, se eles são o mesmo conjunto. Portanto, $[a_1; a_2] = [b_1; b_2]$ se, e somente se $a_1 = b_1$ e $a_2 = b_2$.

2.9.2 Soma

A soma de intervalos $A = [a_1; a_2]$ e $B = [b_1; b_2]$ é equivalente a somar as inequações $a_1 \leqslant a \leqslant a_2$ e $b_1 \leqslant b \leqslant b_2$ a fim de obter a inequação $a_1+b_1 \leqslant a+b \leqslant a_2+b_2$, ou, em outras palavras, somar os conjuntos $\{a \in \mathbb{R} \mid a_1 \leqslant a \leqslant a_2\}$ e $\{b \in \mathbb{R} \mid b_1 \leqslant b \leqslant b_2\}$ para obter o conjunto $\{c \in \mathbb{R} \mid a_1 + b_1 \leqslant c \leqslant a_2 + b_2\} = \{c \in \mathbb{R} \mid c = a + b \text{ para algum } a \in A \text{ e } b \in B\}$. Uma caracterização da soma de dois intervalos em termos de seus extremos é:

$$[a_1; a_2] + [b_1; b_2] = [a_1 + b_1; a_2 + b_2].$$

Exemplo: Sejam $A = [2; 3]$ e $B = [4; 5]$. Então,

$$\begin{aligned} A + B &= [2; 3] + [4; 5] \\ &= [2 + 4; 3 + 5] \\ &= [6; 8]. \end{aligned}$$

Propriedades algébricas da soma em $\mathbb{I}(\mathbb{R})$

Sejam $A, B, C \in \mathbb{I}(\mathbb{R})$ um intervalo de reais, com $A = [a_1; a_2], B = [b_1; b_2]$ e $C = [c_1; c_2]$. Então $A + B \in \mathbb{I}(\mathbb{R})$;

Associatividade: $A + (B + C) = (A + B) + C$;
Comutatividade: $A + B = B + A$;
Elemento Neutro: Seja $0 = [0; 0]$. Então $A + 0 = 0 + A = A$.

Se $A + B = A + C$ então $B = C$;
Se $A + B \subseteq A + C$ então $B \subseteq C$;

2.9.3 Pseudo inverso aditivo intervalar ou intervalo recíproco

É definido por:

$$-A = -[a_1; a_2] = \{-b \in \mathbb{R} \mid b \in [a_1; a_2]\} = [-a_2; -a_1].$$

Ou seja, o intervalo $-A$ é o conjunto de todos os números reais simétricos daqueles contidos em A.

Exemplo: Seja $A = [2; 3]$. Então, $-A = -[2; 3] = [-3; -2]$.

2.9.4 Subtração

Sejam $A = [a_1; a_2]$ e $B = [b_1; b_2]$. A subtração de intervalos é definida por:

$$\begin{aligned} A - B &= \{c \in \mathbb{R} \mid c = a - b, \text{ para algum } a \in A \text{ e } b \in B\} \\ &= \{c \in \mathbb{R} \mid c = a + (-b), \text{ para algum } a \in A \text{ e } b \in B\}. \end{aligned}$$

Uma caracterização em termos dos extremos é:

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

Exemplo: Sejam $A = [2; 3]$ e $B = [4; 5]$. Então,

$$\begin{aligned} A - B &= [2; 3] - [4; 5] \\ &= [2; 3] + (-[4; 5]) \\ &= [2; 3] + [-5; -4] \\ &= [2 - 5; 3 - 4] \\ &= [-3; -1]. \end{aligned}$$

Propriedade algébrica da subtração em $\mathbb{I}(\mathbb{R})$

Se $B - A = C - A$ então $B = C$;

2.9.5 Multiplicação

Sejam $A = [a_1; a_2]$ e $B = [b_1; b_2]$. O produto de seus intervalos é definido por: $A \cdot B = \{c \in \mathbb{R} \mid c = a \cdot b$, para algum $a \in A$ e $b \in B\}$. Uma caracterização em termos dos extremos é:

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

Exemplo: Sejam $A = [-2; 3]$ e $B = [-4; 5]$. Então,

$$\begin{aligned} A \cdot B &= [-2;3] \cdot [-4;5] \\ &= [min\{(-2)\cdot(-4), (-2)\cdot 5, 3\cdot(-4), 3\cdot 5\}; max\{(-2)\cdot(-4), (-2)\cdot 5, 3\cdot(-4), 3\cdot 5\}] \\ &= [min\{8, -10, -12, 15\}; max\{8, -10, -12, 15\}] \\ &= [-12; 15]. \end{aligned}$$

Propriedades algébricas da multiplicação em $\mathbb{I}(\mathbb{R})$

Sejam $A, B, C \in \mathbb{I}(\mathbb{R})$ intervalos reais. Então valem as propriedades de:

Associatividade: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$;
Comutatividade: $A \cdot B = B \cdot A$;
Elemento Neutro: Seja $1 = [1; 1]$. Então $A \cdot 1 = 1 \cdot A = A$;
Subdistributividade: $A \cdot (B + C) \subseteq (A \cdot B) + (A \cdot C)$.
$A \cdot 0 = 0 \cdot A = [0; 0]$.

2.9.6 Pseudo inverso multiplicativo intervalar

Seja $A \in \mathbb{I}(\mathbb{R})$ um intervalo de real, com $A = [a_1; a_2]$ e $0 \notin A$. Então,

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

Exemplo: Seja A=[-3;-2]. Então,

$$\begin{aligned} A^{-1} &= \tfrac{1}{A} \\ &= 1/[-3; -2] \\ &= \left[-\frac{1}{2}; -\frac{1}{3}\right]. \end{aligned}$$

2.9.7 Multiplicação por um escalar

Sejam $A = [a_1; a_2]$ e x um escalar. O produto de um escalar por um intervalo é definido por: $A \cdot x = \{c \in \mathbb{R} \mid c = a \cdot x,$ para algum $a \in A$ e $x \in \mathbb{R}\}$. Uma caracterização em termos dos extremos é:

$$A \cdot x = [min\{a_1 \cdot x, a_2 \cdot x\}; max\{a_1 \cdot x, a_2 \cdot x\}]$$

Exemplo: Sejam $A = [-2; 1]$ e $x = -2$. Então

$$\begin{aligned} A \cdot x &= [-2; 1] \cdot -2 \\ &= [min\{(-2)\cdot(-2), 1\cdot(-2)\}; max\{(-2)\cdot(-2), 1\cdot(-2)\}] \\ &= [min\{4, -2\}; max\{4, -2\}] \\ &= [-2; 4]. \end{aligned}$$

2.9.8 Divisão

Sejam $A = [a_1, a_2]$, $B = [b_1, b_2] \in \mathbb{I}(\mathbb{R})$ tais que $0 \notin B$. A divisão desse intervalos é definida por:

$$\frac{A}{B} = \{c \in \mathbb{R} \mid c = \frac{a}{b}, \text{ para algum } a \in A \text{ e } b \in B\}.$$

Uma caracterização em termos dos extremos é:

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

Exemplo: Sejam A=[2;3] e B=[4;5]. Então,

$$\begin{aligned} [2;3]/[4;5] &= \left[min\left\{\frac{2}{5}, \frac{2}{4}, \frac{3}{5}, \frac{3}{4}\right\}; max\left\{\frac{2}{5}, \frac{2}{4}, \frac{3}{5}, \frac{3}{4}\right\}\right] \\ &= \left[\frac{2}{5}; \frac{3}{4}\right]. \end{aligned}$$

2.9.9 Valor absoluto

Outra função muito importante é o valor absoluto de um intervalo, definida por:

$$|[a_1; a_2]| = \{|a| \mid a \in [a_1; a_2]\} = \begin{cases} [0; max\{|a_1|, |a_2|\}] & \text{, se } 0 \in [a_1; a_2] \\ [min\{|a_1|, |a_2|\}; max\{|a_1|, |a_2|\}] & \text{, senão.} \end{cases}$$

Exemplos:

$|[-4; 0]| = [0; 4]$;
$|[-3; -1]| = [1; 3]$;
$|[-2; 3]| = [0; 3]$.

2.9.10 Intervalo simétrico

Seja $A \in \mathbb{I}(\mathbb{R})$ um intervalo. Dizemos que A é um intervalo simétrico, se $-A = A$.

Exemplo: $[-2; 2], [0; 0], [-\pi; \pi]$.

Sejam $A, B, C \in \mathbb{I}(\mathbb{R})$ intervalos de reais, com B e C simétricos. Então, valem:

1. $A + B = A - B$;
2. $A \cdot B = |A| \cdot B$;
3. $A \cdot B = |A| \cdot |B| \cdot [-1; 1]$;
4. $A \cdot (B \pm C) = (A \cdot B) \pm (A \cdot C)$.

2.9.11 Intervalos degenerados

Por definição, temos intervalos da forma [a,a], que são chamados **intervalos degenerados** representando os números reais propriamente ditos.

2.9.12 Operações de conjunto sobre $\mathbb{I}(\mathbb{R})$

A noção matemática de conjunto é praticamente a mesma que se usa na linguagem comum: é o mesmo que agrupamento, classe, coleção, sistemas. No nosso caso, como um intervalo, antes que tudo, é um conjunto, é possível definir sobre $\mathbb{I}(\mathbb{R})$ operações tradicionais de conjunto, tais como: união e interseção. Mas devemos ter o cuidado que o resultado da operação seja também um intervalo.

1. **União de dois intervalos**

   A união dos intervalos (conjuntos) $A = [a_1; a_2]$ e $B = [b_1; b_2]$ deveria ser definida por $A \cup B = \{a \mid a \in A \text{ ou } a \in B\}$. No entanto, nem sempre o resultado será um intervalo (ex: $[1; 2] \cup [3; 4]$). Note que $A \cup B$ só é um intervalo, se A e B se intersetão, isto é, se $a_1 \leqslant b_2 \leqslant a_2$ ou $b_1 \leqslant a_2 \leqslant b_2$. Portanto, consideramos como união de intervalos a sequinte operação parcial:

$$A \cup B = \begin{cases} [min\{a_1, b_1\}, max\{a_2, b_2\}] & \text{, se } a_1 \leqslant b_2 \text{ e } b_2 \leqslant a_2 \text{ ou} \\ & \quad b_1 \leqslant a_2 \text{ e } a_2 \leqslant b_2 \\ \uparrow & \text{, senão.} \end{cases}$$

   Exemplos:

   $[-3; 2] \cup [0; 5] = [min\{|-3|, 0\}; max\{0, 5\}] = [-3; 5]$;

   $[2; 3] \cup [4; 5]$ é indefinido pois $\{x \mid x \in [2, 3] \text{ ou } x \in [4, 5]\}$ não é um intervalo.

2. **União convexa de dois intervalos**

   A união convexa dos intervalos $A = [a_1; a_2]$ e $B = [b_1; b_2]$, denotado por $\overline{A \cup B}$, é definida como o menor intervalo contendo ambos : A e B. Uma caracterização em termos de extremo da união convexa é definida como segue:

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

   <u>Observação :</u> No caso da união convexa, a intersecção dos dois intervalos pode ser vazia. Neste caso, o intervalo resultante será o menor intervalo que contém, simultaneamente, ambos os intervalos operados.

   Exemplos:

   $\overline{[-3; 2] \cup [0; 5]} = [min\{-3, 0\}; max\{2, 5\}] = [-3; 5];$

   $\overline{[2; 3] \cup [4; 5]} = [min\{2, 4\}; max\{3, 5\}] = [2; 5];$

   $\overline{[3; 8] \cup [-1; 10]} = [min\{3, (-1)\}; max\{8, 10\}] = [-1; 10];$

3. **Intersecção de dois intervalos**

   A intersecção dos intervalos (conjuntos) $A = [a_1; a_2]$ e $B = [b_1; b_2]$ deveria ser definida por $A \cap B = \{a \mid a \in A \text{ ou } a \in B\}$. No entanto, nem sempre o resultado será um intervalo (ex: $[1; 2] \cap [3; 4]$). Note que $A \cap B$ só é um intervalo, se A e B intersetarem, que ocorre quando $b_2 \in A$ ou $a_2 \in B$. Portanto, consideramos como intersecção de intervalos a sequinte operação parcial:

   $$A \cap B = \begin{cases} [max\{a_1, b_1\}; min\{a_2, b_2\}] & \text{, se } a_1 \leqslant b_2 \text{ e } b_2 \leqslant a_2 \text{ ou} \\ & \quad b_1 \leqslant a_2 \text{ e } a_2 \leqslant b_2 \\ \uparrow & \text{, senão.} \end{cases}$$

   Exemplos:
   $[2; 5] \cap [3; 8] = [max\{2, 3\}; min\{5, 8\}] = [3; 5].$
   $[2; 5] \cap [6; 8]$ não existe já que $[2; 5] \cap [6; 8]$ é vazio.

2.9.13 Inclusão monotônica

Sejam $A, B, C, D \in \mathbb{I}(\mathbb{R})$ intervalos de reais, tais que $A \subseteq C$ e $B \subseteq D$. Então valem as relações:

1. $A + B \subseteq C + D$;
2. $-A \subseteq -C$;
3. $A - B \subseteq C - D$;
4. $A \cdot B \subseteq C \cdot D$;
5. $1/A \subseteq 1/C$, sempre que $0 \notin C$;
6. $A/B \subseteq C/D$, sempre que $0 \notin D$.

2.9.14 Relações de ordem

É possível definir muitas relações de ordem sobre $\mathbb{I}(\mathbb{R})$. Uma delas, que é definida sobre o conjunto dos números reais, a saber:

$$[a_1; a_2] \leqslant [b_1; b_2] \quad \text{se, e somente se } a_2 \leqslant b_1.$$

Observe que essa relação reflete o fato de que um intervalo é a representação de todos os números reais dentro dele. E, portanto, um intervalo A é menor que o intervalo B, se todos os reais em A são menores que todos os reais no intervalo em B. Outra ordem que faz uso da natureza conjuntista dos intervalos é a ordem de inclusão entre conjuntos:

$$[a_1; a_2] \leqslant [b_1; b_2] \quad \text{se, e somente se } b_1 \leqslant a_1 \leqslant a_2 \leqslant b_2.$$

Kulish and Miranker [KuM86] definem a seguinte ordem:

$$[a_1; a_2] \leqslant [b_1; b_2] \quad \text{se, e somente se } a_1 \leqslant b_1 \quad \text{e } a_2 \leqslant b_2.$$

As seguintes relações de ordem são muitos importantes, pois são a base da construção da estrutura dos domínios contínuos para $\mathbb{I}(\mathbb{R})$, usadas por Acióly, para darem fundamentação computacional à matemática intervalar [Aci91]:

$$[a_1; a_2] \sqsubseteq [b_1; b_2] \quad \text{se, e somente se } a_1 \leqslant b_1 \leqslant b_2 \leqslant a_2; \text{ e}$$

$$[a_1; a_2] \ll [b_1; b_2] \quad \text{se, e somente se } a_1 < b_1 \leqslant b_2 < a_2.$$

Elas são chamadas, respectivamente, ordem de informação e ordem de aproximação. Isto porque um intervalo é considerado como uma informação parcial de algum número real. Então, se $A \subseteq B$, a informação A é mais precisa que a informação B. Por exemplo: os intervalos $[3; 4]$ e $[3,14; 3,15]$ são informações sobre π, mas o intervalo $[3,14; 3,15]$ é uma informação melhor ou mais aproximada sobre π do que a informação $[3; 4]$.

2.9.15 A topologia de $\mathbb{I}(\mathbb{R})$

Estudaremos nas próximas subseções, superficialmente algumas propriedades topológicas do espaço $\mathbb{I}(\mathbb{R})$. Tais propriedades estão baseadas principalmente nas noções de proximidade e limite. São funções que calculam particularidades geométricas dos intervalos como distância, diâmetro e ponto médio.

1. **Distância entre dois intervalos.**

   Sejam $A = [a_1; a_2]$ e $B = [b_1; b_2]$ dois intervalos de $\mathbb{I}(\mathbb{R})$. Então a distância do intervalo A ao intervalo B, é definida por

$$dist(A, B) = dist([a_1; a_2], [b_1; b_2]) = max\{|a_1 - b_1|; |a_2 - b_2|\}.$$

   Verificamos que $dist$ é um número real não negativo e observamos que $dist$ satisfaz as propriedades de uma métrica para a ordem de posição, e, portanto, $(\mathbb{I}(\mathbb{R}), dist)$ é um espaco métrico. A topologia induzida dessa métrica é denominada topologia de Moore [Moo79].

   Exemplos: Sejam $A = [2; 5]$, $B = [-2; -1]$. Então,

   1. $dist(A, A) = max\{|2 - 2|; |5 - 5|\}$
      $= 0$

   2. $dist(A, B) = max\{|2 - (-2)|; |5 - (-1)|\}$
      $= max\{4; 6\}$
      $= 6.$

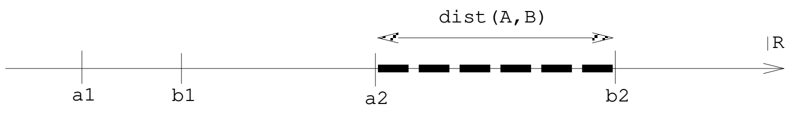

Figura 2.2: Representação geométrica da distância

2. **Diâmetro ou amplitude de um intervalo**

   O diâmetro de um intervalo serve como um indicativo da influência do erro do dado de entrada no erro do resultado final obtido. Este é um tipo de análise da qualidade do resultado que pode substituir execuções de simulações repetidas e dispendiosas. Entretanto, obter uma resposta intervalar não garante que ela inclua algo de interesse. É importante a fundamentação matemática cuidadosa de todos os estágios do desenvolvimento do algoritmo e até sua implementação. Portanto, podemos concluir que o **diâmetro ou amplitude** do intervalo é usado como uma **medida da qualidade da aproximação**. Quanto menor for a amplitude do intervalo tanto maior será a **qualidadade** da aproximação.

   Seja $A = [a_1; a_2] \in \mathbb{I}(\mathbb{R})$ um intervalo. Definimos o diâmetro do intervalo A como sendo o número real não negativo $diam(A) = a_2 - a_1$. Em [San99] o diâmetro do intervalo A é chamado de amplitude do intervalo, o qual é considerado como uma medida da qualidade do intervalo A.

Exemplo: Seja $A = [2; 5]$. Então diam(A) $= 5 - 2 = 3$.

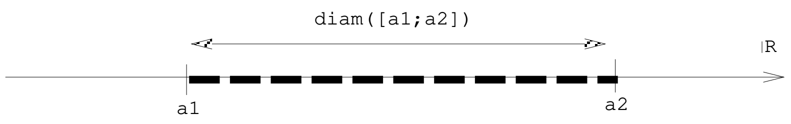

Figura 2.3: Representação geométrica do diâmetro

3. **Ponto médio de um intervalo**

   Seja $A = [a_1; a_2] \in \mathbb{I}(\mathbb{R})$ um intervalo. Definimos o ponto médio do intervalo A como sendo o número real

$$med([a_1; a_2]) = \frac{a_1 + a_2}{2}.$$

   O ponto médio de um intervalo nos dá uma visão do intervalo como um tipo de segmento.

Exemplo: Seja $A = [2; 6]$. Então $med(A) = \frac{2+6}{2} = 4$.

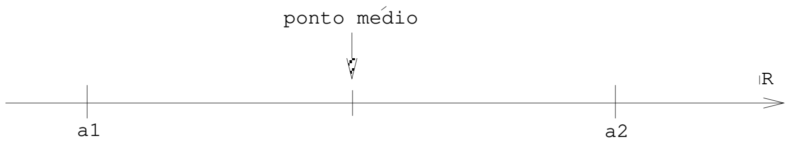

Figura 2.4: Representação geométrica do ponto médio

2.9.16 Inflacionamento

Uma exigência da matemática intervalar [Sag97] é que as operações sejam realizadas com arredondamentos direcionados nos extremos dos intervalos para cima no limite superior e para baixo no limite inferior. Isso garante que o resultado sempre estará contido dentro do intervalo resultante. Entretanto, algumas máquinas não implementam arredondamentos direcionados. A solução adotada por algumas bibliotecas é a de inflacionar os extremos sempre que uma operação seja realizada.

Seja $A = [a_1; a_2] \in \mathbb{F}$ um intervalo. Definimos o inflacionamento como um número real **predecessor** de a_1 pertencente ao conjunto dos números representáveis no computador e um número real **sucessor** pertencente ao conjunto dos intervalos da máquina $\mathbb{F}$.

$$inflacionamento(a_1, a_2) = [predecesssor(a_1); sucessor(a_2)],$$

onde o $predecessor(a_1)$ é o número de ponto flutuante da máquina imediatamente anterior ao número real a_1. Analogamente, o $sucessor(a_2)$ é o número de ponto flutuante da máquina imediatamente posterior ao número real a_2.

Capítulo 3

Funções intervalares

Este capítulo é um estudo das funções intervalares $f : \mathbb{I}(\mathbb{R}) \to \mathbb{I}(\mathbb{R})$, tais como $Exp(X)$, $\sqrt{x}$, etc.. Primeiro estudaremos as definições básicas de funções. Em seguida, veremos o que são funções intervalares, e por fim algumas funções intervalares básicas, trigonométricas, trigonométricas inversas, hiperbólicas e suas inversas.

3.1 Conceitos básicos

3.1.1 Produto cartesiano

Sejam X e Y conjuntos não vazios. **O produto cartesiano** é o conjunto de todos os pares ordenados, onde a primeira coordenada é constituída por elementos do conjunto A e a segunda coordenada é constituida por elementos do conjunto B ou seja,

$$X \times Y = \{(a, b) \mid a \in X, b \in Y\}$$

Exemplo: Sejam $X = \{1, 2, 3\}$ e $Y = \{a, b\}$. Então,
$X \times Y = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$.

3.1.2 Função

Uma **função** f de X em Y, denotada por $f : X \to Y$, é definida como um subconjunto do produto cartesiano $X \times Y$ definido da seguinte forma $\{(x, f(x)) \mid x \in X, f(x) \in Y\}$, de modo que para cada elemento x de X existe um único elemento y de Y, tal que $y = f(x)$.

Notação :

$$\begin{aligned} f : X &\to Y \\ x &\mapsto f(x) = y \end{aligned}$$

Exemplo:

$$\begin{aligned} f : \{4, 9, 16\} &\to \mathbb{R} \\ x &\mapsto f(x) = \sqrt{x} \end{aligned}$$

ou ainda, f={(4,2),(9,3),(16,4)}.

As funções $f : X \to Y$ são constituídas de quatro partes:

a) Domínio: O conjunto $X = Dom(f)$, conhecido como domínio da função, é o conjunto onde a função f é definida. No exemplo anterior, o domínio é $Dom(f) = \{4, 9, 16\}$;

b) Contra-domínio da função: O conjunto $Y = Cod(f)$, chamado de contra-domínio da função, é o conjunto onde a função f toma valores. No exemplo anterior, o contra-domínio são todos os reais;

c) Imagem da função: Seja $f : X \to Y$ uma função. Chama-se **imagem** de X pela função f, denotada por $Img(f)$, o conjunto de valores $y \in Y$ que f assume em todos os pontos x de X.

Assim,

$$Img(f) = \{y \in Y \mid y = f(x) \text{ para algum } x \in X\} \subseteq Y;$$

No exemplo anterior, a imagem é o conjunto dos valores assumidos pela função, ou seja, $Img(f) = \{2, 3, 4\}$.

d) Regra: Diz como associar, de modo bem determinado, a cada elemento $x \in X$, um único elemento $y = f(x) \in Y$, chamado **valor da função** f no ponto x. No exemplo anterior, a regra é a seguinte: a todo elemento do conjunto X se deve associar um elemento $y \in \mathbb{R}$ tal que y seja a raiz quadrada de x, isto é $y = \sqrt{x}$ ou $x = y.y$.

Observação : Não se deve confundir f com $f(x)$: enquanto f é a função, $f(x)$ é o valor que a função assume num determinado ponto x de seu domínio.

3.1.3 Igualdade entre duas funções

Duas funções $f : A \to B$ e $g : C \to D$ são iguais se, e somente se, $A = C, B = D$ e $f(x) = g(x)$, para todo elemento $x \in A$; ou seja, duas funções são iguais quando têm o mesmo domínio, o mesmo contra-domínio e a mesma lei de formação.

3.1.4 Gráfico de uma função

Seja $f : X \to Y$ uma função. O **gráfico** de uma função f é o subconjunto $Graf(f)$ do produto cartesiano $X \times Y$, formado pelos pares ordenados $(x, f(x))$, para todo $x \in X$, ou seja,

$$Graf(f) = \{(x, f(x)) \in X \times Y \mid x \in X\}$$

Exemplo:

$$\begin{array}{ll} f : X & \to \mathbb{R} \\ \quad x & \mapsto f(x) = x^2. \end{array}$$

Se $X = \{1, 2, 3\}$, então $Graf(f) = \{(1, 1), (2, 4), (3, 9)\}$.

3.1.5 Funções contínuas

Geometricamente [Spi75], uma função é contínua se seu gráfico não contém interrupções, nem saltos, nem oscilações. Esta é uma descrição informal e restrita de funções contínuas sobre conjuntos não enumeraveis que possuam um gráfico relativamente claro. Logo não é sempre possível decidir se uma função é contínua ou não observando simplesmente seu gráfico, pois é fácil enganar-se e, nem sempre seu gráfico nos diz quando ela é contínua. Daí a definição rigorosa e geral que daremos a seguir ser necessária.

Seguiremos uma definição de Bartle [Bar83]. Seja f uma função com domínio $Dom(f)$ contido em R^p[1], e contradomínio $R(f)$ contido em R^q. Em geral, não exigiremos que $Dom(f) = R^p$ ou que $p = q$. Definiremos de início a continuidade em termos de vizinhanças[2], passando depois a algumas noções equivalentes.

Se $a \in Dom(f)$, dizemos que f é **continua em** a se, para toda vizinhança V de $f(a)$, existe uma vizinhança U (dependendo de V) de a, tal que, se x é um elemento de $U \cap D(f)$, então $f(x)$ é elemento de V (figura 3.1). Se $A \subseteq D(f)$, dizemos que f é contínua em A se for contínua em cada ponto de A.

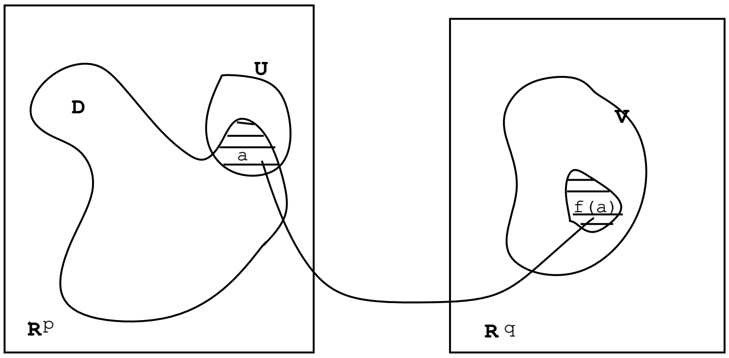

Figura 3.1: Representação gráfica de uma função contínua

3.1.6 Função intervalar

$$\text{Seja } f : \begin{array}{rl} \mathbb{X} & \to \mathbb{Y} \\ X & \mapsto f(X) \end{array} \quad \text{uma função.}$$

Se $\mathbb{X} = Dom(f) \subseteq \mathbb{I}(\mathbb{R})$ e $\mathbb{Y} = Cod(f) \subseteq \mathbb{I}(\mathbb{R})$, então, dizemos que f é uma **função intervalar de uma variável intervalar**.

[1] R^n é o produto cartesiano de R com ele mesmo n vezes $\underbrace{(R \times \ldots \times R)}_{n-vezes}$ ou seja é um espaço cartesiano n-dimensional.

[2] Se $x \in R^p$, então qualquer conjunto que contém um aberto contendo x é chamado vizinhança de x, onde um aberto é um conjunto do pontos interiores. Por exemplo, o conjunto dos pontos x tais que $0 < x < 1$ é um conjunto aberto nos reais.

Exemplo

$$\text{Seja } f: \mathbb{I}(\mathbb{R}) \to \mathbb{I}(\mathbb{R}) \quad X \mapsto f(X) = [2;3] \cdot X + [4;5] \quad \text{uma função intervalar.}$$

3.1.7 Imagem intervalar de uma função real

Sejam f uma função real, X um intervalo tal que $X \subseteq Dom(f)$ e f contínua em X. Definimos a **imagem intervalar** da função f em X (ou simplesmente **imagem** de f em X) como sendo o intervalo determinado por:

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

Observe-se que esta é uma maneira natural de se definir funções intervalares a partir de funções reais (contínuas).

Exemplo: Seja $f(x) = x^2 - x$ e $X = [0;2] \subseteq \mathbb{I}(\mathbb{R}) = Dom(f)$.

$$\begin{aligned} \text{Daí, } \mathcal{I}(f,[0;2]) &= \{f(x) \mid x \in [0;2]\} \\ &= \{x^2 - x \mid x \in [0;2]\} \\ &= [min\{x^2 - x \mid x \in [0;2]\}; max\{x^2 - x \mid x \in [0;2]\}] \\ &= [-\frac{1}{4}; 2]. \end{aligned}$$

3.1.8 Extensão intervalar de uma função real

Se $f : \mathbb{R} \to \mathbb{R}$ é uma função real, então podemos definir a sua **extensão intervalar** ou avaliação intervalar. Sejam f uma função real de variável real x e X um intervalo. A extensão intervalar de f em X, como sendo a função intervalar F(X), é definida da seguinte maneira: cada ocorrência da variável real x é substituída pela variável intervalar X, e cada operação real $(+, -, \times, \div)$ é subtituída pela respectiva operação intervalar. Assim, quando $X = [x;x]$ for um intervalo degenerado, então $F(X) = [f(x); f(x)]$.

Exemplo: Seja $f(x) = x^2 - x = x \cdot x - x$. Então $F(X) = X \cdot X - X$. Para $X = [0; 2]$, temos

$$\begin{aligned} F([0;2]) &= [0;2] \cdot [0;2] - [0;2] \\ &= [0;4] - [0;2] \\ &= [0;4] + [-2;0] \\ &= [-2;4]. \end{aligned}$$

Para $g(x) = x^2 - x = x(x-1)$, temos $G(X) = X \cdot (X - [1;1])$. Daí, para $X = [0; 2]$, temos

$$\begin{aligned} G([0;2]) &= [0;2] \cdot ([0;2] - [1;1]) \\ &= [0;2] \cdot [-1;1] \\ &= [-2;2]. \end{aligned}$$

Assim $F([0,2]) \subseteq G([0,2])$ e portanto $F \neq G$. Isto é conseqüência da propriedade de subdistributividade.

No exemplo acima, embora f e g sejam as mesmas funções, verificamos que as duas extensões intervalares, F e G podem ser diferentes, dependendo de como se expressa a função. Isto não ocorre com a Imagem Intervalar, isto é, $\mathcal{I}(f, X) = \mathcal{I}(g, X)$, pois $\mathcal{I}(f, X)$ independe da expressão da função, enquanto que $F(X)$ e $G(X)$ dependem da expressão e dos operadores envolvidos.

3.2 Funções intervalares básicas

A seguir veremos as definições formais de cada uma das funções intervalares básicas que são extensões intervalares de suas respectivas funções reais.

3.2.1 Função X^2

Definimos a função X^2 por:

$$\begin{aligned} X^2 \ : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}^+) \qquad \text{por} \\ X &\mapsto X^2 \end{aligned}$$

onde $\mathbb{I}(\mathbb{R}^+) = \{[x_1; x_2] \in \mathbb{I}(\mathbb{R}) \mid x_1 \geqslant 0\}$ e,

$$\begin{aligned} [x_1;x_2]^2 &= \mathcal{I}(x^2; [x_1;x_2]) \\ &= \{x \mid x = y^2 \text{ para algum } y \in [x_1;x_2]\} \\ &= \begin{cases} [x_1^2; x_2^2] & \text{se } x_1 > 0 \\ [x_2^2; x_1^2] & \text{se } x_2 < 0 \\ [0; (max\{|x_1|, |x_2|\})^2] & \text{se } 0 \in [x_1; x_2]. \end{cases} \end{aligned}$$

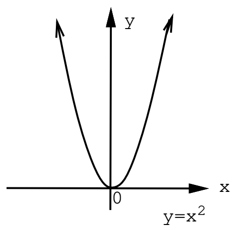

Figura 3.2: Representação gráfica da função quadrado

Exemplos: Seja $X = [-1; 2]$. Então, $[-1; 2]^2 = [-1^2; 2^2] = [0; 4]$.
Seja $X = [1; 2]$. Então, $[1; 2]^2 = [1^2; 2^2] = [1; 4]$.
Seja $X = [-2; -1]$. Então, $[-2; -1]^2 = [-1^2; -2^2] = [1; 4]$.

3.2.2 Função raiz quadrada

Definimos a função raiz quadrada intervalar por:

$$\begin{aligned} \sqrt{\ } : \mathbb{I}(\mathbb{R}^+) &\to \mathbb{I}(\mathbb{R}^+) \qquad \text{por} \\ [x_1; x_2] &\mapsto \sqrt{[x_1; x_2]} \end{aligned}$$

onde,

$$\begin{aligned} \sqrt{[x_1; x_2]} &= \mathcal{I}(\sqrt{\ }; [x_1; x_2]) \\ &= \{x \mid x = \sqrt{y} \text{ para algum } y \in X\} \\ &= [\sqrt{x_1}; \sqrt{x_2}]. \end{aligned}$$

Figura 3.3: Representação gráfica da função raiz quadrada

Exemplos: Seja $X = [9; 25]$. Então, $\sqrt{[9; 25]} = [\sqrt{9}; \sqrt{25}] = [3; 5]$.

Seja $X = [-4; 25]$. Então, $\sqrt{[-4; 25]}$ é indefinido pois $[-4; 25]$ esta fora do domínio da função.

3.2.3 Função exponencial

Definimos a função exponencial intervalar

$$\begin{aligned} Exp : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}^+) \qquad \text{por} \\ X &\mapsto Exp(X) \end{aligned}$$

onde,

$$\begin{aligned} Exp([x_1; x_2]) &= \mathcal{I}(exp, [x_1; x_2]) \\ &= \{x \in \mathbb{R} \mid x = e^y \text{ para algum } y \in [x_1; x_2]\} \\ &= [e^{x_1}; e^{x_2}]. \end{aligned}$$

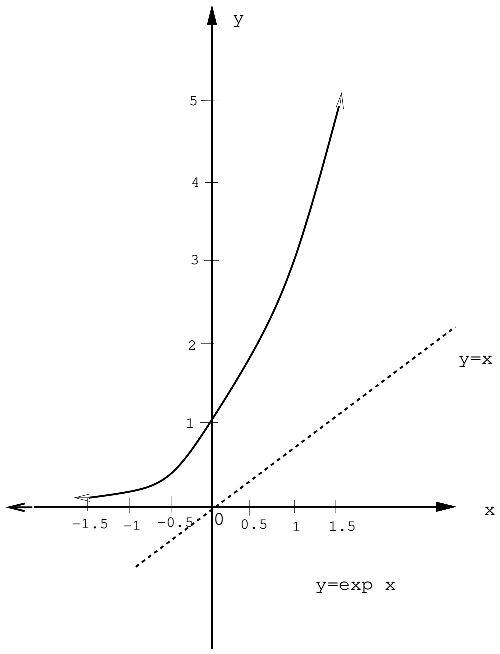

Figura 3.4: Representação gráfica da função exponencial

Exemplos: Seja $X = [0; 1]$. Então,

$$\begin{aligned} Exp(X) &= Exp([0; 1]) \\ &= [e^0; e^1] \\ &= [1; 2.718281828459045]. \end{aligned}$$

Seja $X = [-1; 1]$. Então,

$$\begin{aligned} Exp(X) &= Exp([-1; 1]) \\ &= [e^{-1}; e^1] \\ &= [0.36787944117144233; 2.718281828459045]. \end{aligned}$$

3.2.4 Função exponencial com *base* 2

Definimos a função exponencial intervalar com *base* 2

$$\begin{aligned} Exp2 : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}^+) \quad \text{por} \\ X &\mapsto Exp2(X) \end{aligned}$$

onde,

$$\begin{aligned} Exp2([x_1;x_2]) &= \mathcal{I}(exp2, [x_1;x_2]) \\ &= \{x \in \mathbb{R} \mid x = 2^y \text{ para algum } y \in [x_1;x_2]\} \\ &= [2^{x_1}; 2^{x_2}]. \end{aligned}$$

Exemplo: Seja $X = [-1; 2]$. Então,

$$\begin{aligned} Exp2(X) &= Exp2([-1;2]) \\ &= [2^{-1}; 2^2] \\ &= [0.5; 4]. \end{aligned}$$

3.2.5 Função exponencial com *base* 10

Definimos a função exponencial intervalar com *base* 10

$$\begin{aligned} Exp10 : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}^+) \quad \text{por} \\ X &\mapsto Exp10(X) \end{aligned}$$

onde,

$$\begin{aligned} Exp10([x_1;x_2]) &= \mathcal{I}(exp10, [x_1;x_2]) \\ &= \{x \in \mathbb{R} \mid x = 10^y \text{ para algum } y \in [x_1;x_2]\} \\ &= [10^{x_1}; 10^{x_2}]. \end{aligned}$$

Exemplo: Seja $X = [-2; 1]$. Então,

$$\begin{aligned} Exp10(X) &= Exp10([-2;1]) \\ &= [10^{-1}; 10^1] \\ &= [0.1; 10]. \end{aligned}$$

3.2.6 Função potência

Definimos a função potência intervalar

$$\begin{aligned} X^n : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \quad \text{por} \\ X &\mapsto X^n \end{aligned}$$

onde,

$$\begin{aligned} [x_1;x_2]^n &= \mathcal{I}(x^n; X) \\ &= \{x \in \mathbb{R} \mid x = y^n \text{ para algum } y \in X\} \\ &= \begin{cases} [0; max\{|x_1|, |x_2|\}^n] & \text{, se } n \text{ é par e } 0 \in X \\ [min\{|x_1|, |x_2|\}^n; max\{|x_1|, |x_2|\}^n] & \text{, se } n \text{ é par e } x_2 < 0 \\ [x_1^n; x_2^n] & \text{, senão.} \end{cases} \end{aligned}$$

Exemplos: Seja $X = [-2;3]$ e $n = 3$. Então,
$[-2;3]^3 = [-2^3, 3^3] = [-8;27]$ pois não é par.

Seja $X = [-3;2]$ e $n = 2$. Então,
$[9;4]^2 = [0;\{[max\{|-3|,|2|\}^2] = [0,9]$ pois é ímpar e $0 \in X$.

Seja $X = [-5;3]$ e $n = 2$. Então,
$[-5;3]^2 = [-5^2;3^2] = [9;25]$ pois é par mas $x_2 > 0$.

Seja $X = [2;4]$ e $n = 2$. Então,
$[2;4]^2 = [2^2;4^2] = [4;16]$ pois é par mas $x_2 > 0$.

3.2.7 Função logaritmo

Definimos a função logaritmo intervalar:

$$\begin{array}{rll} Ln : \mathbb{I}(\mathbb{R}^+) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ X & \mapsto Ln(X) & \end{array}$$

onde,

$$\begin{aligned} Ln([x_1;x_2]) &= \mathcal{I}(ln,[x_1;x_2]) \\ &= \{x \in \mathbb{R} \mid x = ln(y) \text{ para algum } y \in [x_1;x_2]\} \\ &= [ln(x_1);ln(x_2)]. \end{aligned}$$

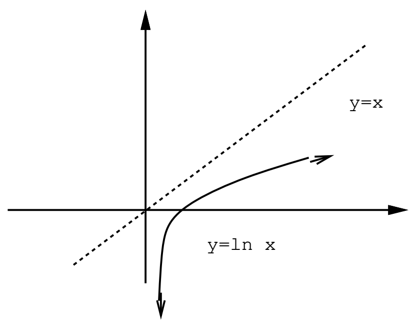

Figura 3.5: Representação gráfica da função logaritmo

Exemplo: Seja $X = [1;e]$. Então,

$$\begin{aligned} Ln(X) &= Ln([1;e]) \\ &= [ln(1), ln(e)] \\ &= [0;1]. \end{aligned}$$

3.2.8 Função logaritmo de *base* 2

Definimos a função logaritmo intervalar de *base* 2:

$$\begin{array}{lll} Log_2 : \mathbb{I}(\mathbb{R}^+) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Log_2(X) & \end{array}$$

onde,

$$\begin{array}{ll} Log_2([x_1; x_2]) & = \mathcal{I}(log_2, [x_1; x_2]) \\ & = \{x \in \mathbb{R} \mid x = log_2(y) \text{ para algum } y \in [x_1; x_2]\} \\ & = [log_2(x_1); log_2(x_2)]. \end{array}$$

Exemplo: Seja $X = [1; 4]$. Então,

$$\begin{array}{ll} Log_2(X) & = Log_2([1; 4]) \\ & = [log_2(1); log_2(4)] \\ & = [0; 2]. \end{array}$$

3.2.9 Função logaritmo de *base* 10

Definimos a função logaritmo intervalar de *base* 10:

$$\begin{array}{lll} Log_{10} : \mathbb{I}(\mathbb{R}^+) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Log_{10}(X) & \end{array}$$

onde,

$$\begin{array}{ll} Log_{10}([x_1; x_2]) & = \mathcal{I}(log_{10}, [x_1; x_2]) \\ & = \{x \in \mathbb{R} \mid x = log_{10}(y) \text{ para algum } y \in [x_1; x_2]\} \\ & = [log_{10}(x_1); log_{10}(x_2)]. \end{array}$$

Exemplo: Seja $X = [2; 3]$. Então,

$$\begin{array}{ll} Log_{10}(X) & = Log_{10}([2; 3]) \\ & = [log_{10}(2); log_{10}(3)] \\ & = [0.30102999566398114; 0.4771212547196623]. \end{array}$$

3.3 Funções trigonométricas intervalares

As seis funções trigonométricas, *seno*, *co−seno*, *tangente*, *co−tangente*, *secante*, *co−secante* e (abreviadas, *sen*, *cos*, *tan*, *sec*, *cot* e *csc*, respectivamente) são bastante familiares. A forma de onda das funções seno e co-seno são responsáveis pela grande utilidade nas aplicações da matemática. Além disso, muitos fenômenos naturais, desde ondas eletromagnéticas até o fluxo e refluxo das marés são periódicos, e, por isso, estas funções são indispensáveis às descrições matemáticas ou modelos associados a tais fenômenos.

3.3.1 Função seno

Definimos a função seno intervalar.

$$\begin{array}{rll} Sen: \mathbb{I}(\mathbb{R}) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ X & \mapsto Sen(X) & \end{array}$$

onde,

$$\begin{array}{ll} Sen([x_1;x_2]) & = \mathcal{I}(sen;[x_1;x_2]) \\ & = \{x \mid x = sen(y) \text{ para algum } y \in [x_1;x_2]\} \\ & = \begin{cases} [-1;1] & \text{, se } 2\pi \leq x_2 - x_1 \text{ ou } \pi < x_2 - x_1 < 2\pi \text{ e } sen(x_1 + \frac{x_2 - x_1}{2}) = 0 \\ [minM;1] & \text{, se } \pi \leq x_2 - x_1 < 2\pi \text{ e } sen(x_1 + \frac{x_2 - x_1}{2}) > 0 \\ [-1;maxM] & \text{, se } \pi \leq x_2 - x_1 < 2\pi \text{ e } sen(x_1 + \frac{x_2 - x_1}{2}) < 0 \\ [minM;1] & \text{, se } (x_2 - x_1) < \pi \text{ e } x_1 \leq k\frac{\pi}{2} \leq x_2 \text{ para algum } k \in Z, \text{ tal que} \\ & \quad k-1 \text{ seja múltiplo de 4} \\ [-1;maxM] & \text{, se } (x_2 - x_1) < \pi \text{ e } x_1 \leq k\frac{\pi}{2} \leq x_2 \text{ para algum } k \in Z, \text{ tal que} \\ & \quad k+1 \text{ seja múltiplo de 4} \\ [minM;maxM] & \text{, senão.} \end{cases} \end{array}$$

e $M = \{sen(x_1), sen(x_2)\}$.
Observe que se existir um tal k, então $k = \lfloor \frac{x_1}{\frac{\pi}{2}} \rfloor + 1 = \lfloor \frac{2x_1}{\pi} \rfloor + 1$ também satisfaz a condição.

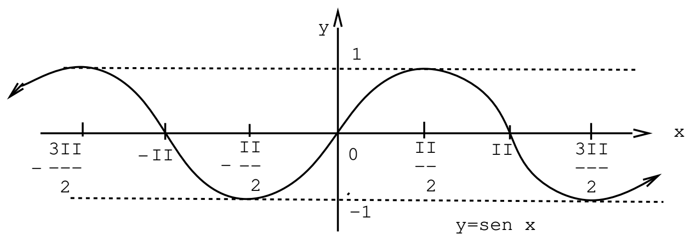

Figura 3.6: Representação gráfica da função seno

Exemplos: Seja $X = [0;1]$. Então,

$$\begin{array}{ll} Sen(X) & = \mathcal{I}(sen;[0;1]) \\ & = [min\{sen(0), sen(1)\}; max\{sen(0), sen(1)\}] \text{ pois só satisfaz a última condição do seno} \\ & = [min\{0, 0.8414709848078965\}; max\{0, 0.8414709848078965\}] \\ & = [0; 0.8414709848078965]. \end{array}$$

Seja $[1;9]$. Então, $Sen([1;9]) = [-1;1]$ pois $2\pi \leq 9 - 1$.

Seja $X = [2; 6]$. Então,

$$\begin{aligned} Sen(X) &= \mathcal{I}(sen; [2;6]) \\ &= [-1; max\{sen(6), sen(2)\}] \text{ pois } \pi \leq 6-2 < 2\pi \text{ e } sen(2 + \tfrac{6-2}{2}) < 0 \\ &= [-1; max\{-0.2794154981989259, 0.9092974268256817\}] \\ &= [-1; 0.9092974268256817]. \end{aligned}$$

Seja $X = [4; 5]$. Então,

$$\begin{aligned} Sen(X) &= \mathcal{I}(sen; [4;5]) \\ &= [-1; Max\{sen(4), sen(5)\}] \text{ pois } (5-4) = 1 \text{ é menor que } \pi, \\ &\qquad\qquad k = \lfloor \tfrac{8}{\pi} \rfloor + 1 = 3 \text{ e } k+1 \text{ é múltiplo de } 4 \\ &= [-1; Max\{-0.75680249530792819, -0.95892427466313843\}] \\ &= [-1; -0.75680249530792819]. \end{aligned}$$

3.3.2 Função co-seno

Definimos a função co-seno intervalar

$$\begin{aligned} Cos : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \qquad \text{por} \\ X &\mapsto Cos(X) \end{aligned}$$

onde,

$$\begin{aligned} Cos(X) &= \mathcal{I}(cos; X) \\ &= \{y \mid y = cos(x) \text{ para algum } x \in X\} \\ &= \{y \mid y = sen(x + \tfrac{\pi}{2}) \text{ para algum } x \in X\} \\ &= Sen(X + [\tfrac{\pi}{2}; \tfrac{\pi}{2}]). \end{aligned}$$

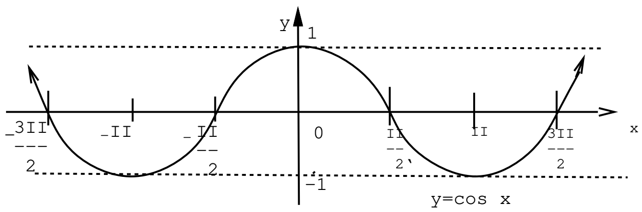

Figura 3.7: Representação gráfica da função co-seno

Exemplos: Seja $X = [0; \frac{\pi}{2}]$. Então,

$$\begin{aligned} Cos(X) &= \mathcal{I}(cos; [0; \tfrac{\pi}{2}]) \\ &= Sen([0; \tfrac{\pi}{2}] + [\tfrac{\pi}{2}; \tfrac{\pi}{2}]) \\ &= Sen([\tfrac{\pi}{2}; \pi]) \\ &= [min\{sen(\tfrac{\pi}{2}), sen(\pi)\}; 1] \\ &= [0; 1]. \end{aligned}$$

Seja $X = [1;9]$. Então,

$$\begin{aligned}
Cos(X) &= \mathcal{I}(cos;[0;\tfrac{\pi}{2}]) \\
&= Sen([1;9]) \\
&= Sen([1;9] + [\tfrac{\pi}{2};\tfrac{\pi}{2}]) \\
&= Sen([1+\tfrac{\pi}{2}; 9+\tfrac{\pi}{2}]) \\
&= [-1;1] \text{ pois } (9+\tfrac{\pi}{2}) - (1+\tfrac{\pi}{2}) = 8 \text{ e } 8 \geq 2\pi.
\end{aligned}$$

Seja $X = [2;5]$. Então,

$$\begin{aligned}
Cos(X) &= \mathcal{I}(cos;[2;5]) \\
&= Sen([\tfrac{\pi}{2};\tfrac{\pi}{2}] + [2;5]) \\
&= Sen([2+\tfrac{\pi}{2}; 5+\tfrac{\pi}{2}]) \\
&= Sen([-1; maxM]) \text{ pois } \tfrac{\pi}{2} \leq (5+\tfrac{\pi}{2}) - (2+\tfrac{\pi}{2}) < \pi \text{ e} \\
&\qquad\qquad\qquad\qquad sen(2+\tfrac{\pi}{2} + \tfrac{(5+\frac{\pi}{2})-(2+\frac{\pi}{2})}{2}) < 0 \\
&= Sen([-1; max\{sen(2+\tfrac{\pi}{2}), sen(5+\tfrac{\pi}{2})\}]) \\
&= Sen([-1; max\{-0.4161468365471423, 0.2836621854632262\}]) \\
&= [-1; 0.2836621854632262]
\end{aligned}$$

Seja $X = [4;5]$. Então,

$$\begin{aligned}
Cos(X) &= \mathcal{I}(cos;[4;5]) \\
&= Sen([\tfrac{\pi}{2};\tfrac{\pi}{2}] + [4;5]) \\
&= Sen([4+\tfrac{\pi}{2}; 5+\tfrac{\pi}{2}]) \\
&= [minM; maxM] \text{ pois só satisfaz a última condição do seno} \\
&\qquad\qquad\qquad\qquad \text{que é a excessão (senão)} \\
&= [-0.6536436208636119; 0.2836621854632262].
\end{aligned}$$

3.3.3 Função tangente

Definimos a função tangente intervalar

$$\begin{aligned}
Tan : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \qquad \text{por} \\
X &\mapsto Tan(X)
\end{aligned}$$

onde,

$$\begin{aligned}
Tan([x_1;x_2]) &= \mathcal{I}(tan;[x_1;x_2]) \\
&= \{y \mid y = tan(x) \text{ para algum } x \in [x_1;x_2]\} \\
&= \{y \mid y = \tfrac{sen(x)}{cos(x)} \text{ para algum } x \in [x_1;x_2]\} \\
&= \frac{Sen([x_1;x_2])}{Cos([x_1;x_2])} \\
&= \begin{cases} \uparrow & , \text{ se } k\frac{\pi}{2} \in [x_1;x_2] \text{ para algum } k \in \mathbb{Z} \text{ ímpar} \\ [tan(x_1); tan(x_2)], & \text{senão.} \end{cases}
\end{aligned}$$

Observe que se existir um tal k, então $k = \lfloor\{\frac{x_1}{\frac{\pi}{2}}\}\rfloor + 1$ também satisfaz a condição.

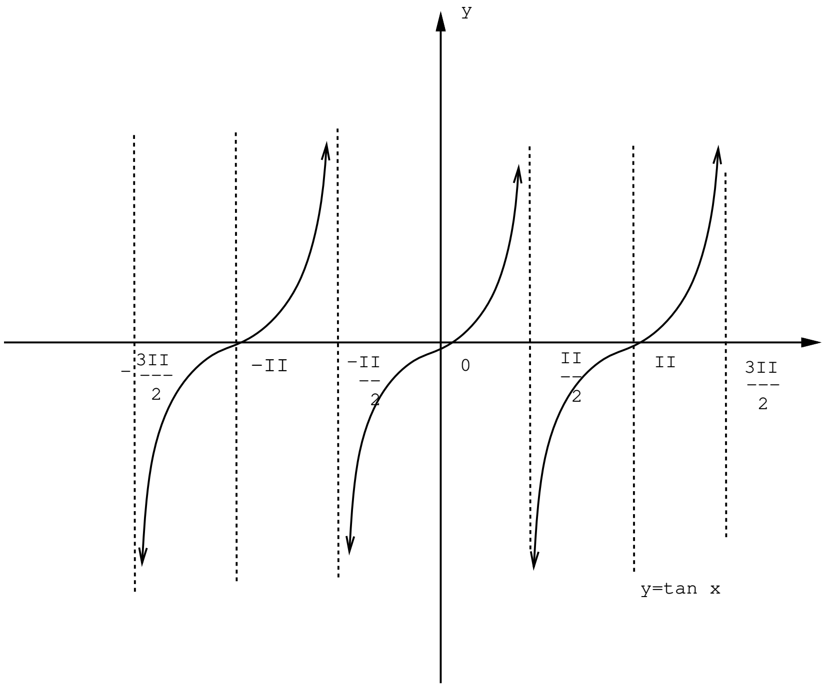

Figura 3.8: Representação gráfica da função tangente

Exemplos: Seja X=[0;1]. Então,

$$\begin{aligned} Tan(X) &= \mathcal{I}(tan;[0;1]) \\ &= \left[\frac{Sen([0;1])}{Cos([0;1])}\right] \\ &= [tan(0);tan(1)] \\ &= [0;1.5574077246549023]. \end{aligned}$$

Seja $X = [1;9]$. Então, $Tan([1;9])$ diverge, pois $9-1 > \pi$ e portanto necessariamente a tangente para algum $x \in [1;9]$ diverge. Por exemplo ocorre com $x = k\frac{\pi}{2} \in [1;9]$ para $k = 3$ e $k = 5$.

Seja X=[2;4]. Então,

$$\begin{aligned} Tan(X) &= \mathcal{I}(cot;[2;4]) \\ &= \left[\frac{Sen([2;4])}{Cos([2;4])}\right] \\ &= [tan(2);tan(4)] \\ &= [-2.1850339863261519;1.1578212823495777]. \end{aligned}$$

Seja X=[4;5]. Então, Tan([4;5]) diverge, pois $k = 3$ é ímpar e $3\frac{\pi}{2} \in [4;5]$.

3.3.4 Função co-tangente

Definimos a função co-tangente intervalar

$$\begin{array}{ll} Cot : \mathbb{I}(\mathbb{R}) & \to \mathbb{I}(\mathbb{R}) \qquad \text{por} \\ \quad X & \mapsto Cot(X) \end{array}$$

onde,

$$\begin{aligned} Cot([x_1;x_2]) &= \mathcal{I}(cot;[x_1;x_2]) \\ &= \{y \mid y = cot(x) \text{ para algum } x \in [x_1;x_2]\} \\ &= \{y \mid y = \tfrac{1}{tan(x)} \text{ para algum } x \in [x_1;x_2]\} \\ &= \frac{1}{Tan([x_1;x_2])} \\ &= \begin{cases} \uparrow & , \quad \text{se } k\pi \in [x_1;x_2] \text{ para algum } k \in \mathbb{Z} \\ [cot(x_1);cot(x_2)], & \text{senão.} \end{cases} \end{aligned}$$

Observe que, se existir um tal k, então $k = \lfloor\{\frac{x_1}{\pi}\}\rfloor + 1$ também satisfaz a condição.

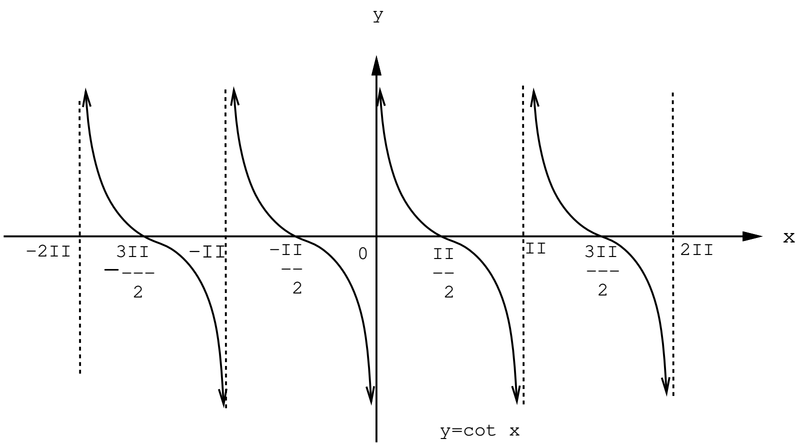

Figura 3.9: Representação gráfica da função co-tangente

Exemplos: Seja $X = [1;2]$. Então,

$$\begin{aligned} Cot(X) &= \mathcal{I}(cot;[1;2]) \\ &= [cot(1);cot(2)] \\ &= [-0.5765755436028577; 0.6420926159343306]. \end{aligned}$$

Seja X=[2.5;3.5]. Então, Cot[2.5;3.5] diverge, pois $k\pi \in [2.5;3.5]$ para algum $k \in \mathbb{Z}$ $(k = 1)$.

Seja X=[1;9]. Então, Cot[-0.5;0.5] diverge, pois $k\pi \in [-0.5;0.5]$ para algum $k \in \mathbb{Z}$ $(k = 0)$.

3.3.5 Função secante

Definimos a função secante intervalar

$$\begin{array}{lll} Sec: \mathbb{I}(\mathbb{R}) & \rightarrow \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Sec(X) & \end{array}$$

onde,

$$\begin{array}{ll} Sec([x_1;x_2]) & = \mathcal{I}(sec;[x_1;x_2]) \\ & = \{y \mid y = sec(x) \text{ para algum } x \in [x_1;x_2]\} \\ & = \{y \mid y = \frac{1}{cos(x)} \text{ para algum } x \in [x_1;x_2]\} \\ & = \dfrac{1}{cos([x_1;x_2])} \\ & = \begin{cases} \uparrow & \text{, se } \exists\ k_1 \in \mathbb{Z} \mid k_1 \text{ é ímpar e } k_1\frac{\pi}{2} \in [x_1,x_2] \\ [min\{sec(x_1),sec(x_2)\};-1] & \text{, se } \exists\ k \in \mathbb{Z} \mid k \text{ é ímpar e } k\pi \in [x_1,x_2] \\ [1;max\{sec(x_1),sec(x_2)\}] & \text{, se } \exists\ k \in \mathbb{Z} \mid k \text{ é par e } k\pi \in [x_1,x_2] \\ [min\{sec(x_1),sec(x_2)\};max\{sec(x_1),sec(x_2)\}] & \text{, senão.} \end{cases} \end{array}$$

Observe que se existirem tais k_1 e k, então $k_1 = \lfloor\{\frac{x_1}{\frac{\pi}{2}}\}\rfloor + 1$ e $k = \lfloor\{\frac{x_1}{\pi}\}\rfloor + 1$ também satisfazem as condições.

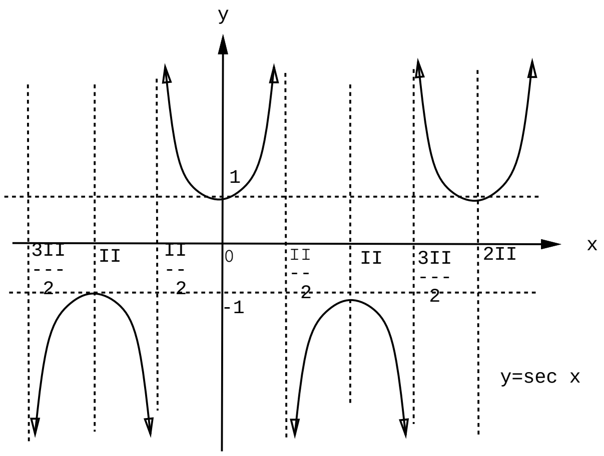

Figura 3.10: Representação gráfica da função secante

Exemplos: Seja $X = [-1; 1.5]$. Então,

$$\begin{array}{ll} Sec(X) & = \mathcal{I}(sec;[-1;1.5]) \\ & = [1; max\{sec(-1), sec(1.5)\}] \text{ pois } \exists\ k \in \mathbb{Z}\ (k=0) \text{ tal que} \\ & \qquad\qquad\qquad\qquad\qquad\qquad k \text{ é par e } k\pi \in [-1;1.5] \\ & = [1; max\{1.8508157176809258, 14.136832902969903\}] \\ & = [1; 14.136832902969903]. \end{array}$$

Seja $X = [0; \pi]$. Então, $Sec([0; \pi])$ diverge, pois $\exists\ k \in \mathbb{Z}$ $(k_1 = 1)$ tal que k_1 é ímpar e $k_1\frac{\pi}{2} \in [0; \pi]$.

Seja $X = [1.7; 4.5]$. Então,

$$
\begin{aligned}
Sec(X) &= \mathcal{I}(sec; [1.7; 4.5]) \\
&= [min\{sec(1.7), sec(4.5)\}; -1] \text{ pois } \exists\ k \in \mathbb{Z}\ (k = 1) \text{ tal que} \\
&\qquad\qquad k \text{ é ímpar e } k\pi \in [1,7; 4.5] \\
&= [min\{sec(-7.7612939960503552, -4.7439275483683251)\}; -1] \\
&= [-7.7612939960503552; -1].
\end{aligned}
$$

Seja $X = [3.5; 6.0]$. Então, $Sec([3.5; 6.0])$ diverge, pois $\exists\ k_1 \in \mathbb{Z}\ (k = 3)$ tal que k_1 é ímpar e $k_1\frac{\pi}{2} \in [3.5; 6.0]$.

Seja $X = [1.4; 4.0]$. Então, $Sec([1.4; 4.0])$ diverge, pois $\exists\ k_1 \in \mathbb{Z}\ (k = 1)$ tal que k_1 é ímpar e $k_1\frac{\pi}{2} \in [1.4; 4.0]$.

Seja $X = [5.5; 6.0]$. Então,

$$
\begin{aligned}
Sec(X) &= \mathcal{I}(sec; [5.5; 6.0]) \\
&= [minM; maxM] \text{ pois só satisfaz a última condição} \\
&\qquad\qquad \text{que é a excessão} \\
&= [1.0414819265951076; 1.4110944706229345]
\end{aligned}
$$

onde M={1.0414819265951076;1.4110944706229345}.

3.3.6 Função co-secante

Definimos a função co-secante intervalar

$$
\begin{aligned}
Csc : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \qquad \text{por} \\
X &\mapsto Csc(X)
\end{aligned}
$$

onde,

$$
\begin{aligned}
Csc() &= \mathcal{I}(csc; [x_1; x_2]) \\
&= \{y \mid y = csc(x) \text{ para algum } x \in [x_1; x_2]\} \\
&= \{y \mid y = \tfrac{1}{sen(x)} \text{ para algum } x \in [x_1; x_2]\} \\
&= \frac{1}{Sen(X)} \\
&= Sec([x_1; x_2] - [\tfrac{\pi}{2}; \tfrac{\pi}{2}]).
\end{aligned}
$$

Exemplos: Seja $X = [-1; 1]$. Então,

$$
\begin{aligned}
Csc(X) &= \mathcal{I}(csc; [-1; 1]) \\
&= Sec([-1,0; 1,0] - [\tfrac{\pi}{2}; \tfrac{\pi}{2}]) \\
&= Sec([-1 - \tfrac{\pi}{2}; 1 - \tfrac{\pi}{2}]) \text{ diverge pois } \exists\ \ k_1 \in \mathbb{Z}\ (k = -1) \text{ tal que} \\
&\qquad\qquad k_1 = 1 \text{ é ímpar e } k_1\tfrac{\pi}{2} \in [-1 - \tfrac{\pi}{2}, 1 - \tfrac{\pi}{2}].
\end{aligned}
$$

Seja $X = [-5.0; -4.0]$. Então,

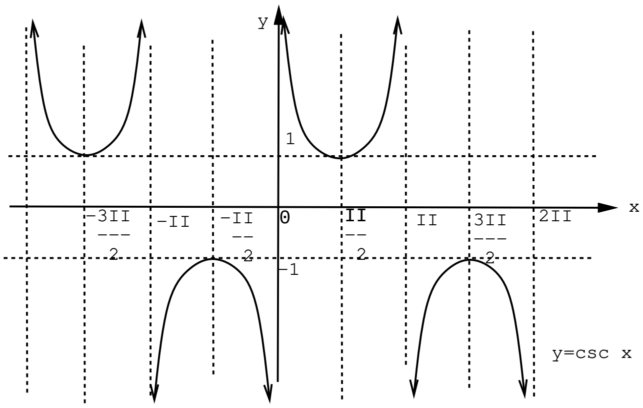

Figura 3.11: Representação gráfica da função co-secante

$$
\begin{aligned}
Csc(X) &= \mathcal{I}(csc;[-5;-4])\\
&= Sec([-5,-4,]-[\tfrac{\pi}{2};\tfrac{\pi}{2}])\\
&= [1;max\{sec(-5-\tfrac{\pi}{2};sec(-4-\tfrac{\pi}{2})\}] \text{ pois } \exists\ k\in\mathbb{Z}\ (k=-1) \text{ tal que}\\
&\qquad\qquad \text{k é ímpar e } k\pi\in[-5-\tfrac{\pi}{2};-4-\tfrac{\pi}{2}]\\
&= [1;max\{sec(1.04283521277140576, 1.3213487088109024)];\\
&= [1;1.3213487088109024].
\end{aligned}
$$

Seja $X=[4.0;6.0]$. Então,

$$
\begin{aligned}
Csc(X) &= \mathcal{I}(csc;[4;6])\\
&= Sec([4,6]-[\tfrac{\pi}{2};\tfrac{\pi}{2}])\\
&= [min\{sec(4-\tfrac{\pi}{2};sec(6-\tfrac{\pi}{2});-1\}] \text{ pois } \exists\ k\in\mathbb{Z}\quad (k=2) \text{ tal que}\\
&\qquad\qquad k \text{ é par e } k\pi\in[4-\tfrac{\pi}{2};6-\tfrac{\pi}{2}]\\
&= [min\{-1.3213487088109024, -3.5788995472544065\};-1];\\
&= [-3.5788995472544065;-1].
\end{aligned}
$$

3.4 Funções trigonométricas inversas

A representação gráfica da função inversa f^{-1} de uma função f é obtida pela simetria do gráfico de f em relação à reta $y=x$ (figura 3.12). Entretanto, para que f tenha realmente uma função inversa, a simetria do gráfico de f em relação à reta $y=x$ não pode passar sob si mesma.

Quanto à notação utilizada nestas funções, note que, embora $sen^2\ x$ signifique $(sen\ x)^2$ e $sen^{3/2}\ x$ signifique $(sen\ x)^{3/2}$, $sen^{-1}x$ não significa $(sen\ x)^{-1}$. Para evitarmos qualquer confusão possível entre $sen^{-1}x$ de um lado e $1/sen\ x$ de outro, adotaremos como notação **"arco seno x" ou, resumindo, "arcsen x"** - significando **"o arco (isto é, o ângulo) cujo seno é x"** ao invés de $sen^{-1}x$.

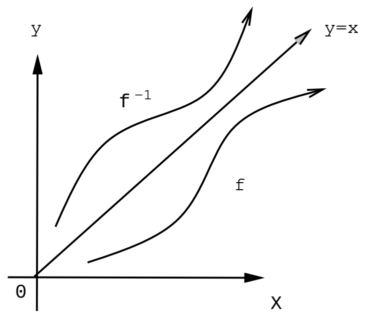

Figura 3.12: Representação gráfica de uma função juntamente com sua inversa

3.4.1 Função arco seno

A representação gráfica da função seno forma um padrão repetitivo porque a função seno é periódica (figura 3.13 (a)); Quando este gráfico é refletido sobre a reta $y = x$, o gráfico resultante passa repetidamente sob si mesmo. Portanto, a função seno não é invertível.

Assim, definimos a função arco seno intervalar

Note-se, entretanto, que a função seno é monotonamente crescente no intervalo $[-\frac{\pi}{2}, \frac{\pi}{2}]$ (figura 3.13 (a)); logo, *neste intervalo*, ela tem uma inversa (figura 3.13 (b)). A função cujo gráfico aparece na (figura 3.13 (b)), é denominada "função inversa do seno" e é denotada por $arcsen$ ou sen^{-1}. A terminologia neste caso não é totalmente correta, pois a função seno (definida em todo $\mathbb{R}$) não tem inversa. Logo $arcsen$ realmente não é a inversa da função seno, mas é a inversa da porção da função seno cujo gráfico está entre $x = \frac{-\pi}{2}$ e $x = \frac{\pi}{2}$, inclusive. Praticamente todos utilizam esta terminologia e notação, apesar do fato de que ela poderia ser enganadora.

Então, estabeleceremos a seguinte definição:

$y = arcsen\ x$ se, e somente se, $x = sen\ y$ e $-\pi/2 \leqslant y \leqslant \pi/2$.

O domínio de $arcsen$ é $[-1; 1]$ e sua imagem é $[-\frac{\pi}{2}; \frac{\pi}{2}]$(figura 3.14). Logo o domínio do $Arcsen$ é $\mathbb{I}([-1; 1]) = \{[x_1; x_2] \in \mathbb{I}(\mathbb{R}) \mid -1 \leqslant x_1$ e $x_2 \leqslant 1\}$. Analogamente, a imagem de $arcsen$ é $\mathbb{I}([-\frac{\pi}{2}; \frac{-\pi}{2}]) = \{[x_1, x_2] \in \mathbb{I}(\mathbb{R}) \mid -\frac{\pi}{2} \leqslant x_1$ e $x_2 \leqslant \frac{\pi}{2}\}$.

$$\begin{array}{lll} Arcsen : \mathbb{I}([-1; 1]) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Arcsen(X) & \end{array}$$

onde,

$$\begin{aligned} Arcsen([x_1; x_2]) &= \mathcal{I}(arcsen; [x_1; x_2]) \\ &= \{y \mid y = arcsen\ x \text{ para algum } x \in [x_1; x_2]\} \\ &= \{y \mid x = sen\ y \text{ para algum } x \in [-1; 1] \text{ e } -\tfrac{\pi}{2} \leqslant y \leqslant \tfrac{\pi}{2}\} \\ &= [arcsen(x_1); arcsen(x_2)]. \end{aligned}$$

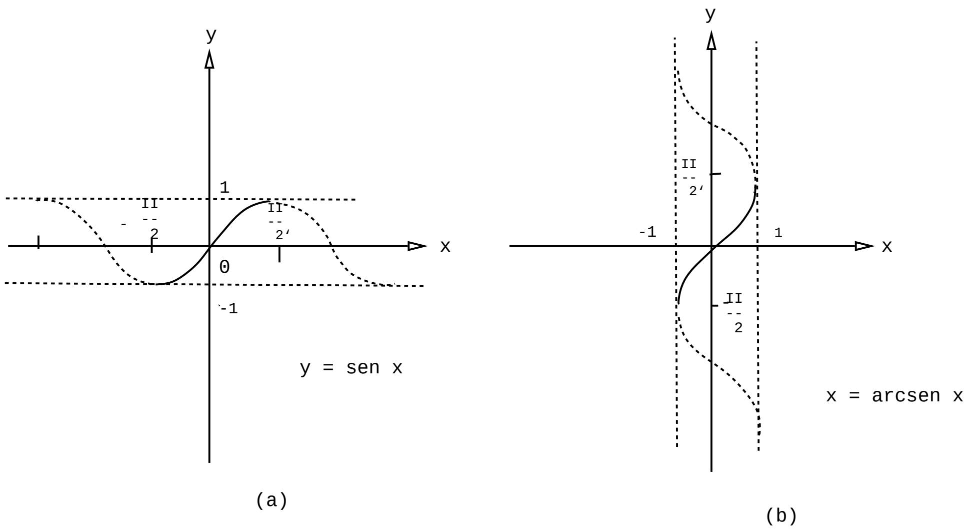

Figura 3.13: Representação gráfica das funções seno e arco seno

Exemplos: Seja $X = [-0.5; 1]$. Então,

$$\begin{aligned} Arcsen(X) &= \mathcal{I}(arcsen; [-0.5; 1]) \\ &= [arcsen(-0.5); arcsen(1)] \\ &= [-0.5235987755982989; 1.5707963267948966]. \end{aligned}$$

Seja $X = [-0.5; \frac{\pi}{2}]$. Então, $Arcsen([-0.5; \frac{\pi}{2}])$ diverge pois neste intervalo a função seno não tem inversa.

<u>Observação :</u>

As cinco funções trigonométricas restantes $co-seno$, $tangente$, $co-tangente$, $secante$ e $co-secante$ também são periódicas; logo, não são invertíveis em todo $\mathbb{R}$. Entretanto, restringindo cada uma dessas funções a intervalos convenientes, nos quais elas sejam monótonas, podemos definir as inversas correspondentes, assim como fizemos para a função seno.

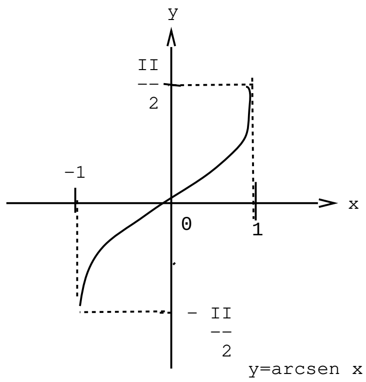

Figura 3.14: Representação gráfica da função arco seno

3.4.2 Função arco co-seno

Definimos a função arco co-seno intervalar

$$\begin{array}{lll} Arccos : \mathbb{I}([-1;1]) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Arccos(X) & \end{array}$$

onde,

$$\begin{aligned} Arccos([x_1;x_2]) &= \mathcal{I}(arccos;[x_1;x_2]) \\ &= \{y \mid y = arccos(x) \text{ para algum } x \in [x_1;x_2]\} \\ &= \{y \mid x = cos(y) \text{ para algum } x \in [x_1;x_2] \text{ e } 0 \le y \le \pi\} \\ &= [arccos(x_2); arccos(x_1)]. \end{aligned}$$

O domínio de $arccos$ é $[-1;1]$ e sua imagem é $[0;\pi]$. Logo o domínio do $Arccos$ é $\mathbb{I}([-1;1]) = \{[x_1;x_2] \in \mathbb{I}(\mathbb{R}) \mid -1 \leqslant x_1 \text{ e } x_2 \leqslant 1\}$. Analogamente a imagem de $Arccos$ é $\mathbb{I}([0;\pi]) = \{[x_1,x_2] \in \mathbb{I}(\mathbb{R}) \mid 0 \leqslant x_1 \text{ e } x_2 \leqslant \pi\}$.

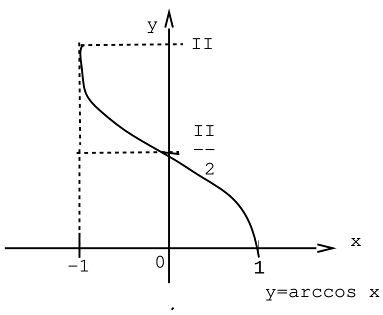

Figura 3.15: Representação gráfica da função arco co-seno

Exemplo: Seja $X = [0.5; 1.0]$. Então,

$$\begin{aligned} Arccos(X) &= \mathcal{I}(arccos; [0.5; 1.0]) \\ &= [arccos(0.5); arccos(1.0)] \\ &= [0.0; 1.0471975511965979]. \end{aligned}$$

Seja $X = [-0.5; 1.5]$. Então, $Arccos([-0.5; 1.5])$ é indefinido, pois neste intervalo a função co-seno não tem inversa.

3.4.3 Função arco tangente

Definimos a função arco tangente intervalar

$$\begin{aligned} Arctan : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \qquad \text{por} \\ X &\mapsto Arctan([x_1; x_2]) \end{aligned}$$

onde,

$$\begin{aligned} Arctan([x_1; x_2]) &= \mathcal{I}(arctan; [x_1; x_2]) \\ &= \{y \mid y = arctan(x) \text{ para algum } x \in [x_1; x_2]\} \\ &= \{y \mid x = tan(y) \text{ para algum } x \in [x_1; x_2] \text{ e } -\tfrac{-\pi}{2} \leqslant y \leqslant \tfrac{\pi}{2}\} \\ &= [arctan(x_1); arctan(x_2)]. \end{aligned}$$

O domínio de $arctan$ é $\mathbb{R}$ e sua imagem é $(-\pi/2; \pi/2)$.

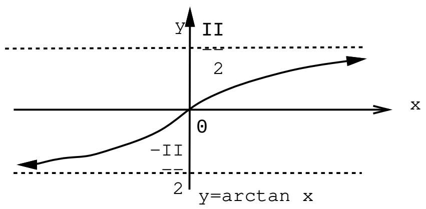

Figura 3.16: Representação gráfica da função arco tangente

Exemplo: Seja $X = [-1.0; 5.0]$. Então,

$$\begin{aligned} Arctan(X) &= \mathcal{I}(arctan; [-1.0; 5.0]) \\ &= [arctan(-1.0); arctan(5.0)] \\ &= [-0.7853981633974483; 1.373400766945016]. \end{aligned}$$

3.4.4 Função arco co-tangente

Definimos a função arco co-tangente intervalar

$$\begin{array}{lll} Arccot : \mathbb{I}(\mathbb{R}) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Arccot(X) & \end{array}$$

onde,

$$\begin{aligned} Arccot([x_1; x_2]) &= \mathcal{I}(arccot; [x_1; x_2]) \\ &= \{y \mid y = arccot(y) \text{ para algum } x \in [x_1; x_2]\} \\ &= \{y \mid x = cot(y) \text{ para algum } x \in [x_1; x_2] \text{ e } 0 < y < \pi\} \\ &= [arccot(x_2); arccot(x_1)]. \end{aligned}$$

O domínio de $arccot$ é $\mathbb{R}$ e sua imagem é $(0, \pi)$.

Exemplo: Seja $X = [0; 1]$. Então,

$$\begin{aligned} Arccot(X) &= \mathcal{I}(arccot; [0; 1]) \\ &= [arccot(1); arccot(0)] \\ &= [0.7853981633974483; 1.5707963267948966]. \end{aligned}$$

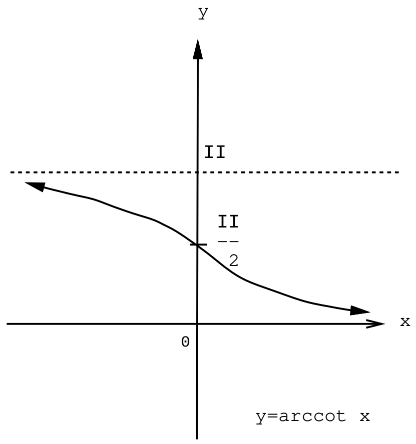

Figura 3.17: Representação gráfica da função arco co-tangente

3.4.5 Função arco secante

Definimos a função arco secante intervalar

$$\begin{array}{lll} Arcsec : \mathbb{I}(\mathbb{R}) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Arcsec(X) & \end{array}$$

onde,

$$\begin{aligned} Arcsec([x_1; x_2]) &= \mathcal{I}(arcsec; [x_1; x_2]) \\ &= \{y \mid y = arcsec(x) \text{ para algum } x \in [x_1; x_2]\} \\ &= \{y \mid x = sec(y) \text{ para algum } x \in [x_1; x_2] \text{ e } 0 \leq y \leq \pi \text{ e } y \neq \tfrac{\pi}{2} \\ &= \begin{cases} [arcsec(x_1); arcsec(x_2)], & \text{se } x_2 \leq -1 \text{ ou } 1 \leq x_1 \\ \uparrow & , \text{ senão.} \end{cases} \end{aligned}$$

O domínio de $arcsec$ é $(-\infty;-1]\cup[1;\infty)$ e sua imagem é $[0;\frac{\pi}{2})\cup(\frac{\pi}{2};\pi]$.

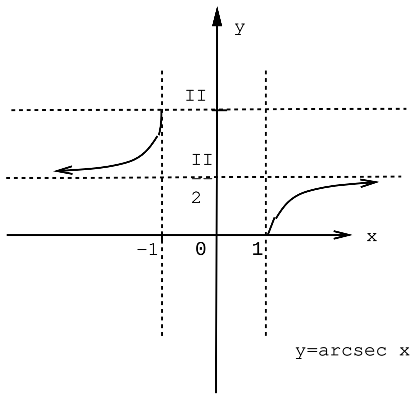

Figura 3.18: Representação gráfica da função arco secante

Exemplo: Seja $X=[1;2]$. Então,

$$\begin{aligned} Arcsec(X) &= \mathcal{I}(arcsec[1;2]) \\ &= [arcsec(1);arcsec(2)] \\ &= [0;\tfrac{\pi}{3}]. \end{aligned}$$

3.4.6 Função arco co-secante

Definimos a função arco co-secante intervalar

$$\begin{aligned} Arccsc : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \qquad \text{por} \\ X &\mapsto Arccsc(X) \end{aligned}$$

onde,

$$\begin{aligned} Arcsec([x_1;x_2]) &= \mathcal{I}(arcscs;[x_1;x_2]) \\ &= \{y \mid y = arccsc\ x \text{ para algum } x\in[x_1;x_2]\} \\ &= \{y \mid x = csc(y) \text{ para algum } x\in[x_1;x_2], y\neq 0 \text{ e } -\tfrac{\pi}{2}\leq y\leq\tfrac{\pi}{2}\} \\ &= \begin{cases} [arccsc(x_2);arccsc(x_1)], & \text{se } 1\leq x_1 \text{ ou } x_2\leq -1 \\ \uparrow \qquad\qquad\qquad , & \text{senão} \end{cases} \end{aligned}$$

O domínio de $arccsc$ é $(-\infty;-1]\cup[1;\infty)$ e sua imagem é $[-\frac{\pi}{2};0)\cup(0;\pi/2]$.

Exemplo: Seja $X=[\frac{-2\sqrt{3}}{3};-1]$. Então,

$$\begin{aligned} Arccsc(X) &= \mathcal{I}(arccsc;[-\tfrac{2\sqrt{3}}{3};-1]) \\ &= [arccsc(-1);arccsc(-\tfrac{2\sqrt{3}}{3})] \\ &= [-\tfrac{\pi}{2};-\tfrac{\pi}{3}]. \end{aligned}$$

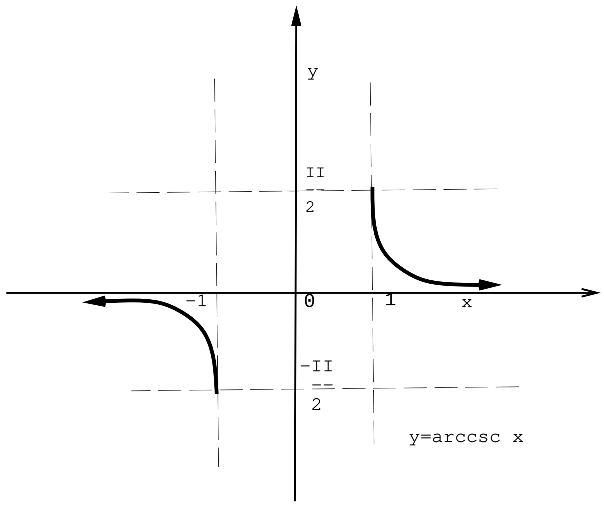

Figura 3.19: Representação gráfica da função arco co-secante

3.5 Funções hiperbólicas intervalares

Certas combinações de funções exponenciais, que estão relacionadas com uma hipérbole, aproximadamente da mesma maneira com que as funções trigonométricas estão relacionadas com o círculo, provaram ser importantes em matemática aplicada. Essas funções são chamadas **funções hiperbólicas** [MuF82], e por suas semelhanças com as funções trigonométricas, são enfatizadas chamando-as de **seno hiperbólico, co-seno hiperbólico, tangente hiperbólica, cotangente hiperbólica, secante hiperbólica, co-secante hiperbólica** e que descreveremos adiante.

3.5.1 Função seno hiperbólico

Definimos a função seno hiperbólico intervalar

$$\begin{array}{lll} Senh : \mathbb{I}(\mathbb{R}) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Senh(X) & \end{array}$$

onde,

$$\begin{aligned} Senh([x_1, x_2]) &= \mathcal{I}(senh; [x_1; x_2])\} \\ &= \{y \mid y = senh(x) \text{ para algum } x \in [x_1; x_2]\} \\ &= \left\{ y \mid y = \frac{e^x - e^{-x}}{2} \text{ para algum } x \in [x_1, x_2] \right\} \\ &= \left[\frac{e^{x_1} - e^{-x_1}}{2}; \frac{e^{x_2} - e^{-x_2}}{2} \right]. \end{aligned}$$

Exemplo: Seja $X = [-\frac{\pi}{2}; \frac{\pi}{2}]$. Então,

$$\begin{aligned} Senh(X) &= \mathcal{I}(senh; [-\tfrac{\pi}{2}; \tfrac{\pi}{2}]) \\ &= \left[\frac{e^{-\frac{\pi}{2}} - e^{-(-\frac{\pi}{2})}}{2}; \frac{e^{\frac{\pi}{2}} - e^{-\frac{\pi}{2}}}{2}\right] \\ &= [-2.3012989023072943; 2.3012989023072943]. \end{aligned}$$

3.5.2 Função co-seno hiperbólico

Definimos a função co-seno hiperbólico intervalar

$$\begin{aligned} Cosh : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \qquad \text{por} \\ X &\mapsto Cosh(X) \end{aligned}$$

onde,

$$\begin{aligned} Cosh([x_1; x_2]) &= \mathcal{I}(cosh; [x_1; x_2]) \\ &= \{y \mid y = cosh(x) \text{ para algum } x \in [x_2; x_1]\} \\ &= \left\{y \mid y = \frac{e^x + e^{-x}}{2} \text{ para algum } x \in [x_1; x_2]\right\} \\ &= \begin{cases} \left[1; \dfrac{e^{x_1} + e^{-x_1}}{2}\right], & \text{se } 0 \in X \text{ e } |x_1| \geqslant |x_2| \\ \left[1; \dfrac{e^{x_2} + e^{-x_2}}{2}\right], & \text{se } 0 \in X \text{ e } |x_1| < |x_2| \\ [minM; maxM], & \text{senão.} \end{cases} \end{aligned}$$

e M $= \{\frac{e^{x_1}+e^{-x_1}}{2}, \frac{e^{x_2}+e^{-x_2}}{2}\}$.

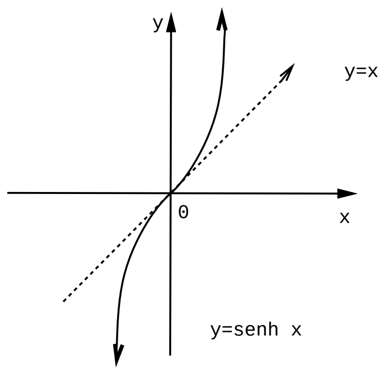

Figura 3.20: Representação gráfica da função seno hiperbólico

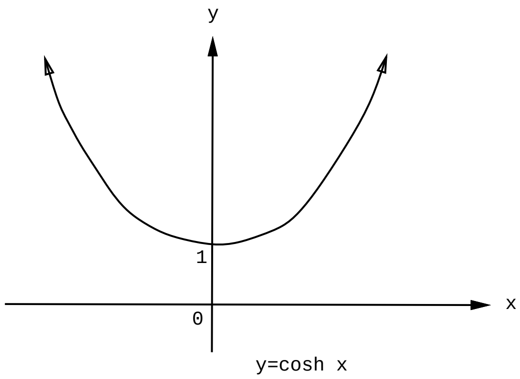

Figura 3.21: Representação gráfica da função co-seno hiperbólico

Exemplos: Seja $X = [-1; \frac{\pi}{2}]$. Então,

$$\begin{aligned} Cosh(X) &= \mathcal{I}(cosh; [-1; \tfrac{\pi}{2}]) \\ &= \left[1; \frac{e^{\frac{\pi}{2}} + e^{\frac{\pi}{2}}}{2}\right] \\ &= [1; 2.509178478658056]. \end{aligned}$$

Seja $X = [-3; -2]$. Então,

$$\begin{aligned} Cosh(X) &= \mathcal{I}(cosh; [-3; -2]) \\ &= \left[min\left\{\frac{e^{-3} + e^{-(-3)}}{2}, \frac{e^{-2} + e^{-(-2)}}{2}\right\}; max\left\{\frac{e^{-3} + e^{-(-3)}}{2}, \frac{e^{-2} + e^{-(-2)}}{2}\right\}\right] \\ &= [3.762195691083631; 10.067661995777764]. \end{aligned}$$

Seja $X = [1; 2]$. Então,

$$\begin{aligned} Cosh(X) &= \mathcal{I}(cosh; [1; 2]) \\ &= \left[min\left\{\frac{e^{1} + e^{1}}{2}, \frac{e^{2} + e^{-2}}{2}\right\}; max\left\{\frac{e^{1} + e^{-1}}{2}, \frac{e^{2} + e^{-2}}{2}\right\}\right] \\ &= [1.5430806348152437; 3.762195691083631]. \end{aligned}$$

3.5.3 Função tangente hiperbólica

Definimos a função tangente hiperbólica intervalar

$$\begin{array}{lll} Tanh : \mathbb{I}(\mathbb{R}) & \rightarrow \mathbb{I}(\mathbb{R}) & \text{por} \\ X & \mapsto Tanh(X) & \end{array}$$

onde,

$$\begin{aligned} Tanh([x_1;x_2]) &= \mathcal{I}(tanh;[x_1;x_2]) \\ &= \left\{ y \mid y = \frac{senh(x)}{cosh(x)} \text{ para algum } x \in [x_1;x_2] \right\} \\ &= \left\{ y \mid y = \frac{e^x - e^{-x}}{e^x + e^{-x}} \text{ para algum } x \in [x_1;x_2] \right\} \\ &= \left[\frac{e^{x_1} - e^{-x_1}}{e^{x_1} + e^{-x_1}}; \frac{e^{x_2} - e^{-x_2}}{e^{x_2} + e^{-x_2}} \right]. \end{aligned}$$

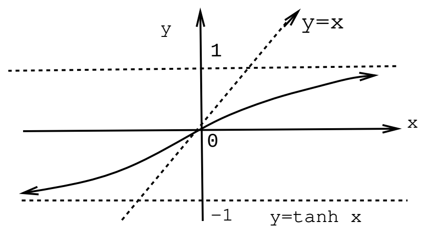

Figura 3.22: Representação gráfica da função tangente hiperbólica

Exemplo: Seja $X = [-1;1]$. Então,

$$\begin{aligned} Tanh(X) &= \mathcal{I}(tanh;[-1;1]) \\ &= \left[\frac{e^{-1} - e^{-(-1)}}{e^{-1} + e^{-(-1)}}; \frac{e^{1} - e^{-1}}{e^{1} + e^{-1}} \right] \\ &= [-0.7615941559557649; 0.7615941559557649]. \end{aligned}$$

3.5.4 Função co-tangente hiperbólica

Definimos a função co-tangente hiperbólica intervalar

$$\begin{array}{lll} Coth : \mathbb{I}(\mathbb{R}) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Coth(X) & \end{array}$$

onde,

$$\begin{aligned} Coth([x_1;x_2]) &= \mathcal{I}(coth;[x_1;x_2]) \\ &= \left\{ y \mid y = \frac{cosh(x)}{senh(x)} \text{ para algum } x \in [x_1;x_2] \right\} \\ &= \left\{ y \mid y = \frac{e^x + e^{-x}}{e^x - e^{-x}} \text{ para algum } x \in [x_1;x_2] \right\} \\ &= \begin{cases} \left[\dfrac{e^{x_2} + e^{-x_2}}{e^{x_2} + e^{-x_2}}; \dfrac{e^{x_1} - e^{-x_1}}{e^{x_1} - e^{-x_1}} \right], & \text{se } 0 \in [x_1;x_2] \\ \uparrow & , \text{ se } 0 \notin [x_1;x_2]. \end{cases} \end{aligned}$$

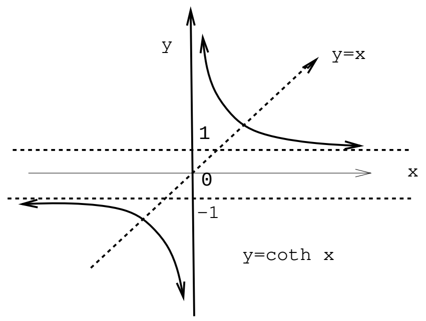

Figura 3.23: Representação gráfica da função co-tangente hiperbólica

Exemplos: Seja $X = [-1;\pi]$. Então, $Coth([-1;\pi])$ diverge, pois o intervalo não pode conter o valor zero.

Seja $X = [1;\pi]$. Então,

$$\begin{aligned} Coth(X) &= \mathcal{I}(coth;[1;\pi]) \\ &= \left[\frac{e^{-1} + e^{-(-1)}}{e^{-1} + e^{-(-1)}}; \frac{e^{\pi} - e^{-\pi}}{e^{\pi} - e^{-\pi}} \right] \\ &= [1.003741873197321; 1.3130352854993312]. \end{aligned}$$

Seja $X = [-2;-1]$. Então,

$$\begin{aligned} Coth(X) &= \mathcal{I}(coth;[-2;-1]) \\ &= \left[\frac{e^{-1} + e^{-(-1)}}{e^{-1} + e^{-(-1)}}; \frac{e^{-2} - e^{-(-2)}}{e^{-2} - e^{-(-2)}} \right] \\ &= [-1.3130352854993312; -1.037314720727548]. \end{aligned}$$

3.5.5 Função secante hiperbólica

Definimos a função secante hiperbólica intervalar

$$\begin{array}{lll} Sech : \mathbb{I}(\mathbb{R}) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto Sech(X) & \end{array}$$

onde,

$$\begin{aligned} Sech([x_1;x_2]) &= \mathcal{I}(sech;[x_1;x_2]) \\ &= \left\{ y \mid y = \frac{1}{cosh(x)} \text{ para algum } x \in [x_1;x_2] \right\} \\ &= \left\{ y \mid y = \frac{2}{e^x + e^{-x}} \text{ para algum } x \in [x_1;x_2] \right\} \\ &= \begin{cases} \left[\frac{2}{e^{x_1}+e^{-x_1}}; \frac{2}{e^{x_2}+e^{-x_2}}\right], & \text{se } 0 \notin [x_1;x_2] \\ \left[\frac{2}{e^{x_1}+e^{-x_1}}; 1\right] & , \text{ se } 0 \in [x_1;x_2], \text{ e } |x_1| \geqslant x_2 \\ \left[\frac{2}{e^{x_2}+e^{-x_2}}; 1\right] & , \text{ se } 0 \in [x_1;x_2], \text{ e } |x_1| < x_2 \end{cases} \end{aligned}$$

Exemplos: Seja $X = [-1; \frac{\pi}{2}]$. Então,

$$\begin{aligned} Sech(X) &= \mathcal{I}(sech; [-1; \tfrac{\pi}{2}]) \\ &= \left[\frac{2}{e^{\frac{\pi}{2}}+e^{-\frac{\pi}{2}}}; 1\right] \text{ pois } 0 \in [-1; \tfrac{\pi}{2}] \text{ e } |x_1| < x_2 \\ &= [0.3985368153383838675; 1]. \end{aligned}$$

Seja $X = [-2; -1]$. Então,

$$\begin{aligned} Sech(X) &= \mathcal{I}(sech; [-2; -1]) \\ &= \left[\frac{2}{e^{-2}+e^{2}}; \frac{2}{e^{-1}+e^{1}}\right] \text{ pois } 0 \notin [-2; -1] \\ &= [0.2658022288340797; 0.6480542736638855]. \end{aligned}$$

Seja $X = [1; 2]$. Então,

$$\begin{aligned} Sech(X) &= \mathcal{I}(sech; [1; 2]) \\ &= \left[\frac{2}{e^{1}+e^{-1}}; \frac{2}{e^{2}+e^{-2}}\right] \text{ pois } 0 \notin [1; 2] \\ &= [0.2658022288340797; 0.6480542736638855]. \end{aligned}$$

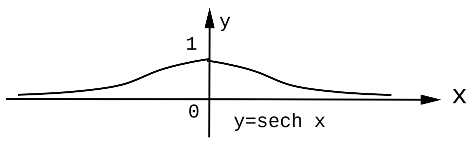

Figura 3.24: Representação gráfica da função secante hiperbólica

3.5.6 Função co-secante hiperbólica

Definimos a função co-secante hiperbólica intervalar

$$\begin{aligned} Csch : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \quad \text{por} \\ X &\mapsto Csch(X) \end{aligned}$$

onde,

$$\begin{aligned} Csch([x_1;x_2]) &= \mathcal{I}(cosch;[x_1;x_2]) \\ &= \{y \mid y = csch(x) \text{ para algum } x \in [x_1;x_2]\} \\ &= \left\{y \mid y = \frac{1}{senh(x)} \text{ para algum } x \in [x_1;x_2]\right\} \\ &= \left\{y \mid y = \frac{2}{e^x + e^{-x}} \text{ para algum } x \in [x_1;x_2]\right\} \\ &= \begin{cases} \left[\dfrac{2}{e^{x_2}+e^{-x_2}}; \dfrac{2}{e^{x_1}+e^{-x_1}}\right], & \text{se } 0 \notin [x_1;x_2] \\ \uparrow & , \text{ se } 0 \in [x_1;x_2]. \end{cases} \end{aligned}$$

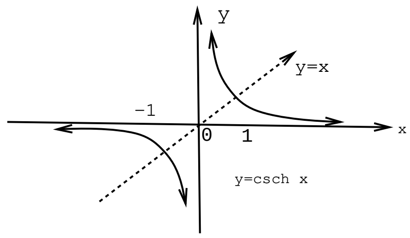

Figura 3.25: Representação gráfica da função co-secante hiperbólica

Exemplos: Seja $X = [1;3]$. Então,

$$\begin{aligned} Csch(X) &= \mathcal{I}(csch;[1;3]) \\ &= \left[\frac{2}{e^3 - e^{-3}}; \frac{2}{e^1 - e^{-1}}\right] \text{ pois } 0 \notin [1;3] \\ &= [0.09982156966882275; 0.8509181282393216]. \end{aligned}$$

Seja $X = [-1;1]$. Então, $Csch(X)$ diverge, pois o intervalo não pode conter o valor zero.

3.6 Funções hiperbólicas inversas intervalares

Por analogia com as seis funções trigonométricas inversas, existem seis funções hiperbólicas inversas, que consideraremos nesta seção. Ao contrário das análogas trigonométricas, as funções $senh$, $tanh$, $coth$ e $csch$ são monótonas, logo não há problemas quanto à existência de $senhI$, $tanhI$, $cothI$, e $cschI$. É sabido que os gráficos destas funções são obtidos pela reflexão dos gráficos de $senh$, $tanh$, $coth$, e $csch$, respectivamente, em relação à reta $y = x$.

3.6.1 Função seno hiperbólico inverso

Definimos a função seno hiperbólico inverso intervalar

$$\begin{array}{ll} SenhI : \mathbb{I}(\mathbb{R}) & \rightarrow \mathbb{I}(\mathbb{R}) \quad \text{por} \\ \quad X & \mapsto SenhI(X) \end{array}$$

onde,

$$\begin{aligned} SenhI([x_1; x_2]) &= \mathcal{I}(senhI; [x_1; x_2]) \\ &= \{y \mid y = ln(x + \sqrt{x^2+1}) \text{ para algum } x \in [x_1; x_2]\} \\ &= \left[ln\left(x_1 + \sqrt{x_1^2+1}\right); ln\left(x_2 + \sqrt{x_2^2+1}\right)\right]. \end{aligned}$$

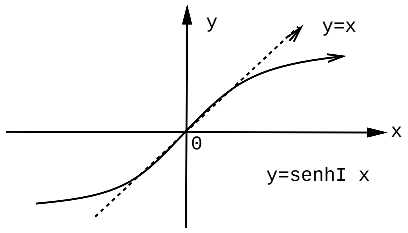

Figura 3.26: Representação gráfica da função seno hiperbólico inverso

Exemplo: Seja $X = [-\frac{\pi}{2}; \frac{\pi}{2}]$. Então,

$$\begin{aligned} SenhI(X) &= \mathcal{I}(senhI; [-\tfrac{\pi}{2}; \tfrac{\pi}{2}]) \\ &= [ln(x + \sqrt{x^2+1}); ln(x + \sqrt{x^2+1})] \\ &= \left[ln\left(-\frac{\pi}{2} + \sqrt{-\frac{\pi}{2}^2 + 1}\right); ln\left(\frac{\pi}{2} + \sqrt{\frac{\pi}{2}^2 + 1}\right)\right] \\ &= [-1.233403117511217; 1.233403117511217]. \end{aligned}$$

3.6.2 Função co-seno hiperbólico inverso

Definimos a função co-seno hiperbólico inverso intervalar

$$\begin{array}{ll} CoshI : \mathbb{I}([1;\infty]) & \to \mathbb{I}(\mathbb{R}) \qquad \text{por} \\ \quad X & \mapsto CoshI(X) \end{array}$$

onde, $\mathbb{I}([1;\infty]) = \{[x_1;x_2] \in \mathbb{I}(\mathbb{R}) \mid x_1 \geqslant 1\}$.

$$\begin{aligned} CoshI([x_1;x_2]) &= \mathcal{I}(coshI;[x_1;x_2]) \\ &= \left\{y \mid y = ln\left(x+\sqrt{x^2-1}\right) \text{ para } x \geqslant 1 \text{ e } x \in [x_1;x_2]\right\} \\ &= \left[ln\left(x_1+\sqrt{x_1^2-1}\right); ln\left(x_2+\sqrt{x_2^2-1}\right)\right]. \end{aligned}$$

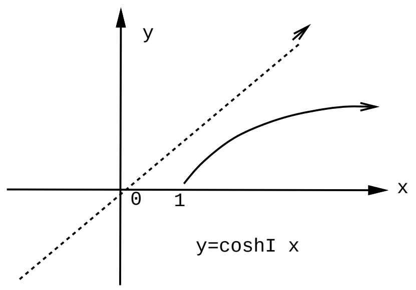

Figura 3.27: Representação gráfica da função co-seno hiperbólico inverso

Exemplos: Seja $X = [2;3]$. Então,

$$\begin{aligned} CoshI(X) &= \mathcal{I}(coshI;[2;3]) \\ &= \left[ln\left(2+\sqrt{2^2-1}\right); ln\left(3+\sqrt{3^2-1}\right)\right] \\ &= [1.3169578969248166; 1.7627471740390859]. \end{aligned}$$

Seja $X = [-\frac{\pi}{2}; \frac{\pi}{2}]$. Então, $CoshI([-\frac{\pi}{2}; \frac{\pi}{2}])$ diverge pois o intervalo não pode conter um número menor que 1.

3.6.3 Função tangente hiperbólica inversa

Definimos a função tangente hiperbólica inversa intervalar

$$\begin{array}{lll} TanhI : \mathbb{I}([-1;1]) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto TanhI(X) & \end{array}$$

onde,

$$\begin{aligned} TanhI([x_1, x_2]) &= \mathcal{I}(tanhI; [x_1, x_2]) \\ &= \{y \mid y = tanhI(x) \text{ para algum } x \in [x_1, x_2]\} \\ &= \left\{y \mid y = \frac{1}{2} ln\left(\frac{1+x}{1-x}\right) \text{ para } |x| < 1 \text{ e } x \in [x_1, x_2]\right\} \\ &= \left[\frac{1}{2} ln\left(\frac{1+x_1}{1-x_1}\right); \frac{1}{2} ln\left(\frac{1+x_2}{1-x_2}\right)\right]. \end{aligned}$$

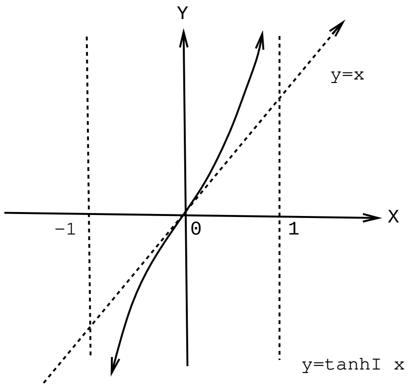

Figura 3.28: Representação gráfica da função tangente hiperbólica inversa

Exemplo: Seja $X = [-\frac{1}{2}; \frac{1}{2}]$. Então,

$$\begin{aligned} TanhI(X) &= \mathcal{I}(tanhI; [-\tfrac{1}{2}; \tfrac{1}{2}]) \\ &= \left[\frac{1}{2} ln\left(\frac{1+(-\frac{1}{2})}{1-(-\frac{1}{2})}\right); \frac{1}{2} ln\left(\frac{1+(\frac{1}{2})}{1-(\frac{1}{2})}\right)\right] \\ &= [-0.5493061443340549; 0.5493061443340549]. \end{aligned}$$

3.6.4 Função co-tangente hiperbólica inversa

Definimos a função co-tangente hiperbólica inversa intervalar

$$\begin{array}{lll} CothI: \mathbb{I}(\mathbb{R}) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto CothI(X) & \end{array}$$

onde,

$$\begin{aligned} CothI([x_1;x_2]) &= \mathcal{I}(cothI;[x_1;x_2]) \\ &= \{y \mid y = cothI(x) \text{ para algum } x \in [x_1;x_2]\} \\ &= \left\{y \mid y = \frac{1}{2}ln\left(\frac{x+1}{x-1}\right) \text{ para } |x| > 1 \text{ e } x \in [x_1;x_2]\right\} \\ &= \begin{cases} \left[\frac{1}{2}ln\left(\frac{x_2+1}{x_2-1}\right); \frac{1}{2}ln\left(\frac{x_1+1}{x_1-1}\right)\right], & \text{se } x_1 > 1 \text{ ou } x_2 < -1 \\ \uparrow & , \text{ senão.} \end{cases} \end{aligned}$$

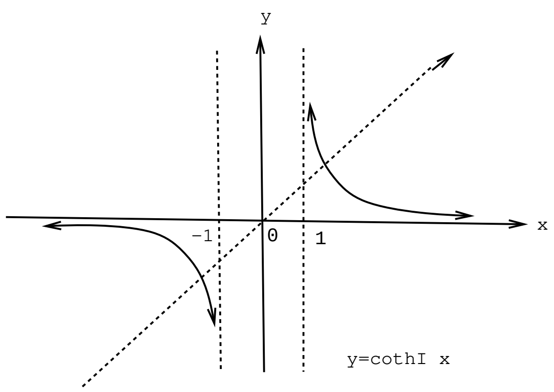

Figura 3.29: Representação gráfica da função co-tangente hiperbólica inversa

Exemplos: Seja $X = [-3;-2]$. Então,

$$\begin{aligned} CothI(X) &= \mathcal{I}(cothI;[-3;-2]) \\ &= \left[\frac{1}{2}ln\left(\frac{-2+1}{-2-1}\right); \frac{1}{2}ln\left(\frac{-3+1}{-3-1}\right)\right] \\ &= [-0.5493061443340549; -0.34657359027997264]. \end{aligned}$$

Seja $X = [-1;1]$. Então, $CothI([-1;1])$ diverge, pois x_1 deve ser maior ou igual a 1 ou x_2 deve ser menor ou igual a -1.

3.6.5 Função secante hiperbólica inversa

Definimos a função secante hiperbólica inversa intervalar

$$\begin{array}{lll} SechI : \mathbb{I}((0;1]) & \to \mathbb{I}(\mathbb{R}) & \text{por} \\ \quad X & \mapsto SechI(X) & \end{array}$$

onde $\mathbb{I}((0;1]) = \{[x_1;x_2] \in \mathbb{I}(\mathbb{R}) \mid x_1 > 0 \text{ e } x_2 \leq 1\}$

$$\begin{aligned} SechI([x_1;x_2]) &= \mathcal{I}(sechI;[x_1;x_2]) \\ &= \{y \mid y = sechI(x) \text{ para algum } x \in [x_1;x_2]\} \\ &= \left\{ y \mid y = ln\left(\frac{1+\sqrt{1-x^2}}{[x_1;x_2]}\right) \text{ para } x \in [x_1;x_2] \right\} \\ &= \left[ln\left(\frac{1+\sqrt{1-x_2^2}}{x_2}\right); ln\left(\frac{1+\sqrt{1-x_1^2}}{x_1}\right)\right]. \end{aligned}$$

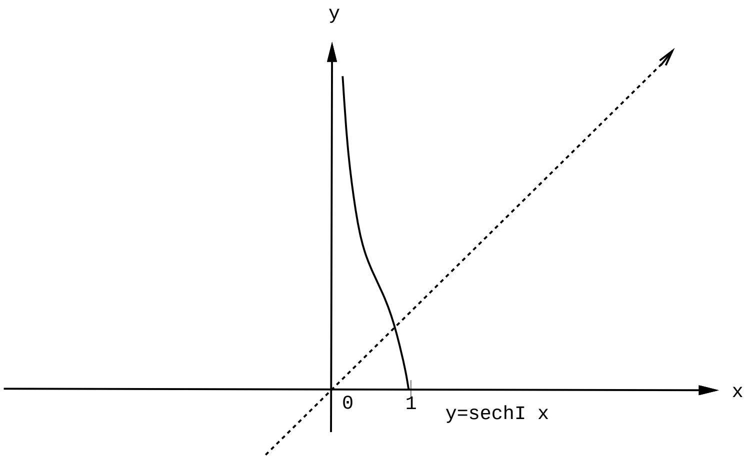

Figura 3.30: Representação gráfica da função secante hiperbólica inversa

Exemplos: Seja $X = [\frac{1}{2};1]$. Então,

$$\begin{aligned} SechI([x_1;x_2]) &= \mathcal{I}(sechI;[\tfrac{1}{2};1]) \\ &= \left[ln\left(\frac{1+\sqrt{1-1^2}}{1}\right); ln\left(\frac{1+\sqrt{1-\frac{1}{2}^2}}{\frac{1}{2}}\right)\right] \\ &= [0; 1.3169578969248166]. \end{aligned}$$

Seja $X = [-2;-1]$. Então, $SechI([-2;-1])$ diverge, pois o extremo direito do intervalo não pode ser menor ou igual a 0.

Seja $X = [10; 20]$. Então, $SechI([10; 20])$ diverge, pois o extremo esquerdo do intervalo não pode ser maior que 1.

Seja $X = [0.3; 0.5]$. Então,

$$\begin{aligned} SechI(X) &= \mathcal{I}(sechI; [0.3; 0.5]) \\ &= \left[ln\left(\frac{1+\sqrt{1-0.5^2}}{0.5}\right); ln\left(\frac{1+\sqrt{1-0.3^2}}{0.3}\right)\right] \\ &= [1.3169578969248166; 1.8738202425274144]. \end{aligned}$$

3.6.6 Função co-secante hiperbólica inversa

Definimos a função co-secante hiperbólica inversa intervalar

$$\begin{aligned} CschI : \mathbb{I}(\mathbb{R}) &\to \mathbb{I}(\mathbb{R}) \quad \text{por} \\ X &\mapsto CschI(X) \end{aligned}$$

onde,

$$\begin{aligned} CschI([x_1, x_2]) &= \mathcal{I}(cschI; [x_1; x_2]) \\ &= \{y \mid y = cschI(x) \text{ para algum } x \in X\} \\ &= \left\{ y \mid y = ln\left(\frac{1}{x} + \frac{\sqrt{1+x^2}}{|x|}\right) \text{ para } x \neq 1 \text{ e } x \in [x_1; x_2] \right\} \\ &= \begin{cases} \left[ln\left(\frac{1}{x_2} + \frac{\sqrt{1+x_2^2}}{|x_2|}\right); ln\left(\frac{1}{x_1} + \frac{\sqrt{1+x_1^2}}{|x_1|}\right)\right], & \text{se } 0 \notin [x_1; x_2] \\ \uparrow, & \text{senão.} \end{cases} \end{aligned}$$

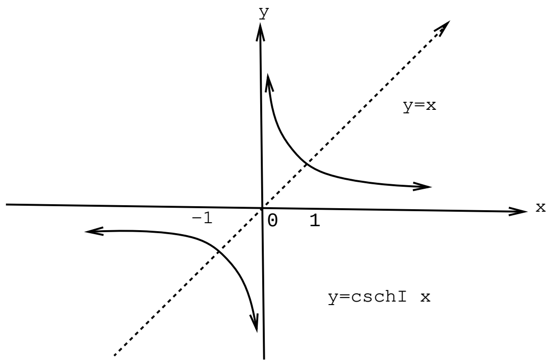

Figura 3.31: Representação gráfica da função co-secante hiperbólica inversa

Exemplos: Seja $X = [1; \frac{\pi}{2}]$. Então,

$$\begin{aligned} CschI(X) &= \mathcal{I}(cschI; [1; \pi]) \\ &= \left[ln\left(\frac{1}{\pi} + \frac{\sqrt{1+\pi^2}}{|\pi|}\right); ln\left(\frac{1}{1} + \frac{\sqrt{1+1^2}}{|1|}\right)) \right] \\ &= [1.3169578969248166; 1.8738202425274144]. \end{aligned}$$

Seja $X = [1; \frac{\pi}{2}]$. Então, $CschI(X)$ diverge, pois o intervalo não pode conter zero.

Capítulo 4

Linguagens XSC

4.1 Introdução

Anteriormente, a ênfase do processo computacional estava na velocidade, mas no presente podemos dar mais ênfase à precisão e a confiança nos resultados. A aritmética atualmente usada (ponto-flutuante ordinária) geralmente retorna um resultado que difere do resultado exato, tendo em vista os processos de arredondamento realizados, sendo o que acontece com a maioria das modernas linguagens de programação. Aos programadores, estas linguagens possibilitam escrever expressões simples, fórmulas ou funções da mesma forma que na notação usual da matemática.

Desde a década de sessenta são desenvolvidas pesquisas com o objetivo de possibilitar que os computadores suportem uma aritmética mais poderosa que a aritmética de ponto-fluante ordinária. Nota-se a importância da Matemática Intervalar que desenvolve algoritmos altamente precisos, e que automaticamente verificam seus resultados aplicando teoremas matemáticos. Isto significa que estas computações carregam seu próprio controle de precisão. Porém, estas implementações requerem suporte aritmético que seja satisfatório, e ferramentas de programação poderosas que não estão geralmente disponíveis. Esta exigência fez com que fosse necessário estender linguagens para computações científicas, sendo estas extensões conhecidas como XSC.

Entre 1976 e 1979, dois Institutos das Universidades de Karlsurhe e Kaiserlautern cooperaram para desenvolver e implementar uma extensão do PASCAL, conhecida como PASCAL-SC. Nos anos seguintes, em cooperação com a IBM, uma correspondente versão do FORTRAN 77 foi implementada em computadores IBM/370, chamada de ACRITH-SC. Descreveremos mais adiante sobre a evolução destas linguagens.

4.2 Características

As atuais extensões de linguagem para computação científica provêem as características indispensáveis para o desenvolvimento de moderno software numérico, tais como:

- Conceito de operador (os operadores são definidos pelo usuário).
- Conceito de overloading.
- Conceito de módulo.
- Arrays dinâmicos.
- Controle de arredondamento.
- Tipos de dados aritméticos predefinidos reais (extensão do real), complex (complexo), interval (intervalos), complex intervals (intervalo complexo), e correspondentes tipos de vetores e matrizes.
- Os operadores aritméticos predefinidos com enorme precisão para tipos de dados aritméticos.
- Funções elementares predefinidas com enorme precisão para os tipos de dados aritméticos.
- Tipos de dados com representação exata após a vírgula.
- Biblioteca com rotinas matemáticas para a resolução de problemas matemáticos com verificação automática de resultado e alta precisão.

4.3 Evolução

Soluções em diferentes hardwares estão disponíveis para computadores pessoais, estações de trabalho, mainframes e super computadores. Em particular, um coprocessador aritmético de vetor para PC foi desenvolvido usando a tecnologia VLSI. Suporte às linguagens XSC estão disponíveis em FORTRAN, PASCAL e C (ACRITH-XSC, PASCAL-XSC e C-XSC respectivamente). Rotinas que resolveram problemas com verificação automática de resultado foram desenvolvidas para muitos problemas usuais da análise numérica linear como para sistemas de equações não lineares, equações diferenciais, integrais, etc, além de rotinas para um grande número de aplicações na engenharia e nas ciências naturais.

Os pacotes (conjuntos de programas) baseados na matemática intervalar podem ser classificados da seguinte forma [Kea96]:

1. **Primeiros pacotes:** A maioria dos conjuntos de programas eram em Fortran ou baseados em ALGOL, e foram construídos com o princípio da pré-compilação.

2. **Linguagens SC:** Incluindo o FORTRAN-SC e PASCAL-SC, estas extensões de linguagens comuns caracterizam um produto de boa precisão, e um conjunto grande e útil de operadores e funções.

3. **Linguagens XSC:** Criadas no modelo "SC ", são linguagens portáteis fundadas principalmente no princípio do operador de overloading.

4. **Outros modernos pacotes:** Incluem o INTLIB_90 e vários pacotes em C++, e outras linguagens que suportam operadores de overloading.

5. **Pacotes para manipuladores simbólicos:** Contêm bibliotecas para matemática intervalar em Mathematica, Maple, etc.

Apresentaremos a seguir uma descrição mais detalhada destes pacotes e linguagens.

Pacotes iniciais

Um dos primeiros pacotes disponíveis com razoável portabilidade foi o conjunto de rotinas de aritmética intervalar Yohe, desenvolvido em FORTRAN-66 com o pré-compilador Augment [Cra76]. À frente de seu tempo, o Augment [Cra76] tem como características o operador de overloading e uma moderna construção. As rotinas de Yohe também foram feitas em linguagem de baixo nivel, como Assembler, e foram projetadas para serem utilizadas em várias máquinas como os antigos mainframes IBM 360 e máquinas Univac.

Jeter e Shriver desenvolveram o pré-processador portátil SEPAFOR (na linguagem FORTRAN IV) para uma aritmética intervalar de precisão variável [JeS84]. Tal precisão da variável é útil, por exemplo, para resolver sistemas de equações diferenciais, onde a precisão pode ser ajustada para considerar a sensibilidade das condições iniciais, etc.

O compilador M77, que processava em mainframes CDC, foi desenvolvido pela Universidade de Minnesota. Era um compilador FORTRAN-77 que tinha uma livraria de matemática intervalar extensa, desenvolvida por W. R. Walster e suportava eficientemente o tipo de dado intervalar dentro do compilador [UMC83].

As Linguagens SC

Constituíram a primeira geração das linguagens científicas, e ficaram conhecidas como SC, devido a sua capacidade limitida. As linguagens iniciais foram FORTRAN-SC e PASCAL-SC [Ral87] que tinham capacidades intervalares e tipos de dados definidos pelo usuário, e que eram extensamente acessíveis. Estas linguagens de programação primeiro foram disponíveis para microcomputadores baseados no sistema operacional CP/M, e depois para microcomputadores com o sistema operacional DOS. PASCAL-SC é citada amplamente na literatura sobre computações intervalares. Numerosos pacotes foram construídos em PASCAL-SC e FORTRAN-SC para tarefas numéricas usuais e para resolução de sistema lineares.

FORTRAN-77 e FORTRAN-SC são compiladores para mainframes IBM 370 que faziam uso do pacote de subrotinas da ACRITH-SC [Rum85, JaW86].

Ambos, ACRITH-SC e PASCAL-SC, caracterizam-se por uma boa precisão, por considerarem tipos de dados intervalares intrínsecos, por terem a capacidade de alocação dinâmica de vetores e tipos de dados definidos pelo usuário, com operador de overloading.

Uma das desvantagens das linguagens PASCAL-SC e ACRITH-SC é que elas estão disponíveis para uma variedade limitada de computadores.

4.4 As linguagens XSC

O advento das linguagens de programação modernas, como Ada, C++, FORTRAN90 e JAVA, facilitaram a portabilidade das linguagens, isto é, a capacidade que tem a linguagem de se adaptar a diversos equipamentos e softwares. Uma segunda geração de linguagens, as linguagens XSC, modeladas em PASCAL-XSC e ACHITH-XSC também conhecido como FORTRAN-XSC, têm portabilidade. Desenvolvidas através da liderança do Prof. Dr. Kulisch, da Universidade de Karlsruhe, estas linguagens têm como característica a maximização da precisão e operações intervalares.

PASCAL-XSC [HNR93] é essencialmente um compilador com portabilidade que adere às especificações do PASCAL-SC. O próprio compilador é escrito em C. Uma caixa de ferramentas numéricas, isto quer dizer, um pacote bem-documentado com rotinas para avaliação de polinômios, diferenciação automática, raízes únicas de equações, precisão, verificação de sistemas de equações lineares, sistemas de equações não-lineares, optimização global, etc. está disponível em PASCAL-XSC. Krämer [Kra93] escreveu um conjunto de módulos para a precisão de variáveis de ponto flutuante e matemática intervalar em PASCAL-XSC. O próprio compilador é comercial: há licenças para versões executáveis para máquinas específicas e compiladores de C++.

C-XSC [KKW+93] é uma extensão da linguagem de programação C que consiste em um conjunto de classes de bibliotecas em C++. A extensão C-XSC é uma das várias implementações de matemática intervalar baseadas em C++.

FORTRAN-XSC [Wal93], baseado no FORTRAN90 (a mais recente versão) ainda está em desenvolvimento. Porém, a ênfase no projeto do Fortran-XSC está em um produto de boa precisão e em módulos para aritmética de precisão, como também para intervalos de vetor e operações de matriz.

4.5 PASCAL-XSC

A linguagem PASCAL-XSC [HHD97] é uma extensão do PASCAL *Standard* sendo, pois, uma linguagem de programação de propósito geral, mas que também proporciona condições especiais à implementação de algoritmos numéricos sofisticados. Esse novo sistema tem a vantagem de ser portável a várias plataformas e contém um tipo de dado específico para intervalos, conhecido como *interval*.

4.5.1 Características básicas da linguagem PASCAL-XSC

A extensão científica PASCAL-XSC da linguagem PASCAL possui muitas vantagens se comparada ao PASCAL *Standart*, de fato possui todas as características do PASCAL acrescidas de outras, citadas a seguir:

- o conceito de operador universal (operador definido pelo usuário).
- funções e operadores com tipo de resultado arbitrário.
- sobrecarga de procedimentos.
- rotinas de leitura e escrita.
- conceito de módulo.
- *arrays* dinâmicos, acesso a *subarrays*.
- conceito de *string*.
- arredondamento controlado.
- produto escalar ótimo (exato).
- tipo padrão *dotprecision* e tipos aritméticos adicionais.
- aritmética de alta-exatidão para todos os tipos.

- funções de alta-exatidão e avaliação exata de expressões.
- funções e operadores.

O PASCAL-XSC possui, ainda, **aritmética intervalar**, aritmética complexa, aritmética intervalar complexa e as correspondentes aritméticas de vetores e matrizes. Encontram-se também no PASCAL-XSC módulos para a resolução de problemas numéricos, tais como: sistema de equações lineares, inversão de matrizes, sistemas de equações não-lineares, autovalores e autovetores, avaliação de expressões aritméticas e muitos outros.

4.5.2 Aplicações em PASCAL-XSC

A linguagem PASCAL-XSC pode ser aplicada na solução dos mais diversos problemas em que se exigem alta-exatidão e grande confiabilidade dos resultados obtidos. A seguir são citadas algumas aplicações do PASCAL-XSC em computações científicas e nas engenharias.

- **Cálculos de múltipla precisão com verificação do resultado.** Têm sido desenvolvidos para o PASCAL-XSC módulos de intervalos e reais de múltipla precisão.

- **Núcleo de** *Hardware* **para Engenharia e Computação Científica.** As atuais unidades de ponto-flutante ou processadores são caracterizadas por arquiteturas desenvolvidas há algumas décadas atrás, quando o *hardware* era um recurso precioso e a necessidade de uma aritmética de precisão não havia sido investigada exaustivamente. Resultados não dignos de confiança e algumas vezes impróprios eram as conseqüências. Investigações desses efeitos levaram a uma definição matemática de uma necessidade aritmética. Adicionalmente ao cercamento dos resultados, através de uma aritmética intervalar, até mesmo a verificação da existência e unicidade dos resultados computados é possível com a utilização do PASCAL-XSC.

- **Soluções confiáveis.** Estas têm sido computadas para um certo número de problemas científicos e de engenharia, com rotinas para a resolução de problemas com verificação automática de resultados, para problemas básicos de análise numérica, fornecidas na linguagem de programação PASCAL-XSC.

4.5.3 Módulo I_ARI - aritmética intervalar

Este módulo contém todos os operadores e procedimentos necessários aos cálculos com intervalos. Todos os operadores predefinidos retornam um resultado do tipo *interval*, que é um tipo definido na linguagem PASCAL-XSC para intervalo. Existem os operadores *monadic*$(+, -)$ e as quatro operações básicas +, -, *, /, cada qual com o arredondamento para o menor intervalo que contenha o intervalo resposta. O módulo I_ARI inclui também o operador **in** para a relação "está contido" entre um dado *real* e um *interval*, ou entre um *interval* e outro *interval*.

Os operadores +* e ** denotam a união de intervalos e a intersecção, respectivamente. O operador +* retorna o menor intervalo que ainda contenha os dois operandos. O operador ** retorna a intersecção. Há um erro quando a intersecção é vazia.

Exemplo: Se A e B são do tipo interval e são definidos como $A = [-1, 3]$ e $B = [-1; 4]$ então os operadores $+$, $-$, $*$, $+$, $*$ e $**$ retornam os seguintes resultados:

Expressão	Resultado
A+B	[2;7]
A-B	[-5;0]
A*B	[-4;12]
A+*B	[-1,4]
A**B	[3,3]

Funções de transferência

As seguintes funções de transferência são suportadas para a conversão de tipos entre *real* e *interval*.

PASCAL-XSC		
Funções de transferência		
Função	Tipo de Resultado	Significado
intval(r1,r2)	interval	Intervalo com inf=r1 e sup=r2
intval(r)	interval	Intervalo com inf=sup=r
inf(i)	real	Parte baixa de i
sup(i)	real	Parte alta de i
$r, r1, r2$ = **Expressão real, i=expressão interval**		

Tabela 4.1: Funções de transferência do PASCAL-XSC.

PASCAL-XSC		
Módulo I_ARI-Funções intervalares		
Função	Tipo do Resultado	Nome genérico
sqr(i)	interval	Quadratica
sqrt(i)	interval	Raiz quadrada
exp(i)	interval	Exponencial
exp2(i)	interval	Exponencial com base 2
exp10(i)	interval	Exponencial com base 10
ln(i)	interval	Logaritmo natural
log2(i)	interval	Logaritmo de base 2
log10(i)	interval	Logaritmo de base 10
sin(i)	interval	Seno
cos(i)	interval	Cosseno
tan(i)	interval	Tangente
cot(i)	interval	Co-tangente
arcsin(i)	interval	Arco seno
arccos(i)	interval	Arco co-seno
arctan(i)	interval	Arco tangente
arctan2(i1,i2)	interval	Arco tangente de i1/i2
arccot(i)	interval	Arco co-tangente
$i, i1, i2 =$ **expressões intervalares**		

Tabela 4.2: Módulo I_ARI-Funções Intervalares em PASCAL-XSC.

4.5.4 Funções predefinidas

Todas as funções matemáticas do PASCAL-XSC, disponíveis para argumentos reais, são também suportadas para argumentos intervalares. Além disso, funções para o cálculo do ponto médio e diâmetro de intervalos estão também disponíveis.

Exemplo: Sejam A e B do tipo intervalo e definidos como:

A:=intval(-1;3)
B:=intval(2);
então as funções abs, sqr, mid e $diam$ resultam:

Expressão	Resultado
abs(A)	$[0; 3]$
abs(B)	$[2; 2]$
sqr(A)	$[0; 9]$
sqr(B)	$[4; 4]$
mid(A)	1
diam(A)	4

PASCAL-XSC		
Módulo I_ARI-Funções intervalares		
Função	Tipo do Resultado	Nome genérico
sinh(i)	interval	Seno hiperbólico
cosh(i)	interval	Cosseno hiperbólico
tanh(i)	interval	Tangente hiperbólica
coth(i)	interval	Cotangente hiperbólica
arsinh(i)	interval	Seno hiperbólico inverso
arcosh(i)	interval	Cosseno hiperbólico inverso
artanh(i)	interval	Tangente hiperbólica inversa
arcoth(i)	interval	Co-tangente hiperbólica inversa
abs(i)	interval	Absoluto
mid(i)	real	Ponto médio
diam(i)	real	Diâmetro
blow(i)	interval	Inflação
$i, i1, i2 =$ **expressões intervalares**		

Tabela 4.3: Módulo I_ARI-Funções Intervalares em PASCAL-XSC (continuação).

Em PASCAL-XSC, expressões intervalares são escritas na notação intervalar. Nomes das funções genéricas são usados para funções intervalares, como raiz quadrada intervalar, seno e cosseno intervalares. O programa seguinte é o método de Newton em PASCAL-XSC.

```
program inewt (input, output)
use
  i-ari; (i-ari : módulo de aritmética intervalar)
var x,y : interval;

funcion f(r : real) : interval;
var x : interval;
begin
  x:=r;(converte r p/ o tipo intervalo p/ obter uma inclusão verificada)
  f:= sqrt(x)+(x+1)*cos(x)
end;

function deriv (x: interval) : interval;
begin
  deriv:= 1 / (2 * sqrt(x)) + cos(x) - (x+1) * sin(x)
end;
```

```
funcion criter (x: interval) : boolean;
begin
  criter:= (sup(f(inf(x)) * f(sup(x))) < 0) and not (0 in deriv(x));
end;


begin
write("Entre com o intervalo inicial:");
read(y);
while inf(y) ≠ sup(y) do
  begin
      if criter(y) then
       repeat
         x:=y;
         writeln (x);
         y:= ( mid(x) - f(mid(x))/deriv(x) ** x);
       until x=y
      else
            writeln ("Critério não satisfeito !");
      writeln;
      write ("Entre o intervalo inicial");
      read(y);
  end;
end.
```

4.6 INTLIB90

Esta extensão consiste em sub-rotinas e funções em FORTRAN 77 standard para operações elementares. O arquivo da INTLIB alcança portabilidade com arredondamento. A extensão INTLIB é diretamente acessível por FORTRAN 90.

4.6.1 Filosofia

Em INTLIB [Kea95] as rotinas são bem documentadas, geralmente avaliadas, com qualidade e totalmente portáteis. Assim, os usuários podem transferir programas de aplicação de uma máquina para outra e o programa, sem modificação, executará eficazmente em uma variedade de máquinas. Além disso, rotinas em INTLIB são significativamente legíveis aos usuários, e fornecidas suas instruções, permitindo o usuário entendê-las e melhorá-las. As rotinas são claramente precisas, sem grande número de funções supérfluas. E, quando do seu desenvolvimento, não foram incluídas rotinas onde havia ambigüidade de interpretação matemática.

INTLIB	
Rotinas de aritmética elementar para intervalos	
Nome	Função
ADD	Adição
IDIV	Divisão
MULT	Multiplicação
RNDOUT	Arredondamento
SCLADD	Adição de intervalos e números reais
SCLMLT	Multiplicação de escalar
SUB	Subtração

Tabela 4.4: Rotinas de aritmética elementar para intervalos em INTLIB.

4.6.2 Conteúdos da INTLIB

A INTLIB é organizado em subconjuntos: Rotinas aritméticas elementares [Tabela 4.4], Rotinas de funções padrão [Tabela 4.5], Rotinas de funções úteis [Tabela 4.6], Rotinas de impressões de erro, Rotinas de constantes matemáticas de programas que é utilizada para inicializar constantes globais e parâmetros. Veremos a seguir as três principais rotinas.

O módulo INTLIB-ARITHMETIC define as quatro operações elementares (+, -, *, e /) em tipos de dados intervalares, como também negação (ex: menos unário). As operações mistas são permitidas somente entre intervalos e números com dupla precisão (ou tipo equivalente), ou entre intervalos e números inteiros. Exponenciação * * está também definida para intervalo para inteiro, intervalo para intervalo, intervalo para números com dupla precisão e de intervalo para dupla precisão.

Tipos de dados intervalares

Este módulo define nomes genéricos para as funções elementares. As seguintes funções aceitam argumentos de intervalos:

MAG, IWID, IMIG, ABS, ACOS, ACOT, ASIN, ATAN, COS, COT, EXP, LOG, SINH, SIN, SQRT, TAN, MAX

O módulo de INTLIB-ARITHMETIC é o que dá suporte às rotinas INTLIB. Este módulo é usado no progrma abaixo. Este programa usa o compilador Fortran 90 para testar intervalos e produz uma saída.

INTLIB	
Rotinas de funções padrão para intervalos	
Nome	Função
IACOS	Arco co-seno
IACOT	Arco co-tangente
IASIN	Arco seno
IATAN	Arco tangente
ICOS	Co-seno
IEXP	Exponencial
HIIPOWR	Potência para um intervalo não negativo
ILOG	Logaritmo natural
ISIN	Seno
ISINH	Seno hiperbólico
ISQRT	Raiz quadrada
POWR	Potência inteira

Tabela 4.5: Rotinas de funções padrão para intervalos em INTLIB.

INTLIB	
Rotinas de funções úteis para intervalos	
Nome	Função
ICAP	atribui a um novo intervalo a interseção de 2 intervalos
IDISJ	.verdadeiro. se dois intervalos são disjuntos
IHULL	atribui um novo intervalo a união convexa de 2 intervalos
IILEI	.verdadeiro. se intervalo estiver contido em outro
IILTI	.verdadeiro. se intervalo estiver entre extremos de outro
IINF	atribui o extremo inferior de um intervalo a um real
IMID	ponto médio de um intervalo
IMIG	atribui a um real o extremo maior de um intervalo
IRLEI	.verdadeiro. se x pertencer a um intervalo
IRLTI	.verdadeiro. se um determinado valor estiver no intervalo
ISUP	atribui o extremo inferior a uma real
IVL1	construção de um intervalo degenerado através de um numero real
IVLABS	construção de um intervalo a partir de outro
IVLI	atribuir um intervalo a outro
IWID	diâmetro de um intervalo

Tabela 4.6: Funções úteis para intervalos em INTLIB.

4.6.3 Operadores lógicos

A $INTLIB - ARITHMETIC$ define os seguinte operadores lógicos binários entre os intervalos A e B, e número de precisão dupla C.

.IS.
A.IS.B processa a interseção dos intervalos A e B. Uma condição de erro é sinalizada em INTLIB se a interseção estiver vazia.
.CH.
A.CH.B processa a união entre os intervalos A e B.
.SB.
A.SB.B retorna .TRUE. se A é um subconjunto de B, e retorna .FALSE. Em caso contrário.
.SP.
A.SP.B retorna .TRUE. se A é um superconjunto de B, e retorna .FALSE. Em caso contrário.
.DJ.
A.DJ.B retorna .TRUE. se A e B são disjuntos, e .FALSE. Em caso contrário.
.IN.
C.IN.A retorna .TRUE. se C é um elemento de A, e .FALSE. Em caso contrário.
.LT.
A.LT.B retorna .TRUE. se cada elemento de A é menor que cada elemento de B. Ambos, ou A, ou B podem ser um número de precisão dupla como operador.
.GT.
A.GT.B retorna .TRUE. se cada elemento de A é maior que cada elemento de B. Ambos, ou A, ou B podem ser número(s) de precisão dupla, como operador(es).
.LE.
A.LE.B retorna .TRUE. se cada elemento de A é menor ou igual a cada elemento de B. Ambos, ou A ou B podem ser número(s) de precisão dupla, como operador(es).
.GE.
A.GE.B retorna .TRUE. se cada elemento de A é maior ou igual a cada elemento de B. Ambos, ou A, ou B podem ser número(s) de precisão dupla, como operador(es).
.NE.
A.NE.B. retorna .TRUE. se A e B não representam o mesmo conjunto.

Exemplo: Programa que utiliza a biblioteca INTLIB.

Este programa em Fortran 90 utiliza a biblioteca INTLIB e que tem por objtivo testar o comportamento desta biblioteca. Para facilitar o entendimento, faremos o acompanhamento passo a passo do programa:

1. Define três intervalos A,B,C e um inteiro N.
2. Chama a biblioteca INTLIB pelo comando CALL.
3. Atribui valores aos intervalos A,B e ao inteiro N.
4. Calcula a soma dos dois intervalos e imprime o resultado.
5. Calcula a potência do intervalo elevado a um inteiro N, atribui o resultado ao intervalo C e imprime C.
6. Calcula a potência de um intervalo elevado a outro intervalo, atribui o resultado ao intervalo C e imprime C.
7. Calcula o seno do intervalo A, atribui ao intervalo C o resultado e imprime.
8. Fim lógico do programa.

Observação: Verifica-se que os valores resultantes estão inflacionados, já que a Biblioteca INTLIB, assim procede.

PROGRAM $TEST-INTERVAL-ARITHMETIC$
c Este programa testa um programa em fortran 90 que utiliza
c a biblioteca INTLIB para fazer algumas operações básicas como:
c A aritmética de intervalos na biblioteca INTLIB.

Passos USE $INTLIB-ARITHMETIC$
1. TYPE(INTERVAL) A, B, C
1. INTEGER N
2. $CALL\ SIMINI$
3. A = INTERVAL(1,2)
3. B = INTERVAL(3,4)
3. N = 3
4. C = A+B; WRITE(6,*) C
5. C = A**N; WRITE(6,*) C
6. C = A**B; WRITE(6,*) C
7. C = COS(A); WRITE(6,*) C
8. STOP
END PROGRAM $TEST-INTERVAL-ARITHMETIC$

Listagem de saída do programa acima.

3.9999999999999991 6.0000000000000018
0.9999999999999995 8.0000000000000036
0.9999999999999946 16.0000000000005578
-0.4161468365471566 0.5403023058681523

4.7 C-XSC

4.7.1 Introdução

A linguagem C-XSC [Wie96] é uma ferramenta para o desenvolvimento de algoritmos numéricos que permite enorme precisão e verificação automática dos resultados. Contém um grande número de tipos e operadores numéricos. Estes tipos são implementados como classes de C++. Assim, a extensão C-XSC permite programação de alto-nível de aplicações numéricas em C e C++. O pacote C-XSC está disponível para muitos computadores que tenham um compilador de C++ da ATT standard 2.0. A linguagem científica C-XSC fornece um número grande de tipos dados e contém aproximadamente 1.600 operadores predefinidos com alta precisão.

Algumas deficiências na linguagem de programação C a fazem imprópria para a programação de algoritmos numéricos. A linguagem C não fornece as estruturas de dados numéricos para vetores e matrizes. Além disso, a linguagem C (até mesmo o ANSI padrão de C) não especifica a precisão ou a direção de arredondamento dos operadores aritméticos. O mesmo se aplica para a entrada e saída de dados das bibliotecas de funções de C. O padrão ANSI de C não estabelece qual é o tipo de conversão de erro dos dados de entrada ou saída.

A linguagem de programação C++ é uma linguagem orientada a objeto, uma extensão da linguagem C que se tornou muito popular nos últimos anos, mas que não fornece facilidades para a resolução de determinados problemas. O novo conceito de estrutura de dados abstratos (classes) e o conceito de operadores overloaded (sobrecarregados) e funções fornecem a possibilidade para criar uma ferramenta de programação que elimina as desvantagens de C mencionadas anteriormente. A linguagem C-XSC, que é uma extensão C para computação científica, proporciona para a C e para programadores de C++ uma ferramenta para escrever algoritmos numéricos que produzem resultados seguros em um ambiente de programação agradável. As características de C++ de ser orientada a objeto fornecem recursos poderosos adicionais que reduzem o esforço de programação e aumentam a confiança e a clareza dos programas.

Com suas estruturas de dados abstratos, operadores e funções predefinidas, C-XSC fornece uma interface entre a computação científica e as linguagens de programação C e C++. Além disso, C-XSC suporta a programação de algoritmos que inclui automaticamente a solução de problemas matemáticos em certos limites confiáveis. Tal algoritmo fornece uma declaração matemática precisa sobre a verdadeira solução.

4.7.2 Características básicas da linguagem C-XSC

A extensão C-XSC possui as seguintes características básicas:

- Os tipos real, complexo, intervalo e aritmética de intervalo complexo com propriedades matematicamente definidas.
- Vetores dinâmicos e matrizes.
- Subarrays de vetores e matrizes.
- Tipos com precisão após a vírgula (Dotprecision).
- Operadores aritméticos predefinidos com grande precisão.
- Funções padrão com grande precisão.
- Controle de arredondamento dos dados de entrada e saída.
- Aritmética dinâmica de múltipla precisão.
- Biblioteca que contém rotinas para resolução de problemas.

4.7.3 Tipos de dados padrão, operadores predefinidos e funções

A linguagem C-XSC fornece os tipos de dados numéricos simples como real, interval, complex, e cinterval. Cada um com suas aritméticas apropriadas, operadores relacionais e funções matemáticas padrão. Todos os operadores aritméticos predefinidos entregam resultados com uma precisão de pelo menos 1 ulp (unidade na última casa decimal). Assim, eles têm a máxima precisão na computação científica nos seus tipos.

Os arredondamentos dos operadores aritméticos podem ser controlados usando os tipos de dados interval e cinterval. Existem funções que estão disponíveis para combinações matematicamente úteis. Podem ser convertidas constantes literais com o máximo de precisão. Todas as funções matemáticas standard para os tipos de dados numéricos simples podem ser chamadas pelos nomes genéricos e têm garantia de alta precisão para os resultados de seus argumentos.

Em C-XSC estão disponíveis tipos de dados numéricos para vetores e matrizes reais e complexas:

rvector, ivector, cvector, e civector, rmatrix, imatrix, cmatrix, e cimatrix.

O usuário pode alocar ou desalocar espaço de armazenamento para um array dinâmico (vetor ou matriz) em tempo de execução. Assim, sem necessidade de recompilação, o mesmo programa pode usar arrays de tamanhos só restringidos pelo armazenamento do computador. Além disso, a memória é usada eficazmente desde que os arrays sejam armazenados somente nos tamanhos exigidos por eles.

4.7.4 Subarrays de vetores e matrizes

A ferramenta C-XSC fornece uma notação especial para manipular subarrays de vetores e matrizes. Subarrays são partes retangulares arbitrárias dos arrays. Operadores predefinidos também podem usar subarrays como operandos. Um subarray de uma matriz ou vetor é acessado utilizando o operador ( ) ou o operador [ ]. O operador ( ) especifica um subarray de um objeto do mesmo tipo que o objeto original.

4.7.5 Avaliação de expressões com alta precisão

Ao avaliar expressões aritméticas, a precisão faz um papel decisivo em muitos algoritmos numéricos. Até mesmo se todos os operadores de aritmética e funções padrão forem de precisão máxima, expressões compostas de vários operadores e funções necessariamente não fornecem resultados com precisão máxima. Então, foram desenvolvidos métodos para avaliar expressões numéricas com precisão alta e matematicamente garantida.

Para obter uma avaliação com precisão de 1 ulp, a extensão C-XSC provê os tipos de dados de boa precisão. Como veremos a seguir:

dotprecision, cdotprecision, idotprecision, e cidotprecision.

4.7.6 Aritmética dinâmica de múltipla precisão

Além das classes real e interval, em C-XSC existem as classes dinâmicas de maior precisão ou reais longos (I_real) e intervalo longo (I_interval), como também são implementados os vetores dinâmicos correspondentes e matrizes, inclusive toda a aritmética, operadores relacionais e funções standard de múltipla-precisão.

Ex: include << *interval.hppp* >>

Main() interval a, b; /* intervalos degenerados constantes
a = 1.0; /* a = [1.0,1.0]
b = 3.0; /* b = [3.0,3.0]
cout << "*a/b=* "<< *a/b* = /* a/b = [0.333333333333,0.333333333333]

C-XSC	
Funções matemáticas padrão	
Função	Nome Genérico
sin	seno
cos	co-seno
tan	tangente
cot	co-tangente
asin	arco seno
acos	arco co-seno
atan	arco tangente
acot	arco co-tangente
sinh	seno hiperbólico
cosh	co-seno hiperbólico
tanh	tangente hiperbólica
coth	co-tangente hiperbólica
asinh	seno hiperbólico inverso
acosh	co-seno hiperbólico inverso
atanh	tangente hiperbólica inversa
acoth	co-tangente hiperbólica inversa
sqr	quadrado
power	potência
exp	exponencial
pow	potência
sqrt	raiz quadrada
sqrtn	raiz de n
ln	logaritmo natural
abs	valor absoluto

Tabela 4.7: Funções matemáticas padrão em C-XSC.

Ex:include $<< interval.hppp >>$

Main()

I_interval a, b; /* intervalo com múltipla precisão
a = 1.0; /* a = [1.0,1. 0]
b = 3.0; /* b = [3.0,3.0]
cout $<<$ "$a/b =$ "$<<$ a/b=
a/b = [0.3333333333333333333333333;0.3333333333333333333333333]

4.7.7 Entrada e saída em C-XSC

Usando o conceito de fluxo e os operadores de overloadable $<< e >>$ de C++, a linguagem C-XSC fornece arredondamento e controle durante I/0 (input/output) para todos os tipos de dados.

4.7.8 Biblioteca com rotinas para a resolução de problemas

O C -XSC contém uma biblioteca com uma coleção de rotinas para problemas standard de análise numérica e que produzem resultados com alta precisão. As seguintes áreas estão cobertas:

- Avaliação de zeros de polinômios.
- Matriz inversa, sistemas lineares.
- Transformações de Fourier.
- Zeros de uma equação não-linear.
- Sistemas de equações não-lineares.
- Problemas de valores iniciais em equações diferenciais ordinárias.

4.7.9 Conclusões

Em contraste com C e C++, todos os operadores predefinidos de aritmética, especialmente o vetor e operações de matriz, devolvem um resultado de pelo menos precisão de 1 ulp em C-XSC. Não há necessidade de aprender as características novas de C++ para poder usar a ferramenta C-XSC, que é o programa ambiente para aplicações numéricas. Na maioria dos casos, o conhecimento da linguagem C é suficiente para se trabalhar com C-XSC.

O usuário avançado pode estender C-XSC usando programação orientada a objeto que é característica de C++. Podem ser combinados programas escritos em C-XSC com qualquer outro software de C++, se algumas regras de programação elementares forem respeitadas. Os programas de C-XSC sempre fornecem os mesmos resultados numéricos compatíveis em computadores com diferentes compiladores C++. Quer dizer, C-XSC é uma ferramenta para alcançar a compatibilidade completa de resultados numéricos na matemática intervalar.

4.8 Outros pacotes modernos

Vários outros pacotes de intervalo, além de C-XSC, têm-se construído usando C++, pois a mesma foi uma das primeiras linguagens extensamente disponíveis com tipo de dados definidos pelo usuário e operador de overloading. Jeff Ely [Ely93] desenvolveu um pacote em matemática intervalar com precisão variável, o qual tem sido portado a várias máquinas e vetorizado em um Cray YMF.

Anthony Leclerc [Lec93] construiu um pacote em C++ que tem sido portado para vários tipos de estações de trabalho, e é usado para resolver problemas difíceis de otimização global em paralelo, em uma rede distribuída de estações de trabalho.

Jüllig criou, em C++, o pacote conhecido como BIBINS [Jul91] para computações intervalares que inclui módulos para vetor e aritmética de matriz.

Em Ada, no pacote GENERAL_SCIENTIFIC_COMPUTATION, desenvolvido em colaboração com o suporte da Comunidade Europeia [Kok90], a ênfase está em um produto de boa precisão, operações de vetor e de matriz, enquanto a biblioteca de funções padrão de intervalo parece estar ausente. George Corliss desenvolveu um pacote em Ada para aritmética intervalar e aritmética de Taylor.

O pacote Modula-SC [FGK+92] é uma extensão da linguagem Modula-2, que provê um produto de boa precisão, como também as operações de alta-ordem (vetor e matriz).

4.8.1 Biblioteca Libavi

No Brasil, foi desenvolvida a partir de 1993 baseada em FORTRAN 90, uma biblioteca intervalar conhecida como libavi [Lis97, Sag97], no Centro Nacional de Supercomputação (CESUP/RS), com o uso do supercomputador Cray. A biblioteca intervalar libavi compreende não só a biblioteca libavi de rotinas intervalares, mas também uma rotina para álgebra linear utilizando intervalos. A biblioteca libavi tornou-se disponível apenas no final do primeiro semestre de 1997, quando o núcleo de aritmética de alta exatidão foi incorporado à biblioteca. A libavi.a é, atualmente, uma ferramenta poderosa para a resolução de problemas práticos de grande parte dos pesquisadores do Centro de Supercomputação da UFRGS.

4.9 Pacotes para manipuladores simbólicos

Intervalos e computações simbólicas usam computadores para obterem declarações matematicamente rigorosas. Como computações intervalares se beneficiam a priori de um rearranjo e simplificações de expressões e algoritmos, como o método de Newton intervalar, requerem uma avaliação de funções e derivadas com tipos de dados diferentes. Desse modo, é natural considerar computações intervalares dentro de um ambiente de computação simbólica.

No estado presente, pacotes de manipulação simbólica deveriam ser usados com um pouco de cuidado, uma vez que simplificações internas nas expressões algébricas podem resultar em valores de intervalo que não correspondem às expressões algébricas exibidas. Isso pode implicar que os pacotes possam rearranjar expressões sem falar para o usuário, e tais rearranjos comportem extensões de intervalos diferentes. Além disso, programas que incluem ciclos, escritos nestas linguagens, executam milhares de vezes mais lentamente do que programas correspondentes em linguagens como FORTRAN90.

Capítulo 5

A linguagem JAVA

A independência de plataforma, que é uma característica própria da linguagem JAVA [Roc96], (as outras linguagens de programação não a possuem), é uma vantagem excepcional principalmente na utilização da matemática intervalar. Na resolução de problemas, é de suma importância que processamentos em máquinas diferentes levem a resultados iguais. Daí a vantagem de se ter uma biblioteca para computações intervalares desenvolvida nesta linguagem.

Em outras linguagens de programação, por exemplo em C, quando se compila um programa, o compilador transforma seu arquivo-fonte em código de máquina, ou seja, em instruções específicas para o processador que seu computador está executando. Ao se compilar seu código em um sistema baseado em Intel, o programa resultante será executado em outros sistemas baseados em Intel, mas não funcionará em computadores Macintosh, Commodore VIC-20 ou em outras máquinas. Se você quiser usar o mesmo programa em outras plataformas, deve transferir seu código fonte para a nova plataforma e recompilá-lo para produzir o código de máquina específico para esse sistema. Em muitos casos, serão exigidas alterações no código-fonte antes que ele seja compilado na nova versão da máquina, devido às diferenças em seus processadores e outros fatores. Todos estes problemas supracitados poderão levar a uma diferença nos resultados que pode causar uma inexatidão dos mesmos.

A vantagem dos programas em JAVA é que estes atingem essa independência através da utilização de uma máquina virtual, uma espécie de computador dentro do outro. A máquina virtual toma os programas JAVA compilados e converte suas instruções em comandos que um sistema operacional possa manipular. O mesmo programa compilado, que existe em um formato chamado "bytecode", pode ser executado em qualquer plataforma e sistema operacional que possua uma máquina virtual JAVA.

Desta maneira, utilizar-se-á o mesmo bytecode em um browser que roda em um

Mac, um PC, ou em uma workstation da Silicon Graphics. O arquivo bytecode multiplataforma é somente um aspecto da portabilidade da linguagem JAVA. Os projetistas da mesma também tiveram um trabalho extra para eliminar qualquer dependência de plataforma da linguagem.

Assim, não se encontrará na linguagem de programação JAVA qualquer referência específica a hardware.

5.1 História

A linguagem JAVA [Roc96] nasceu das cinzas de um projeto fracassado. Inicialmente foi desenvolvida para programar chips de aplicativos de consumo, como Pdas, fornos de microondas, etc. A equipe que desenvolvia o projeto na Sun, liderada por James Gosling, tinha inicialmente pensado em utilizar C ou C++, mas descobriu logo cedo que a linguagem não se adequava ao projeto. Como resultado, em 1990, Gosling começou a projetar uma nova linguagem que batizou de Oak, segundo ele, por causa da árvore que via da sua janela. O nome posteriormente teve que ser mudado por que já existia uma outra linguagem registrada com o mesmo nome.

O projeto avançou, mas o mercado decepcionou, e a Sun decidiu redirecionar o rumo do projeto investindo no ramo da TV interativa. O programa durou dois anos e consumiu milhões de dólares. Consistia no desenvolvimento de um protótipo de controle para TV interativa, usando uma linguagem desenvolvida especialmente para este fim (Oak). Quando o projeto finalmente foi apresentado em 1993, descobriu-se que o mercado de TV interativa não existia, e não havia uma previsão para quando iria deslanchar. O financiamento para o projeto estava para ser cortado e a equipe transferida para outras pesquisas quando a Sun decidiu abandonar o hardware do projeto e focar o seu desenvolvimento na Web.

A partir daí, a equipe trabalhou em um ritmo alucinante e, em segredo, até liberar a primeira versão Alfa da linguagem, em maio de 1995.

5.2 A linguagem da internet

Grande parte do sucesso da linguagem JAVA surgiu na Internet. As primeiras aplicações desenvolvidas em JAVA foram pequenos programas chamados applets, utilizados para dar maior interatividade às páginas Web, acessadas através do browser HotJAVA, escrito totalmente em ferramenta JAVA. Em uma fase posterior, o Netscape Navigator, browser de maior sucesso na Internet, implementou a API do

HotJAVA e passou a suportar a execução de applets nas suas páginas. A partir daí, passou a ser comum acessar a Web e encontrar um logotipo se movendo, um relógio musical, ou jogar uma partida de Tetris em uma página Web.

A ferramenta JAVA pegou carona na euforia já existente com relação à Web e à Internet. A sua enorme popularidade se deve principalmente ao fato de ter a Web e a Internet como seu principal foco. As primeiras notícias que se ouviram da aplicação JAVA era que ela possibilitava colocar animações em uma página Web. Ela fazia isso através de um descritor HTML especial ($< applet >$), aceito pelo browser HotJAVA, que permitia incluir um programa em uma página, da mesma maneira que se coloca uma imagem. O programa era transferido e rodava na máquina do usuário. A sua aplicação não se limitava a animações. O que se podia fazer com a linguagem JAVA dependia apenas da imaginação do programador. Esses programas, que rodavam dentro das páginas Web, foram chamados de Applets.

As pessoas começaram a ver que o mundo estava cheio de applets, quando o Netscape Navigator, browser usado por 86% da população da Web na época, passou a suportá-los também .

Muitos ainda achavam que a linguagem JAVA era só mais um brinquedo para tornar a Web mais atraente. O interesse cresceu quando as pessoas começaram a descobrir que JAVA era mais que isso. Era uma poderosa linguagem que podia ser usada para desenvolver aplicações, como as que se fazem hoje usando C++. E mais: a linguagem JAVA é muito mais simples, segura e robusta, e possui uma qualidade que a diferencia de todas as outras - a independência de plataforma. Com ela, é possível desenvolver um único programa e rodá-lo em qualquer lugar. Essa é a filosofia da linguagem: "Escreva uma vez. Rode em qualquer lugar."

Desde então, a evolução não parou. As maiores empresas de software do mundo, inclusive a poderosa Microsoft, licenciaram JAVA. Várias delas já têm projetos para desenvolver aplicativos de grande porte, em um futuro próximo, usando a programação JAVA. Já existe uma interface independente para acesso a bancos de dados, suportada pelas maiores empresas do ramo. Tudo isso, em menos de um ano de existência.

A linguagem JAVA nasceu na rede e foi feita para a rede. Enquanto outras linguagens são limitadas pela arquitetura da máquina onde processam, a linguagem JAVA executa em qualquer lugar e fala a linguagem da Internet. O suporte à rede e à Web faz parte da linguagem. A facilidade de ler um arquivo local ou uma URL remota é a mesma. É uma linguagem que já nasceu poliglota. O seu conjunto de caracteres de 16 bits suporta, desde o início, todos os idiomas conhecidos no mundo.

5.3 A máquina virtual x BYTECODES

Mas a linguagem JAVA não é só uma linguagem de programação para dar maior brilho às páginas Web. É uma poderosa ferramenta de programação orientada a objetos do porte de C++, porém muito mais simples. Pretende acabar com a dependência que os softwares de hoje têm em relação ao sistema operacional. Aqueles programas "for Windows" serão coisa do passado, pois se terá a liberdade de escolher o sistema onde seu programa irá rodar.

Com a grande variedade de máquinas e sistemas ligados à Internet, é imenso o tempo perdido desenvolvendo aplicativos para tantas plataformas diferentes. A ferramenta JAVA veio para acabar de vez com esse problema. Dentro da filosofia, como já citamos, "escreva uma vez; processe em todo lugar", o programador desenvolve um único aplicativo na ferramenta, compila, e executa em qualquer plataforma Unix, Windows, NT, AIX, OS/2, e outras.

Como se consegue isto? Através da idéia de uma máquina virtual JAVA. Toda plataforma que tiver uma máquina virtual JAVA tem a habilidade de processar qualquer programa compilado em JAVA. Não se escreve um programa na linguagem JAVA para uma determinada plataforma. Faz-se para a máquina virtual, seja qual for a plataforma onde ela está instalada. Qualquer aplicação escrita em JAVA é independente de plataforma. Somente a máquina virtual precisa ser transportada. Ela é a interface entre o computador do usuário e a aplicação JAVA. Desenvolver uma aplicação em JAVA significa escrevê-la uma única vez e rodá-la em qualquer lugar.

A máquina virtual interpreta instruções simples com comprimento de um byte chamadas bytecodes. O compilador JAVA converte os métodos JAVA em bytecodes que são armazenados em um arquivo *.class* juntamente com todas as informações de tipos para uma classe. Os bytecodes não dependem de uma determinada arquitetura de processador. Por serem instruções pequenas, são independentes da ordem dos bytes, logo podem ser executadas da mesma forma, tanto em máquinas Big-Endian como em máquinas Little-Endian.

Uma das principais diferenças entre JAVA e C/C++ é o processo de compilação e execução. A compilação de um programa em C++ traduz o código-fonte em instruções que são interpretadas pelo processador do computador onde irá rodar. Já o processo de compilação de um programa em JAVA traduz o código-fonte em instruções (chamadas de bytecodes) que são interpretadas por um sistema de execução que simula o processador de um computador virtual genérico.

O computador virtual é uma camada de software que interpreta os bytecodes e pode estar implantada sobre qualquer plataforma real. Por ocorrer ao nível de software, a interpretação de bytecodes tende a ser bem mais lenta que a interpretação direta de linguagem de máquina nativa pelo processador. Apesar disso, devido à semelhança dos bytecodes com instruções de máquina, o desempenho é o melhor em relação a outras linguagens interpretadas.

O processo de interpretação dos bytecodes é totalmente independente da compilação. Portanto, nada impede também que se desenvolvam compiladores que convertam outras linguagens para bytecodes. Esses bytecodes serão 100 % compatíveis com os bytecodes gerados por um compilador JAVA e serão igualmente independentes de plataforma.

O código-fonte em JAVA também pode ser compilado diretamente para a linguagem de máquina nativa, embora assim, perca a sua característica principal que é a independência de plataforma. Mesmo assim, não deixa de ser uma boa alternativa para garantir um maior desempenho em trechos de código que exigem muito do sistema. Desempenho deve se tornar, em breve, um problema menor em JAVA com a chegada das máquinas virtuais, capazes de traduzir bytecodes para linguagem de máquina em tempo de execução.

5.4 Características da linguagem JAVA

5.4.1 Básicas

1. **Orientada a objetos**

   A linguagem JAVA é um membro do paradigma orientado a objetos (OO) das linguagens de programação. As linguagens que aceitam este paradigma, como JAVA e C++, seguem a mesma filosofia básica, mas diferem em sintaxe e estilo. As linguagens orientadas a objetos oferecem muitas vantagens sobre as linguagens procedurais tradicionais. Como os objetos encapsulam dados e funções relacionados em unidades coesas, é fácil localizar dependências de dados, isolar efeitos de alterações e realizar outras atividades de manutenção, e talvez, o mais importante, as linguagens OO facilitam a reutilização.

   É uma verdadeira linguagem de programação orientada a objetos. Tudo em JAVA é objeto. A única exceção são os tipos simples e primitivos como números e variáveis booleanas, que não são objetos por questões de desempenho. Mesmo assim, podem ser encapsulados em objetos quando for necessário.

   Os programas escritos em JAVA são organizados em classes, que podem ser instanciadas para produzir objetos. As classes também podem herdar características (métodos e variáveis) de outras classes.

A linguagem JAVA não suporta herança de múltiplas classes mas permite que uma classe implemente mais de uma interface, que é um tipo especial de classe que não contém detalhes de implementação.

Em JAVA, não existem variáveis globais ou funções independentes. Toda variável ou método pertence a uma classe ou objeto e só pode ser invocada através dessa classe ou objeto.

2. **Distribuído**

   Distribuição de informações para compartilhamento e trabalho conjunto, com a distribuição de carga de trabalho do procesamento, é uma característica essencial dos aplicativos cliente/servidor.

   Felizmente, para os programadores JAVA, há uma biblioteca de procedimentos TCP/IP incluída nos códigos-fonte e de distribuição binária do JAVA. Isso facilita aos programadores o acesso remoto às informações, usando protocolos como HTTP e FTP.

3. **Compilado e interpretado**

   Os programas do JAVA são compilados em formato binário de código de bytes, que então são interpretados por um ambiente de execução do JAVA específico da plataforma em questão. Portanto é, ao mesmo tempo, compilado e interpretado.

4. **Independente de arquitetura**

   A neutralidade do JAVA em relação à arquitetura é fascinante, mas não se trata de um conceito novo. Derivado da natureza distribuída de cliente/servidor, um importante recurso de projeto do JAVA é o suporte a clientes e servidores em configurações heterogêneas de rede. O método escolhido para atingir esse objetivo foi uma representação binária de arquitetura neutra para os programas em JAVA.

5. **Portável**

   A característica de neutralidade da arquitetura JAVA é o grande motivo pelo qual os programas em JAVA são portáveis. Outro aspecto da portabilidade envolve a estrutura ou os tipos de dados inerentes da linguagem, como

inteiro, string e ponto flutuante.
O compilador JAVA foi escrito com o próprio JAVA, enquanto seu ambiente de tempo e execução foi escrito em ANSI C e tem uma interface de portabilidade bem definida e concisa.

6. **Multitarefa**

   Os objetos binários de códigos de bytes do JAVA são formados por seqüências de execução múltiplas e simultâneas. Essas seqüências são conhecidas como contextos de execução ou processos leves. As linguagens C e C++ são membros de um paradigma de execução em seqüência única por não oferecerem suporte a seqüências no nível de linguagem. O JAVA, no entanto, oferece suporte no nível de linguagem para multitarefas, resultando em uma abordagem de programação mais poderosa e de múltiplas facetas.

7. **Dinâmica**

   O projeto dinâmico permite que os programas JAVA se adaptem aos ambientes computacionais mutantes. Por exemplo, a maior parte dos desenvolvimentos típicos em C++ se baseia muito em bibliotecas de classe que podem ser desenvolvidas por terceiros. Muitas bibliotecas de terceiros, como as distribuídas com sistemas operacionais ou sistemas de janelas, são linkeditadas dinamicamente e vendidas ou distribuídas separadamente dos aplicativos que delas dependam. Quando essas bibliotecas são atualizadas, os aplicativos que dependerem delas poderão apresentar problemas, até que sejam recompilados e redistribuídos. Isso adiciona mais um custo à manutenção do software.

   A ferramenta JAVA evita esse problema atrasando a união dos módulos. Isso permite que os programadores tirem total proveito da orientação a objetos. É possível adicionar novos métodos e variáveis de instâncias em classes de bibliotecas sem causar problemas aos programas, aplicativos ou clientes já existentes.

8. **Robusto**

   Quanto mais robusto um aplicativo, mais confiável ele será. Isso é desejável tanto para os desenvolvedores de software quanto para os consumidores. A maioria das linguagens OO, como o C ++ e JAVA, possuem tipos bastante fortes. Isso significa que a maior parte da verificação de tipos de dados é realizada em tempo de compilação, e não em tempo de execução. Isso evita muitos erros e condições aleatórias nos aplicativos. A linguagem JAVA, ao contrário do C ++, exige declarações explícitas de métodos, o que aumenta a confiabilidade dos aplicativos. Quando o programa JAVA falha, o computador não trava e não é gerado arquivo "core", e somente uma exceção é gerada, além de que é fácil fazer "trace" na pilha para descobrir o ponto de falha do programa.

9. **Segurança**

   Como o JAVA foi criado para ambientes de rede, os recursos de segurança receberam muita atenção. Por exemplo, ao se executar um binário transferido por download da rede, o mesmo poderá estar infectado por vírus. Os aplicativos JAVA apresentam garantia de resistência contra vírus e de que não são infectados pois não são capazes de acessar heaps, stacks ou memória do sistema. No JAVA, a autenticação do usuário é implementada com um método de chave pública de criptografia. Isso impede de maneira eficaz que hakers e crakers examinem informações protegidas como nomes e senhas de contas. A sua segurança também reside no fato de não ter ponteiros, sua tabela de nomes de funções é gerada em tempo de execução e também as posições de memória determinadas em tempo de execução. Não existem none, applet host, firewall, unrestricted.

10. **Simples**

   Um dos principais objetivos do projeto do JAVA foi criar uma linguagem o mais próxima possível do C ++, para garantir sua rápida aceitação no mundo do desenvolvimento OO. Outro objetivo do seu projeto foi eliminar os recursos obscuros e danosos do C ++, que fugiam à compreensão, e aumentavam a confusão que poderia ocorrer durante as fases de desenvolvimento, implementação e manutenção do software. Apesar da semelhança, a linguagem JAVA não herdou as complexidades da C++. Todos os recursos desnecessários do C++ (como estruturas, unions e typedef) foram deixados de lado. A linguagem JAVA não tem apontadores explícitos, arquivos de cabeçalho, pré-processadores, estruturas, uniões, matrizes multidimensionais, gabaritos nem sobrecarga de operadores.

11. **Compacto**

   O interpretador básico JAVA ocupa aproximadamente 40k de RAM, excluindo-se o suporte, a multitarefa e as bibliotecas padrão que ocupam outros 175k. Mesmo a memória combinada de todos esses elementos é insignificante, se comparada a outras linguagens e ambientes de programação. A maioria das aplicações JAVA cabem em 1 Mb de ROM e 1 Mb de RAM.

12. **Alto desempenho**

    Há muitas situações em que a interpretação de objetos de códigos de bytes proporciona desempenho aceitável. Mas outras circunstâncias exigem desempenhos mais altos. O JAVA concilia tudo isso oferecendo a tradução dos códigos de bytes para o código de máquina nativo em tempo de execução. O alto desempenho permite a implementação de seus aplicativos Web em JAVA, na forma de programas pequenos e velozes, que podem ampliar significantemente os recursos tanto do cliente quanto do servidor.

5.4.2 Adicionais

A linguagem JAVA possui controle automático de limites de arrays e as suas interfaces das classes são idênticas em todas as plataformas.

1. **Tipagem estática**

   Os tipos de objetos (caracteres, arrays, etc.) devem ser definidos quando o código é escrito. A ferramenta JAVA permite maior controle em tempo de compilação; detecção de vários erros, mais cedo, no ciclo de desenvolvimento, e percebe a violação de limites de arrays.

2. **Interfaces**

   As interfaces para todas as classes JAVA padrão são as mesmas em todas as plataformas de modo que os programadores podem fazer uso de todos os recursos disponíveis, sem se preocuparem com a plataforma.

5.4.3 Avançadas

1. **Exceções**

   A linguagem JAVA implementa exceções do mesmo estilo do C++, ajudando na construção de "código seguro". Quando um erro ocorre, pode ser tratado adequadamente através da exceção gerada.

2. **Interface gráfica**

   A linguagem JAVA implementa vários elementos usados para construir a interface gráfica do usuário tais como: caixas de entradas de dados, botões, rótulos (usar import JAVA.awt.*), listas, campos de texto, barras rolantes e tabelas.

3. **Frames**

   Em JAVA, Frame é a janela básica usada para criar uma interface gráfica com o usuário. No appletviewer da Sun, a classe Frame é a classe básica usada na construção da interface utilizada pelo applet que está sendo visualizado.

   JAVA.lang.Object
   JAVA.awt.Component
   JAVA.awt.Container
   JAVA.awt.Window
   JAVA.awt.Frame
   sun.applet.AppletViewer

4. **Threads e multhreads**

   A ferramenta JAVA foi projetada com uma série de recursos para suportar a implementação de multithreading ao nível de programação. Ou seja, pode-se definir diferentes trechos do programa que executarão paralelamente. Contém recursos para garantir a integridade dos dados e fazer a divisão de uma aplicação em threads, uma tarefa simples. Primitivas da linguagem controlam a sincronização automática e o travamento de métodos e estruturas críticas. Qualquer método declarado synchronized recebe uma trava que impede seu uso por duas ou mais linhas de execução ao mesmo tempo.

5. **Entrada e saída em arquivos**

   Todas as linguagens fornecem alguns mecanismos de interação com o sistema de arquivos local; a linguagem JAVA não é uma exceção. Classes de objetos de manipulação de I/O estão no pacotes JAVA.io.

6. **Acesso a bases de dados**

   Denomina-se JDBC a interface padrão para acesso a bases de dados SQL e que permite que aplicações JDBC executem usando drivers ODBC existentes.

5.5 Diferenças entre as linguagens JAVA e C/C++

Para quem conhece C++, num primeiro momento a linguagem JAVA parece mais um novo dialeto que uma nova linguagem. O programador C/C++ imediatamente se sente familiarizado com a sua estrutura e palavras-chave. Esta familiaridade tem sido uma das principais responsáveis pelo grande sucesso da linguagem entre os programadores C/C++. Muitos já acreditam que a linguagem JAVA é a virtual sucessora do C++ como linguagem de programação preferida entre os desenvolvedores de aplicações.

Mas as semelhanças param por aí. Ela é uma nova linguagem, muito mais robusta e muito mais simples que a C/C++. Elimina uma gama de redundâncias e complicações da C/C++, e introduz novos recursos que facilitam o seu uso no desenvolvimento de aplicações distribuídas. Esta ferramenta de programção JAVA não tem aritmética de apontadores, estruturas nem pré-processador. Possui um modelo de gerenciamento de memória otimizado que dispensa o programador da tarefa de liberar memória. Como resultado, programas escritos em JAVA têm menos bugs, não têm vazamentos de memória e não provocam core-dumps ou GPFs.

Diferentemente da linguagem C++, que é uma mera extensão da C com suporte a classes e objetos, a ferramenta JAVA é uma verdadeira linguagem de programação orientada a objetos. Não há como escapar das classes e objetos em JAVA. Eles estão tão presentes na linguagem que é impossível fazer um programa sem utilizá-los. Todo método tem que estar dentro de um objeto. Até o método main(), que define o corpo do programa, tem que ser encapsulado em um objeto. Os tipos primitivos (int, long, float, etc.), que não são objetos, são encapsulados objetos (Integer, Long, Float, etc.) quando preciso, para permitirem também a sua manipulação como objetos.

A orientação a objetos simplifica os procedimentos em JAVA. Como tudo é objeto, há uma sintaxe consistente que pode ser utilizada em todo lugar. Não é como C++, onde há formas diferentes de lidar com apontadores, referências e valores. Em JAVA, não há apontadores e apenas os tipos primitivos recebem valores. Todo objeto em JAVA é acessado através de uma referência a ele, e todas as operações sobre objetos ocorrem sobre suas referências.

A referência pode ser entendida como uma espécie de apontador para o objeto, mas não deve ser confundida com os apontadores em C/C++. Enquanto os apontadores C/C++ lidam diretamente com um modelo de posições de memória física, as referências em JAVA são apontadores para objetos dinamicamente alocados. Não há como saber o endereço das posições físicas de memória ocupadas por eles.

5.5.1 Destrutor

Uma diferença importante é a ausência de um destrutor na classe JAVA. Não é necessário destruir objetos criados em JAVA pois objetos não mais utilizados são automaticamente destruídos por um mecanismo de "coleta de lixo", e a memória ocupada por eles é automaticamente liberada. Sendo assim, a classe não precisa um método destrutor. Caso seja necessário realizar algo antes de destruir o objeto, é possível sobrepor o método finalize() (definido na classe Object e portanto herdado por todas as classes) com a finalização desejada.

5.5.2 Condições excepcionais

O controle de condições excepcionais não é característica exclusiva da linguagem JAVA. Seu uso é permitido em C++. A diferença é que em JAVA o controle de exceções é obrigatório. Se o uso de um método provoca uma exceção, a linguagem JAVA não deixa compilar o programa, enquanto não se incluir um trecho de código que instrua o programa a lidar com ela ou declarar explicitamente que o método provoca tal exceção.

O controle de exceções torna o código mais robusto, permitindo lidar com ocorrências anormais em poucos pontos do programa, em vez de ter que se preocupar com o que vai acontecer em cada linha de execução.

5.5.3 Ausência de pré-processador

Em JAVA não existe nada parecido com os pré-processadores utilizados em C e C++. Todas as informações que definem uma classe, seus métodos, construtores e variáveis são incluídos em um único arquivo. Pode-se implementar várias classes em um único arquivo-fonte, mas o compilador criará cada classe em um arquivo-bytecode distinto.

5.5.4 Herança múltipla

Diferentemente da linguagem C++, a herança de métodos e variáveis de mais de uma super-classe não está implementada em JAVA. Porém, é possível utilizar constantes e métodos que foram implementados em classes diferentes através das interfaces.

Interfaces são um novo tipo de referência que especificam métodos e constantes porém não definem qualquer especificação. Todos os métodos das interfaces são implicitamente abstratos e necessitam ser implementados nas classes que os utilizam.

5.6 Microprocessadores JAVA

Existe uma família de microprocessadores otimizados para operarem com JAVA. São três grupos de microprocessadores: picoJAVA, microJAVA e ultraJAVA.

ultraJAVA

O microprocessador ultraJAVA foi projetado para equipamentos eletrônicos básicos (telefones celulares, impressoras, e outros periféricos). Oferece para a indústria a melhor relação preço/performance. É de baixo custo, em torno de 25 dólares.

picoJAVA

É baseado no núcleo do ultraJAVA, adicionando características específicas (I/O, memória, funções de comunicação e controle). Ideal para dispositivos baseados em rede (controladores e equipamentos usados por empresas de telecomunicação) e jogos simples. Seu preço varia entre 25 e 100 dólares.

microJAVA

É um processador mais rápido (performance de 3 a 5 vezes melhor do que a do picoJAVA). Ideal para trabalhos com gráficos/imagens 3D e aplicações que usam intensamente recursos multimídia interativa. Preço médio de 100 dólares.

5.7 Desempenho da linguagem JAVA

Um fator que interessa a todos os que pensam seriamente em desenvolver aplicações em JAVA é a questão do desempenho. Como lidar com uma velocidade de execução de 5 a 30 vezes menor que programas em C++ equivalentes? Esta diferença pode até passar despercebida nos programas que implementam GUIs, e que passam a maior parte do tempo parados, esperando a ação do usuário. Mas, quando é necessária uma grande quantidade de processamento, logo se sente o peso da linguagem interpretada.

Nestes casos, pode-se implementar a parte mais pesada do código em linguagem nativa, ou compilar todos os bytecodes para a linguagem de máquina da plataforma onde o programa executa. Isto, porém, acaba com a possibilidade de rodar o programa inalterado em outras plataformas, embora o processo de porte continue simples e consista apenas de uma compilação. O novo sistema converte os bytecodes em linguagem de máquina durante o tempo de execução. Haverá um pequeno atraso na primeira vez que um método for chamado, em função do processo de compilação, mas daí em diante, o desempenho será o melhor possível, devendo atingir os mesmos patamares de programas escritos em C ou C++. De fato, as máquinas virtuais com compilador JIT (compilador Just-in-time, desenvolvido pela JAVASOFT) já desenvolvidas pela Symantec e Borland, mostram isto. A Symantec afirma que, em testes de benchmark, os programas interpretados pela sua nova máquina virtual, com compilador JIT, mostraram um desempenho 23 vezes maior, em média, que os mesmos programas interpretados com a versão atual da máquina virtual distribuída pela Sun. A Symantec também conseguiu desenvolver um compilador mais rápido que, segundo ela, leva um décimo do tempo que necessita o compilador JAVAc, da Sun. Atualmente, a máquina virtual JAVA com compilação JIT, ainda é exclusividade do ambiente Windows.

5.8 Ambientes e ferramentas de desenvolvimento JAVA

Há três conjuntos básicos de ferramentas, com as características:

JDK - JAVA Developers Kit

- Classes JAVA.
- JAVA - interpretador JAVA.
- JAVAc - compilador JAVA (cria bytecodes independentes de plataforma).
- appletviewer - visualizador de applets.
- jdb - debugger simples (linha de comandos).
- plataformas: Windows 95/NT, Solaris SPARC, Macintosh.
- gratuito - não tem suporte.

Observação : Em JAVA-XSC, estamos utilizando estas ferramanentas no desenvolvimento da biblioteca.

JAVA WorkShop

- Contém várias ferramentas sofisticadas (escritas em JAVA), integradas através da interface de um browser Web.
- Project Manager- ajuda na criação e organização do projeto e processo de desenvolvimento.
- Build Manager - auxilia na construção e compilação do código fonte. Gera mensagens de erro num formato de hipertexto.
- Source Editor - edita e visualiza o código fonte.
- Applet Tester - carrega e testa os applets.
- Source Browser - monta e apresenta a estrutura de classes de sua aplicação ou applet. Desenha a árvore da hierarquia, e gera uma lista hipertexto dos métodos das classes e dos construtores.
- Debugger - ferramenta de apoio para depuração do código. Está integrada ao Source Editor e com o Applet Tester. Pode-se observar o comportamento do applet à medida que o código é depurado.
- Portfolio Manager - é basicamente um repositório onde podem ser agrupados projetos (applet, aplicação, arquivos).
- Ferramentas a serem adicionadas num futuro próximo: construtor visual de interfaces gráficas (GUI), permitindo fácil geração de código JAVA; editor HTML WYSIWYG; um compilador JIT (Just-In-Time) para converter bytecodes em código de máquina (aumento de velocidade).
- Plataformas: Windows 95/NT, Solaris SPARC, x.86 e Macintosh.

EGOR animator

- fácil de trabalhar.
- cria o arquivo fonte html sozinho.
- plataformas: Windows 95/NT.

5.9 O JAVA Computing: novo paradigma da computação de rede

As empresas que olham o futuro como a chave do seu sucesso estão a desenvolver as suas próprias intranets em torno do mais novo e extraordinário paradigma desta última década: o JAVA Computing. Baseado em aplicações JAVA distribuídas e em dispositivos de rede de baixo custo, o JAVA Computing permite às empresas diminuirem os custos de posse, ao mesmo tempo que desenvolve, distribui e gera as aplicações que são necessárias em todos os desktops que rodam JAVA.

Empresas como a IBM, Microsoft, Novell, SGI e Hewlett-Packard fizeram do JAVA Computing parte integrante das suas estratégias, para os sistemas operativos, o que significa milhares de aplicações JAVA a ajudarem as empresas a capitalizarem os seus lucros. A linguagem JAVA tem sido descrita como originadora da última grande "revolução" na computação. Uma forma ainda mais precisa para descrever o JAVA Enterprise Computing será vê-lo como uma mudança da computação desktop-centric para uma computação network-centric.

Uma das principais conseqüências desta mudança é retirar toda a complexidade concentrada no desktop e removê-la para a rede e para os servidores, onde ela poderá ser centralmente é gerida. JAVA Computing faz tudo isto e, simultaneamente, continua a canalizar para o desktop todo o poder de processamento local necessário e tão importante para as aplicações gráficas e de multimídia. Existe um descontentamento subjacente na computação empresarial dos nossos dias e que tem várias causas. Verifica-se que nas empresas, a computação:

- É demasiado complexa;
- A permanente administração e upgrading dos desktops é bastante onerosa;
- Não é suficientemente segura e confiável;
- As características, funções e aplicações não estão disponíveis para todos os utilizadores;
- As aplicações levam muito tempo para serem desenvolvidas e implementadas.

O paradigma JAVA Computing soluciona todas estas questões e esta é uma das razões pelas quais está sendo empregado como uma nova forma de implementar aplicações. Graças ao fato de ser independente da plataforma, este paradigma pode ser adotado de uma forma cada vez mais crescente sem tornar obsoletos os investimentos já existentes em hardware e software.

Ao mesmo tempo, a ferramenta JAVA pode assegurar que as características e funções estão disponíveis para todos os utilizadores da empresa. O acesso aos front ends aplicacionais faz-se através de um ambiente Webtop comum. Corresponde a um ambiente desktop que pode processar aplicações JAVA, que tanto pode ser um browser preparado para JAVA numa plataforma desktop já existente, como um “desktop thin client”.

Uma outra força colocou a linguagem JAVA na vanguarda da computação. É bastante claro que a Internet se transformou num novo canal para chegar aos clientes, fornecedores e parceiros. A World Wide Web permite à empresa que, de uma forma seletiva, forneça acesso à informação na Internet. A ferramenta JAVA consegue alargar esta caracterização, incluindo acesso seletivo a aplicações de negócios transacionais, através do firewall. A capacidade de oferecer aos clientes, fornecedores e parceiros, uma “janela” criada em JAVA, para a entrega de produtos, serviços e informação advem da adoção de uma arquitetura de computação empresarial baseada em JAVA. Tudo isto acrescenta uma motivação de gestão de topo para a computação empresarial JAVA às constrangedoras questões de economia de custos. No futuro, aqueles investimentos consideráveis em hardware ou software operativo não terão que levar em consideração se uma aplicação específica processa ou não no seu ambiente.

Capítulo 6

JAVA-XSC

6.1 Objetivos

Esta dissertação tem por objetivo desenvolver na linguagem de programação JAVA uma extensão cuja característica principal seja facilitar aplicações da computação científica. Utilizamos o módulo de matemática intervalar do PASCAL-XSC, conhecido como I_ARI somente para determinar que funções seriam implementadas. A biblioteca foi gerada a partir do desenvolvimento da caracterização matemática das operações e funções intervalares em termos dos extremos de seus argumentos conforme estão descritas nos capítulos 2 e 3. Acrescentamos também novas funções intervalares na biblioteca intervalar em JAVA-XSC não existentes no modulo I_ARI.

Como teste e verificação desta nova biblioteca, usaremos alguns métodos intervalares que façam chamadas a esta biblioteca como, por exemplo, o método de Newton intervalar.

São várias as vantagens de acrescentar computações de intervalo à programação JAVA. Uma característica especialmente importante da mesma é ser uma linguagem de programação independente de plataforma. Processando o mesmo programa com uma linguagem de programação diferente de JAVA, e dependente da plataforma, diferentes plataformas com distintas precisões do computador, freqüentemente conduzem a resultados diferentes. Se não tivermos disponíveis intervalos para faixa de erros numéricos, os usuários não saberão, quando eles adquirem resultados diferentes em plataformas diferentes, se as diferenças são o resultado de um algum tipo de "bug", ou se há pouca estabilidade numérica.

Quanto ao inflacionamento, em algumas bibliotecas como a biblioteca **libivi.a** [Lis97] da UFRGS, é feito através da função *nearest* que retorna o mais próximo valor do parâmetro a ela passado. Essa solução também é adotada por outras bibliotecas, tais como a INTLIB [Kea95]. Ganha-se em portabilidade, porém se perde em exatidão. A perda é grande e não compensa a troca, porém pode ser a única alternativa em alguns casos. Até que os fabricantes atentem para o fato, soluções como essas continuarão a comprometer alguns trabalhos.

Mas JAVA-XSC não terá esse problema, porque é uma linguagem que tem portabilidade e que portanto pode ser executada em diferentes sistemas operacionais, versões e plataformas, mantendo um mesmo padrão de exatidão numérica.

Como a linguagem JAVA não possui uma biblioteca intervalar, resulta importante para a computação científica estender esta linguagem com tais bibliotecas.

6.2 Os módulos

Nesta biblioteca agrupamos os métodos intervalares em módulos de acordo com a natureza de suas operações. A biblioteca de rotinas intervalares é composta de seis módulos. Esta estrutura foi baseada na classificação utilizada pela biblioteca Libavi.a [Lis97], conforme veremos a seguir:

Módulo 1-Operações básicas. Contém a definição do intervalo real e as operações aritméticas de adição, subtração, multiplicação e divisão, que servem de base a todos os demais módulos.

Módulo 2-Funções entre conjuntos. São conjuntos de métodos que tratam de operações como intersecção, união, união convexa entre intervalos.

Módulo 3-Funções geométricas. São funções que calculam particularidades geométricas dos intervalos, ou seja, propriedades topológicas no espaço $\mathbb{I}(\mathbb{R})$ como distância, diâmetro e ponto médio.

Módulo 4-Funções elementares. São funções básicas como valor absoluto, raiz quadrada, x elevado ao quadrado, potenciação, exponencial, logaritmo e etc..

Módulo 5-Funções trigonométricas. São funções específicas assim como as hiperbólicas e suas inversas, utilizadas em problemas geométricos.

Módulo 6-Funções de transferência e constantes. São funções que tratam de converter reais em intervalos, ou inteiros em intervalos e vice-versa, como também transferir os extremos inferior e superior de um intervalo para um real. As constantes intervalares são importantes e necessárias durante a execução de alguns tipos de operações intervalares trigonométricas. Daí a necessidade de definirmos estas constantes intervalares como, por exemplo, π, π dividido por inteiro n, e 1 dividido por π.

As tabelas seguintes informam sobre os métodos desenvolvidos dentro da classe IAMath para computações intervalares.

Biblioteca JAVA-XSC			
Módulo 1-Operações básicas			
N^0	Método	Tipo do Resultado	Nome genérico
.	realInterval	(Double, Double)	Intervalo real
1.	add	RealInterval	Soma
2.	uminus	RealInterval	Intervalo Recíproco
3.	sub	RealInterval	Subtração
4.	mul	RealInterval	Multiplicação
5.	Invmul	RealInterval	Pseudo inverso multiplicativo
6.	sclmul	RealInterval	Multiplicação por um escalar
7.	div	RealInterval	Divisão
Módulo 2-Operações entre conjuntos			
8.	union	RealInterval	União
9.	hull	RealInterval	União convexa
10.	intersect	RealInterval	Interseção
Módulo 3-Operações geométricas			
11.	dist	Double	Distância
12.	diam	Double	Diâmetro
13.	midpoint	Double	Ponto médio

Tabela 6.1: Operações em JAVA-XSC.

6.3 Pacote IAMath

A biblioteca de matematica intervalar, JAVA-XSC, foi desenvolvida em pacotes (package) que é a forma em JAVA de se agrupar classes e interfaces relacionadas. O nome do pacote, IAMath é o acrônimo de Interval Arithmetic Mathematics, baseado na classe que trata os conjuntos dos números reais que é a JAVA.lang.Math. Este pacote contém três classes básicas:

Classes IAMath

1) RealInterval
2) IAMath
3) IAException

Observação : A classe IAException é uma pequena classe que utilizamos em alguns metódos quando se deseja que o programa descontinue se tiver ocorrido erro em determinado método ou alguma situação incomum.

Biblioteca JAVA-XSC			
Módulo 4-Funções elementares			
N^0	Método	Tipo do Resultado	Nome genérico
14.	abs	RealInterval	Valor absoluto
15.	sqr	RealInterval	Quadratica
16.	sqrt	RealInterval	Raiz quadrada
17.	exp	RealInterval	Exponencial
18.	exp2	RealInterval	Exponencial com base 2
19.	exp10	RealInterval	Exponencial com base 10
20.	power	RealInterval	Potência
21.	ln	RealInterval	Logaritmo natural
22.	log2	RealInterval	Logaritmo de base 2
23.	log10	RealInterval	Logaritmo de base 10

Tabela 6.2: Funções elementares em JAVA-XSC.

6.3.1 Definições em JAVA

As definições a seguir são importantes para facilitar a compreensão das classes desenvolvidas para a biblioteca JAVA-XSC.

Construtores: Ao criar uma instância de objeto, seu programa atribui normalmente valores iniciais aos membros de dados do objeto, JAVA oferece suporte a uma função especial, chamada *construtor* [CGI99], que é executada automaticamente quando é criada a instância.

Método: Um grupo de instruções em uma classe, que define como os objetos da classe se comportarão. Os métodos [LeC99] são análogos às funções em outras linguagens, mas sempre devem estar localizados dentro de um classe.

Double: Um tipo variável de número flutuante de precisão dupla (8 bytes) onde se armazenam os números reais.

String: Tipo especial de variável utilizada para armazenar texto.

6.3.2 Classe RealInterval

Esta classe representa o intervalo de números reais. Esta classe define as estruturas do intervalo e contém as seguintes variáveis, construtores e métodos:

Biblioteca JAVA-XSC			
Módulo 5-Funções trigonométricas			
N^0	Método	Tipo do Resultado	Nome genérico
24.	sin	RealInterval	Seno
25.	cos	RealInterval	Co-seno
26.	tan	RealInterval	Tangente
27.	cot	RealInterval	Co-tangente
28.	sec	RealInterval	Secante
29.	csc	RealInterval	Co-secante
30.	arcsin	RealInterval	Arco seno
31.	arccos	RealInterval	Arco co-seno
32.	arctan	RealInterval	Arco tangente
33.	sinh	RealInterval	Seno hiperbólico
34.	cosh	RealInterval	Co-seno hiperbólico
35.	tanh	RealInterval	Tangente hiperbólica
36.	coth	RealInterval	Co-tangente hiperbólica
37.	sech	RealInterval	Secante hiperbólica
38.	scsh	RealInterval	Co-secante hiperbólica
39.	senhI	RealInterval	Seno hiperbólico inverso
40.	coshI	RealInterval	Co-seno hiperbólico inverso
41.	tanhI	RealInterval	Tangente hiperbólica inversa
42.	cothI	RealInterval	Co-tangente hiperbólica inversa
43.	sechI	RealInterval	Secante hiperbólica inversa
44.	cschI	RealInterval	Co-secante hiperbólica inversa

Tabela 6.3: Funções trigonométricas em JAVA-XSC.

Biblioteca JAVA-XSC			
Módulo 6-Funções de transferência e constantes			
N^0	Método	Tipo do Resultado	Nome genérico
45.	int_interv	RealInterval	Converte inteiro $\rightarrow$ intervalo
46.	real_interv	RealInterval	Converte real $\rightarrow$ intervalo
47.	inf	Double	retorna o extremo inferior do intervalo
48.	sup	Double	retorna o extremo superior do intervalo
49.	Pi	RealInterval	Intervalo de π
50.	PI_n	RealInterval	Intervalo de $\frac{\pi}{n}$
51.	A	RealInterval	Intervalo de $\frac{1}{\pi}$
52.	blow	RealInterval	Inflacionamento

Tabela 6.4: Funções transferência e constantes em JAVA-XSC.

Variáveis:

double lo: armazena o extremo inferior do intervalo;

double hi: armazena o extremo superior do intervalo;

string Err: armazena erros que ocorrerem nos métodos desenvolvidos durante a execução da classe IAMath.

Construtores:

1. RealInterval(double, double): Constrói o intervalo real a partir de duas variáveis que contêm os extremos inferiores e superiores.
2. RealInterval(double): Constrói um intervalo degenerado (lo = hi), a partir de um unica variável double.
3. RealInterval(): Constrói um intervalo contendo como extremo inferior - infinito e + infinito, que é uma forma de criar e preencher este intervalo.

Método: Na Classe RealInterval temos método toString que faz o layout do intervalo colocando os cochetes e o ponto e virgula no intervalo para que seja impresso desta forma.

6.3.3 Classe IAMath

Esta Classe contém 52 métodos (verificar nas tabelas 6.1, 6.2, 6.3 e 6.4) para executar operações e funções intervalares baseados no módulo I_ARI da linguagem PASCAL-XSC. Em anexo, no final deste trabalho listagem da classe IAMath.

6.4 Testes

Nesta seção faremos um teste de todos os métodos desenvolvidos na biblioteca IAMath. Utilizaremos, como teste, todos os exemplos descritos nos capitulos 2 e 3 que tratam das operações e funções intervalares. O nome do programa é TesteBib. A listagem deste programa está, como anexo, no final da dissertação.

6.5 Relatório do programa TesteBib.JAVA

```
UFRN/DIMAP/LABLIC
Relatorio do teste na Biblioteca JAVA-XSC,
com todos os exemplos descritos na dissertacao nos capitulos 2 e 3.
------------------------------------------------------------
MODULO-1 OPERACOES BASICAS - TESTE - JAVA-XSC
------------------------------------------------------------
1)Soma
A=[2.0;3.0] B=[4.0;5.0]
add(A,B)= [6.0 ; 8.0]
------------------------------------------------------------
2)Pseudo aditivo
A=[2.0;3.0]
uminus(A,B)= [-3.0 ; -2.0]
------------------------------------------------------------
3)Subtracao
A=[2.0;3.0] B=[4.0;5.0]
sub(A,B)= [-3.0 ; -1.0]
------------------------------------------------------------
4)Multiplicacao
A=[-2.0;3.0] B=[-4.0;5.0]
mul(A,B)= [-12.0 ; 15.0]
------------------------------------------------------------
5)Pseudo inverso multiplicativo
A=[-3.0;-2.0]
invmul(A)= [-0.5 ; -0.3333333333333333]
------------------------------------------------------------
6)Multiplicacao por um escalar
A=[-2.0;1.0] e x = -2.0
sclmul(A)= [-2.0 ; 4.0]
------------------------------------------------------------
7)Divisao
A=[2.0;3.0] B=[4.0;5.0]
div(A,B)= [0.4 ; 0.75]
------------------------------------------------------------
MODULO-2 FUNCOES ENTRE CONJUNTOS - TESTE - JAVA-XSC
------------------------------------------------------------
8)Uniao
A=[-3.0;2.0] B=[0.0;5.0]
union(A,B)= [-3.0 ; 5.0]

C=[2;3] D=[4;5]
union(C,D)= union: A intersecao e vazia!
------------------------------------------------------------
9)Uniao convexa
A=[-3.0;2.0] B=[0.0;5.0]
hull(A,B)= [-3.0 ; 5.0]

C=[2.0;3.0] D=[4.0;5.0]
hull(C,D)= [2.0 ; 5.0]
```

```
E=[3.0;8.0] F=[-1;10.0]
hull(E,F)= [-1.0 ; 10.0]
------------------------------------------------------------
10)Intersecao
A=[2.0;5.0] B=[3.0;8.0]
intersect(A,B)= [3.0 ; 5.0]

C=[2.0;5.0] D=[6.0;8.0]
intersect(C,D)= intersect: Nao ha intersecao
------------------------------------------------------------
MODULO-3 FUNCOES GEOMETRICAS - TESTE - JAVA-XSC
------------------------------------------------------------
11)Distancia
A=[2.0;5.0] B=[-2.0;-1.0]

dist(A,A)= 0.0
dist(A,B)= 6.0
------------------------------------------------------------
12)Diametro
A=[2.0;5.0]
diam(A)= 3.0
------------------------------------------------------------
13)Ponto medio
A=[2.0;6.0]
midpoint(A)= 4.0
------------------------------------------------------------
MODULO-4 FUNCOES ELEMENTARES - TESTE - JAVA-XSC
------------------------------------------------------------
14)Valor absoluto
A=[-4.0;0.0] B=[-3.0;-1.0] C= [-2.0;3.0]
abs(A)= [0.0 ; 4.0]
abs(B)= [1.0 ; 3.0]
abs(C)= [0.0 ; 3.0]
------------------------------------------------------------
15)Quadrado
A=[-1;2]
sqr(A)= [0.0 ; 4.0]

B=[-2;-1]
sqr(B)= [1.0 ; 4.0]

C=[1;2]
sqr(C)= [1.0 ; 4.0]
------------------------------------------------------------
16)Raiz quadrada
A=[9.0;25.0]
sqrt(A)= [3.0 ; 5.0]

A=[-4.0;25.0]
sqrt(A)= sqrt: O intervalo nao pode possuir raizes imaginarias
------------------------------------------------------------
```

```
17)Exponencial
A=[0.0;1.0]
exp(A)= [1.0 ; 2.718281828459045]

A=[-1.0;1.0]
exp(A)= [0.36787944117144233 ; 2.718281828459045]
-------------------------------------------------------------
18)Exponencial com base 2
A=[-1.0;2.0]
exp(A)= [0.5 ; 4.0]
-------------------------------------------------------------
19)Exponencial com base 10
A=[-2.0;1.0]
exp(A)= [0.01 ; 10.0]
-------------------------------------------------------------
20)Potencia
A=[-2.0;3.0] n = 3
power(A,n)= [-8.0 ; 27.0]

B=[-3.0;2.0] n = 2
power(B,n)= [0.0 ; 9.0]

C=[-5.0;-3.0] n = 2
power(C,n)= [9.0 ; 25.0]

D=[2.0;4.0] n = 2
power(D,n)= [4.0 ; 16.0]
-------------------------------------------------------------
21)Logaritmo
A=[1.0;e]
ln(A)= [0.0 ; 1.0]
-------------------------------------------------------------
22)Logaritmo de base 2
A=[1.0;4.0]
log2(A)= [0.0 ; 2.0]
-------------------------------------------------------------
23)Logaritmo de base 10
A=[2.0;3.0]
log2(A)= [0.30102999566398114 ; 0.4771212547196623]
-------------------------------------------------------------
MODULO-5 FUNCOES TRIGONOMETRICAS  TESTE - JAVA-XSC
-------------------------------------------------------------
24)Seno
A=[0,1]
sin(A)= [0.0 ; 0.8414709848078965]


A=[1.0;9.0]
sin(A)= [-1.0 ; 1.0]


A=[2.0;6.0]
```

```
sin(A)= [-1.0 ; 0.9092974268256817]


A=[4.0;5.0]
sin(A)= [-1.0 ; -0.7568024953079282]
------------------------------------------------------------
25)Co-seno
A=[0.0;PI/2]
cos(A)= [0.0 ; 1.0]


A=[1;9]
cos(A)= [-1.0 ; 1.0]


A=[2.0;5.0]
cos(A)= [-1.0 ; 0.2836621854632262]


A=[4.0;5.0]
cos(A)= [-0.6536436208636119 ; 0.2836621854632262]
------------------------------------------------------------
26)Tangente
A=[0.0;1.0]
tan(A)= [0.0 ; 1.5574077246549023]


A=[1.0;9.0]
tan(A)= tan: Diverge, pois o intervalo nao pertence ao dominio da funcao


A=[2.0;4.0]
tan(A)= [-2.185039863261519 ; 1.1578212823495777]


A=[4.0;5.0]
tan(A)= tan: Diverge, pois o intervalo nao pertence ao dominio da funcao
------------------------------------------------------------
27)Co-tangente
A=[1.0;2.0]
cot(A)= [-0.45765755436028577 ; 0.6420926159343306]

B=[2.5;3.5]
cot(B)= cot: Diverge, pois o intervalo nao pertence ao dominio da funcao

C=[-0.5;0.5]
cot(C)= cot: Diverge, pois o intervalo nao pertence ao dominio da funcao
------------------------------------------------------------
28)Secante
A=[-1.0;1.5]
sec(A)= [1.0 ; 14.136832902969903]
```

```
B=[0.0;PI]
sec(B)=  Diverge, pois o intervalo nao pertence ao dominio da funcao

C=[1.7;4.5]
sec(C)= [-7.761293996050358 ; -1.0]

D=[3.5;6.0]
sec(D)=  Diverge, pois o intervalo nao pertence ao dominio da funcao

E=[1.4;4.0]
sec(E)=  Diverge, pois o intervalo nao pertence ao dominio da funcao

F=[5.5;6.0]
sec(F)= [1.0414819265951076 ; 1.4110944706229345]
------------------------------------------------------------
29)Co-secante
A=[-1.0;1.0]
csc(A)=  Diverge, pois o intervalo nao pertence ao dominio da funcao

B=[-5.0;-4.0]
csc(B)= [1.0 ; 1.3213487088109024]

C=[4.0;6.0]
csc(C)= [-3.5788995472544065 ; -1.0]
------------------------------------------------------------
30)Arco_seno
A=[-0.5;1.0]
arcsin(A)= [-0.5235987755982989 ; 1.5707963267948966]

B=[0.5;PI/2]
arcsin(B)= arcsen: Neste intervalo a funcao nao tem inversa
------------------------------------------------------------
31)Arco co-seno
A=[0.5;1.0]
arccos(A)= [0.0 ; 1.0471975511965979]

B=[-0.5;1.5]
arccos(B)= arccos: Neste intervalo a funcao nao tem inversa
------------------------------------------------------------
32)Arco-tangente
A=[-1.0;5.0]
arctan(A)= [-0.7853981633974483 ; 1.373400766945016]
------------------------------------------------------------
33)Seno hiperbolico
A=[-PI/2;PI/2]
sinh(A)= [-2.3012989023072943 ; 2.3012989023072943]
------------------------------------------------------------
34)Co-seno hiperbolico
A=[-1;PI/2]
cosh(A)= [1.0 ; 2.509178478658056]

B=[-3;-2]
```

```
cosh(B)= [3.762195691083631 ; 10.067661995777764]

C=[1;2]
cosh(C)= [1.5430806348152437 ; 3.762195691083631]
------------------------------------------------------------
35)Tangente hiperbolica
A=[-1.0;1.0]
tanh(A)= [0.7615941559557649 ; -0.7615941559557649]
------------------------------------------------------------
36)Co-tangente hiperbolica
A=[-1.0;PI]
coth(A)= coth: O intervalo nao pode conter o valor zero

B=[1;PI]
coth(B)= [1.003741873197321 ; 1.3130352854993312]

C=[-2;-1]
coth(C)= [-1.3130352854993312 ; -1.037314720727548]
------------------------------------------------------------
37)Secante hiperbolica
A=[-1.0;PI/2]
sech(A)= [0.39853681533838675 ; 1.0]

B=[-2.0;-1]
sech(B)= [0.2658022288340797 ; 0.6480542736638855]

B=[1;2]
sech(B)= [0.2658022288340797 ; 0.6480542736638855]
------------------------------------------------------------
38)Co-secante hiperbolica
A=[1.0;3.0]
csch(A)= [0.09982156966882275 ; 0.8509181282393216]

B=[-1.0;1.0]
csch(B)= csch: O intervalo nao deve conter zero
------------------------------------------------------------
39)Seno hiperbolico inverso
A=[-PI/2;PI/2]
senhI(A)= [-1.233403117511217 ; 1.233403117511217]
------------------------------------------------------------
40)Co-seno hiperbolico inverso
A=[2;3]
coshI(A)= [1.3169578969248166 ; 1.7627471740390859]

B=[-PI/2;PI/2]
coshI(B)= coshI: O intervalo [a,b] deve satisfazer a condicao a,b >= 1
------------------------------------------------------------
41)Tangente hiperbolica inversa
A=[-1/2;1/2]
tanhI(A)= [-0.5493061443340549 ; 0.5493061443340548]
------------------------------------------------------------
42)Co-tangente hiperbolica inversa
```

```
A=[-3.0;-2.0]
cothI(A)= [-0.5493061443340549 ; -0.34657359027997264]

B=[-1.0;1.0]
cothI(B)= cothI: O intervalo nao pode estar entre -1 e 1
------------------------------------------------------------
43)Secante hiperbolica inversa
A=[1/2;1.0]
sechI(A)= [0.0 ; 1.3169578969248166]

B=[-2.0;-1.0]
sechI(B)= sechI: O intervalo tem que estar entre  0 e 1

C=[10;20]
sechI(C)= sechI: O intervalo tem que estar entre  0 e 1

D=[0.3;0.5]
sechI(D)= [1.3169578969248166 ; 1.8738202425274144]
------------------------------------------------------------
44)Co-secante hiperbolica inversa
A=[1.0;PI]
cschI(A)= [0.31316588045086846 ; 0.8813735870195429]

B=[-PI/2;3PI/2]
cschI(B)= cschI: O intervalo nao deve conter zero
-----------------------------------------------------------------
MODULO-6 FUNCOES DE TRANSFERENCIA E CONSTANTES - TESTE - JAVA-XSC
-----------------------------------------------------------------
45)Inteiro para Intervalo
a=3
int_interv(a)= [3.0 ; 3.0]
------------------------------------------------------------
46)Real para Intervalo
a=3.0
real_interv(a)= [3.0 ; 3.0]
------------------------------------------------------------
47)Extremo inferior e 48)Extremo superior
A=[1.0;4.0]

inf(A) = 1.0
sup(A) = 4.0
------------------------------------------------------------
Constantes PI
49)PI 50)PI/n 51)1/PI, n=2

PI() = [3.141592653589793 ; 3.1415926535897936]
Seja n=2 PI/n= [1.5707963267948966 ; 1.5707963267948968]
1/PI= [0.31830988618379064 ; 0.3183098861837907]
------------------------------------------------------------
52)Inflacionamento
A=[3.0;4.0]
blow(A)= [2.9999999999999996 ; 4.000000000000001]
```

Capítulo 7

Algoritmos intervalar: Método de Newton

Utilizaremos e testaremos a biblioteca desenvolvida com o algoritmo clássico conhecido como Método de Newton Intervalar.

7.1 Introdução ao método de Newton intervalar

Iniciaremos esta etapa revisando a versão real do Método de Newton. Depois, mostraremos porque não se pode fazer a extensão intervalar do Método de Newton real. Finalmente, definiremos o **operador newtoniano**. Como existem diversas versões do método de Newton intervalar, apresentaramos a versão clássica e duas variações do algoritmo. Com o uso da biblioteca de operações intervalares JAVA-XSC, utilizamos para programar os algoritmos recursivos, para simulação computacional, as três versões do método de Newton intervalar, o que está registrado no final de cada versão.

O método de Newton real é um algoritmo que serve para calcular a raiz de uma dada equação, através da construção de uma seqüência convergente de pontos da reta real. De maneira análoga, a versão intervalar do Método de Newton permite construir uma seqüência convergente de intervalos, cujo limite será um intervalo que contém a raiz real da função dada.

Por ser um método autovalidável, se tomarmos um intervalo inicial que não contenha a raiz real, então, numa dada iteração, iremos obter um intervalo vazio como resultado. Caso contrário, se tomarmos um intervalo inicial, que contém a raiz real da equação $f(x) = 0$ e, considerando que a seqüencia intervalar que se obtém é de intervalos encaixados, então obteremos 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.

7.1.1 O método de Newton na versão real

Obtenção da fórmula do Operador newtoniano

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 $f'(x_*) \neq 0$. Tomemos 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 $\dfrac{y - y_P}{x - x_P} = a$ é a equação da reta que passa pelo ponto $P = (x_P, y_P)$ e que tem inclinação a.

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

Daí, $f'(x_0) \cdot (x - x_0) + f(x_0) = f'(x_0)x - f'(x_0)x_0 + f(x_0)$

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

Generalizando, temos que $x_{k+1} = x_k - \dfrac{f(x_k)}{f'(x_k)}$ é a fórmula que permite gerar a seqüência de pontos convergindo para a raiz x_* de $f(x_*) = 0$. O ponto inicial x_0 deve ser tomado suficientemente próximo da raiz x_* para que o método convirja para a raiz desejada. Em geral, costuma-se tomar $x_0 = \dfrac{a+b}{2}$ ou, ainda, $x_0 \in [a;b]$, tal que $f(x_0) \cdot f"(x_0) > 0$.

Definição de Operador newtoniano real

Chamamos de operador **newtoniano** à expressão $N(x) = x - \dfrac{f(x)}{f'(x)}$.

Da definição acima, temos o seguinte algoritmo, o qual chamaremos de método de Newton real:

Teorema 1 (Método de Newton real)

Seja $f(x)$ uma função real contínua com derivada $f'(x)$ contínua num intervalo $[a;b]$, que contém a raiz real x_* de $f(x) = 0$. Tomando $x_0 \in [a;b]$ tal que $f(x_0) \cdot f"(x_0) > 0$ e, definindo a seqüência recursiva $x_{k+1} = N(x_k)$, então a seqüência x_k é convergente e $lim_{k\to\infty} x_k = x_*$.

k	X_k
0	$2/1 = 2$
1	$3/2 = 1.5$
2	$17/12 = 1.416666...$
3	$577/408 = 1.414215686275...$
4	$665857/470832 = 1.41421356237...$
$\vdots$	$\vdots$
$lim_{k\to\infty} x_k$	$\sqrt{2}$

Tabela 7.1: Seqüência X_k pelo método de Newton real.

Exemplo 1: Calcular, pelo método de Newton real, o valor de $\sqrt{2}$.

Solução:

$x = \sqrt{2} \therefore x^2 = 2 \therefore x^2 - 2 = 0.$

Daí, $f(x) = x^2 - 2, f'(x) = 2x$ e $f''(x) = 2$.

Como $1 \leqslant 2 \leqslant 4 \therefore \sqrt{1} \leqslant \sqrt{2} \leqslant \sqrt{4} \therefore 1 \leqslant \sqrt{2} \leqslant 2.$

Então, a raiz real pertence ao intervalo [1;2].

Tomamos $x_0 = 2 \in [1;2]$, pois $f(2) \cdot f''(2) = (2^2 - 2) \cdot 2 = 2 \cdot 2 = 4 > 0$ e definimos a seqüência

$$\begin{aligned} x_{k+1} &= N(x_k) \\ &= x_k - \frac{f(x_k)}{f'(x_k)} \\ &= x_k - \frac{(x_K^2 - 2)}{2x_k} \\ &= \frac{(x_k^2 + 2)}{2x_k}. \end{aligned}$$

Assim, $\sqrt{2} = lim_{k\to\infty} x_k \cong 1.41421356237....$

7.1.2 O método de Newton intervalar

Nesta subseção será definida uma versão intervalar para o método de Newton real.

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 $f'(x)$.

A seguir, veremos porque o método de Newton construído dessa maneira é sempre divergente.

Por exemplo, para calcularmos a raiz $\sqrt{2}$ da função $f(x) = x^2 - 2 = 0$ no intervalo $[1; 2]$, temos $F(X) = X^2 - [2; 2]$ e $F'(X) = [2; 2].X$. Tomando $X_0 = [1; 2]$, calculamos

$$\begin{aligned} X_1 &= X_0 - \frac{F(X_0)}{F'(X_0)} \\ &= [1;2] - \frac{[1;2]^2 - [2;2]}{[2;2].[1;2]} \\ &= [1;2] - \frac{[1;4] - [2;2]}{[2;4]} \\ &= [1;2] - \frac{[-1;2]}{[2;4]} \\ &= [1;2] - [-1;2].\left[\frac{1}{4};\frac{1}{2}\right] \\ &= [1;2] + \left[-1; -\frac{1}{2}; 1\right] \\ &= [1;2] + \left[-1; \frac{1}{2}\right] \\ &= \left[0; \frac{5}{2}\right]. \end{aligned}$$

Observe que

$$diam(X_1) = diam\left([0; \frac{5}{2}]\right) = \frac{5}{2} > 1 = diam(X_0)$$

e que $X_0 \subseteq X_1$. Observe também que

$$F'(X_1) = [2; 2] \cdot X_1 = [2; 2] \cdot \left[0; \frac{5}{2}\right] = [0; 5]$$

contém zero, logo na próxima iteração (cálculo de X_2), não poderemos efetuar a operação de divisão.

Como queremos que a seqüência de intervalos X_k convirja para o intervalo pontual $[x_*; x_*]$, é necessário que $diam(X_k) \to 0$, ao $k \to +\infty$. Mas como

$$\begin{aligned} diam(X_{k+1}) &= diam\left(X_k - \frac{F(X_k)}{F'(X_k)}\right) \\ &= diam(X_k) + diam\left(\frac{F(X_k)}{F'(X_k)}\right) > diam(X_k), \end{aligned}$$

conclui-se que a seqüência de intervalos definida por $X_{k+1} = X_k - \dfrac{F(X_k)}{F'(X_k)}$ é sempre divergente.

Assim, devemos modificar o método, de maneira que o diâmetro de um intervalo recém calculado X_{k+1} diminua em relação ao diâmetro de X_k e que ainda contenha x_*. A idéia agora é construirmos uma seqüência de intervalos encaixados $X_0 \supseteq X_1 \supseteq X_2 \supseteq X_3 \supseteq X_4 \supseteq ...X_k \supseteq ...[x_*; x_*]$, cujos diâmetros vão diminuindo a medida que o valor de k aumenta. Para tanto, vamos avaliar com intervalos somente a derivada, cuidando para que o intervalo resultante da avaliação intervalar da derivada não contenha o zero. Desta forma, precisamos exigir que a função f satisfaça certas condições no intervalo inicial X_0, ou seja, para que a avaliação intervalar da derivada não contenha o zero, é necessário que a função f não tenha pontos críticos (máximos ou mínimos locais) em X_0. Veremos, a seguir, como definir o método de Newton Intervalar que é sempre convergente.

Teorema 2 (Método de Newton intervalar)

Seja $f(x)$ uma função real contínua em $X_0 = [a; b]$, de modo que $f(a) \cdot f(b) < 0$. Assim, existe $x_* \in X_0$ tal que $f(x_*) = 0$. Calculamos um intervalo $M = [m_1; m_2]$, tal que $0 < m_1 \leqslant \dfrac{f(x) - f(y)}{x - y} \leqslant m_2 \leqslant +\infty$, para todo $x, y \in X_0$, com $x \neq y$, que servirá como uma inclusão intervalar para a imagem intervalar da derivada $f'(x)$ da função $f(x)$. Finalmente, definindo o **operador newtoniano intervalar** por $N(X) = med(X) - \dfrac{f(med(X))}{M}$, construímos a seqüência intervalar recursiva $X_{k+1} = X_k \cap N(x_k)$ e $X_0 = [a, b]$, que tem as seguintes propriedades

1. $x_* \in X_k, \forall k \geqslant 0$;

2. $X_0 \supseteq X_1 \supseteq X_2 \supseteq X_3 \supseteq ...$ e $lim_{k \to \infty} X_k = [x_*; x_*]$;

3. $diam(X_{k+1}) \leqslant \left(1 - \dfrac{m_1}{m_2}\right) \cdot diam(X_k) \therefore diam(X_{k+1}) \leqslant \left(1 - \dfrac{m_1}{m_2}\right)^k \cdot diam(X_0)$, o que implica que $lim_{k \to \infty}\ diam(X_k) = 0$;

4. Se existe $k_0 \geqslant 0$ tal que $X_{k_0} \cap N(X_{k_0}) = \emptyset$, então não existe raiz da função f no intervalo inicial X_0.

A condição 4, acima, garante a autovalidação do algoritmo, ou seja, se escolhermos um intervalo inicial que não contém a raiz da função, então o método pára, caso contrário, ele calcula o intervalo limite da seqüência X_k, que contém x_*, que é a raiz da função f, validando a resposta.

Observação :

1. $x_* \in x - \dfrac{f(x)}{M}, \forall x \in X_0$;

2. Podemos tomar $M_1 = \mathcal{I}\left(\dfrac{f(x)}{x - x_*}, X_0\right) \subseteq M$ ou ainda $M_2 = [m_1; m_2] = \mathcal{I}(f'(x), X_0) \subseteq M$.

Exemplo 2: Calcular a raiz da função $f(x) = x^2 - 2 = 0$ em $X_0 = [1; 2]$.

Solução : $f(x) = x^2 - 2$ é uma função contínua em $\mathbb{R}$, logo é contínua no intervalo inicial $X_0 = [1; 2]$.

Temos que

$$\begin{aligned} f(1) \cdot f(2) &= (1^2 - 2) \cdot (2^2 - 2) \\ &= (-1) \cdot (2) \\ &= -2 < 0, \end{aligned}$$

logo a função tem pelo menos uma raiz no intervalo $[1; 2]$.

O cálculo da inclusão intervalar M para a derivada pode ser feito por

$$\begin{aligned} \frac{f(x) - f(y)}{x - y} &= \frac{(x^2 - 2) - (y^2 - 2)}{x - y} \\ &= \frac{x^2 - y^2}{x - y} \\ &= \frac{(x - y).(x + y)}{x - y} \\ &= x + y \end{aligned}$$

e, daí, temos que $0 < 2 \leqslant \dfrac{f(x) - f(y)}{x - y} = x + y \leqslant 4$, para quaisquer $x, y \in X_0 = [1; 2]$, pois $1 \leqslant x \leqslant 2$ e $1 \leqslant y \leqslant 2$ logo, $1 + 1 \leqslant x + y \leqslant 2 + 2$.

Como o cálculo da derivada $f'(x)$ da função $f(x)$ é neste caso, mais fácil, podemos tomar

$$\begin{aligned} M &= [m_1; m_2] \\ &= \mathcal{I}(f'(x), X_0) \\ &= \mathcal{I}(2x, [1; 2]) \\ &= [min\{2x / x \in [1; 2]\}; max\{2x / x \in [1; 2]\}] \\ &= [2; 4]. \end{aligned}$$

k	X_k
0	$[1; 2]$
1	$[1.3; 1.5]$
2	$[1.41; 1.42]$
3	$[1.414; 1.415]$
4	$[1.4142; 1.4143]$
5	$[1.414212; 1.414215]$
6	$[1.4142135; 1.4142137]$
7	$[1.41421355; 1.41421357]$
8	$[1.414213562; 1.414213563]$
9	$[1.4142135623; 1.4142135624]$
10	$[1.414213562372; 1.414213562374]$
11	$[1.4142135623730; 1.4142135623732]$
12	$[1.41421356237309; 1.41421356237310]$
13	$[1.414213562373094; 1.4142135623730096]$

Tabela 7.2: Seqüência X_k do método de Newton intervalar.

Assim, percebe-se que as duas inclusões que foram calculadas para a derivada intervalar coincidem.

O operador intervalar newtoniano fica, então:

$$N(X) = [med(X); med(X)] - \frac{[(med(X))^2 - 2; (med(X))^2 - 2]}{[2; 4]}.$$

Finalmente, construímos a seqüência intervalar recursiva

$$X_{k+1} = X_k \cap N(X_k) \text{ e } X_0 = [1; 2].$$

Computando os valores da seqüência, temos:

Logo, o valor da raiz da função $f(x) = x^2 - 2$ é um valor real $x_* \in [1.414213562373094; 1.414213562373096]$.

7.1.3 Teste utilizando JAVA-XSC com o método de Newton

Este teste tem por objetivo principal verificar se as operações intervalares desenvolvidas em JAVA estão corretas. Utilizaremos o método Newton intervalar que constrói uma seqüência intervalar recursiva.

Etapas do desenvolvimento do programa Newton.JAVA:

1. Criar o programa Newton com o formato .java.

2. Definir a derivada da função que é o intervalo $[2; 4]$ que está no denominador do operador intervalar sendo representado pela variável DERIV. Inicializar o intervalo inicial X_0 que, no programa, é representado pela variavel X, que conterá os valores da seqüência intervalar recursiva até que fique com os valores exigidos.

3. Desenvolver o operador intervalar newtoniano N(X) em JAVA-XSC:

$$N(X) = [med(X); med(X)] - \frac{[(med(X))^2 - 2; (med(X))^2 - 2]}{[2; 4]}.$$

   Foram utilizados os seguintes métodos desenvolvidos em JAVA- XSC:

   IAMath.sub - subtração de intervalos
   IAMath.midpoint - ponto médio do intervalo
   IAMath.div - divisão de intervalos

   X=IAMath.sub((new RealInterval(IAMath.midpoint(X),IAMath.midpoint(X))),
   IAMath.div(new RealInterval((Math.pow(IAMath.midpoint(X),2)2),
   (Math.pow(IAMath.midpoint(X),2)2)),DERIV));

4. Utilizamos o comando de controle *while* do JAVA(abaixo), para que o programa fizesse as iterações enquanto o extremo inferior fosse diferente do extremo superior e também que o extremo inferior da última iteração fosse diferente da atual iteração, de forma que os valores resultantes convirjam para o resultado final.

   Enquanto(while) ((X.lo diferente X.hi) e (ANTX.lo diferente X.lo)).

Conclusão do teste

Conforme anexo da listagem do programa e do teste, o programa Newton teve ótimo desempenho, tendo o resultado do programa Newton convergindo para o intervalo $X = [1.4142135622373095; 1.4142135623730951]$, conforme o esperado, e tendo sido aplicados os métodos intervalares da biblioteca JAVA-XSC.

k	X_k
0	$[1.0; 2.0]$
1	$[1.375; 1.4375]$
2	$[1.411865234375; 1.41748046875]$
3	$[1.4140232130885124; 1.4143480323255062]$
4	$[1.4142053788392064; 1.4142251349714035]$
5	$[1.414212860473436; 1.4142140586893706]$
6	$[1.414213532266103; 1.4142136049508025]$
7	$[1.4142135597903254; 1.4142135641993892]$
8	$[1.4142135622623118; 1.4142135625297663]$
9	$[1.4142135623635914; 1.4142135623798153]$
10	$[1.4142135623726875; 1.4142135623736714]$
11	$[1.41421356237306; 1.4142135623731198]$
12	$[1.41421356223730936; 1.4142135623730971]$
13	$[1.4142135622373095; 1.4142135623730951]$
14	$[1.4142135622373095; 1.4142135623730951]$

Tabela 7.3: Seqüência X_k gerados pelo programa Newton em JAVA-XSC.

O programa Newton

```
import ia_math.*;
public class  Newton {
       public static void main (String arguments[ ]) {
          RealInterval DERIV = new RealInterval(2.0,4.0);
          RealInterval X = new RealInterval(1.0,2.0);
          RealInterval ANTX = new RealInterval(0.0,0.0);
          int k = 0;
          System.out.println("___________________________________________________");
          System.out.println("| k |                         X                     |");
          System.out.println("| "+k+" | "+X+" |");
          while((X.lo != X.hi) && (ANTX.lo != X.lo))
          {
          ANTX=X;
          X=IAMath.sub((new RealInterval(IAMath.midpoint(X),IAMath.midpoint(X))),
IAMath.div(new RealInterval((Math.pow(IAMath.midpoint(X),2)-2),
(Math.pow(IAMath.midpoint(X),2)-2)),DERIV));
          k=k+1;
          System.out.println("| "+k+" | "+X+" |");
            }
          System.out.println("---------------------------------------------------");
         }
}
```

7.1.4 Variações do método de Newton intervalar

Nesta seção, vamos citar apenas duas das possíveis variações para o Método de Newton Intervalar. Vale dizer que estas variações permitem que o algoritmo convirja com uma velocidade maior, ou seja, com número menor de iterações.

7.1.5 Variação 1 do método de Newton

Consiste em tomar a inclusão intervalar da derivada sendo atualizada em cada iteração, ou seja, definimos $M_k = F'(X_k)$, onde $F'(X)$ é uma avaliação intervalar para a derivada $f'(x)$ da função $f(x)$.

Teorema 3 (Variação 1 para o método de Newton intervalar)

Seja $f(x)$ uma função real contínua em $X_0 = [a; b]$, de modo que $f(a) \cdot f(b) < 0$. Assim, existe $x_* \in X_0$ tal que $f(x_*) = 0$. Seja $F'(X)$ uma avaliação intervalar da derivada $f'(x)$ da função $f(x)$. Finalmente, definindo o operador newtoniano intervalar por $N_1(X) = med(X) - \dfrac{f(med(X))}{F'(X)}$, construímos a seqüência intervalar recursiva $X_{k+1} = X_k \cap N_1(X_k)$ e $X_0 = [a; b]$.

Exemplo 3: Calcular a raiz da função $f(x) = x_2 - 2 = 0$ em $X_0 = [1; 2]$, usando este novo operador newtoniano intervalar.

Solução: Como a derivada da função $f(x)$ é $f'(x) = 2x$, tomamos a avaliação intervalar $F'(X) = [2; 2] \cdot X$. Daí, o operador intervalar newtoniano fica:

$$N_1(X) = [med(X); med(X)] - \frac{[(med(X))^2 - 2; (med(X))^2 - 2]}{[2; 2] \cdot X}.$$

Finalmente, construímos a seqüência intervalar recursiva

$X_{k+1} = X_k \cap N_1(X_k)$ e $X_0 = [1; 2]$.

Computando os valores da seqüência, temos:

Logo, o valor da raiz da função $f(x) = x^2 - 2$ é um valor real

$x_* \in [1.414213562373094; 1.414213562373096]$.

Note-se que o intervalo limite é o mesmo, porém se percebe que o número de iterações é menor do que quando usamos a primeira versão do método de Newton intervalar.

k	X_k
0	$[1;2]$
1	$[1.3;1.5]$
2	$[1.414;1.415]$
3	$[1.41421355;1.41421357]$
4	$[1.414213562373094;1.4142135623730096]$
5	$[1.414213562373094;1.4142135623730096]$
6	$[1.414213562373094;1.4142135623730096]$
7	$[1.414213562373094;1.4142135623730096]$
8	$[1.414213562373094;1.4142135623730096]$
9	$[1.414213562373094;1.4142135623730096]$
10	$[1.414213562373094;1.4142135623730096]$

Tabela 7.4: Seqüência $X_{k+1} = X_k \cap N_1(X_k)$.

7.1.6 Teste em JAVA-XSC utilizando a variação 1 de Newton

Na variação 1 do método de Newton como vimos acima, este novo operador newtoniano intervalar teve as seguintes modificações: o valor da derivada, que é o dividendo, passou a ter o valor intervalar DERIV = [2;2] e a multiplicação por uma avaliação intervalar no caso X. Estão em negrito as alterações que teve o operador newtoniano. Fizemos estas alterações também no programa, tendo utilizado mais um método da biblioteca JAVA-XSC, no caso a multiplicação (IAMATh.mul), conforme programa Newton1 e teste apresentados a seguir.

$$N_1(X) = [med(X); med(X)] - \frac{[(med(X))^2 - 2; (med(X))^2 - 2]}{\mathbf{[2;2].X}}.$$

Operador newtoniano em JAVA-XSC

X=IAMath.sub((new RealInterval(IAMath.midpoint(X),IAMath.midpoint(X))), IAMath.div(new RealInterval((Math.pow(IAMath.midpoint(X),2)-2), (Math.pow(IAMath.midpoint(X),2)-2)),**IAMath.mul(DERIV,X)**));

O programa Newton1

```
import ia_math.*;
public class  Newton1 {
       public static void main (String arguments[ ]) {
          RealInterval DERIV = new RealInterval(2.0,2.0);
          RealInterval X = new RealInterval(1.0,2.0);
```

k	X_k
0	$[1.0; 2.0]$
1	$[1.375; 1.4375]$
2	$[1.4140625; 1.4144176136363635]$
3	$[1.4142135592945242; 1.4142135659471788]$
4	$[1.4142135623730951; 1.4142135623730951]$

Tabela 7.5: Seqüência X_k gerados pelo programa Newton1 em JAVA-XSC.

```
          RealInterval ANTX = new RealInterval(0.0,0.0);
          int k = 0;
          System.out.println("_______________________________________________");
          System.out.println("| k |                       X                    |");
          System.out.println("| "+k+" | "+X+" |");
          while((X.lo != X.hi) && (ANTX.lo != X.lo))
          {
          ANTX=X;
          X=IAMath.sub((new RealInterval(IAMath.midpoint(X),IAMath.midpoint(X))),
IAMath.div(new RealInterval((Math.pow(IAMath.midpoint(X),2)-2),
(Math.pow(IAMath.midpoint(X),2)-2)),IAMath.mul(DERIV,X)));
          k=k+1;
          System.out.println("| "+k+" | "+X+" |");
            }
          System.out.println("-----------------------------------------------");
         }
}
```

7.1.7 Variação 2 do método de Newton

Consiste em tomar a inclusão intervalar da derivada, sendo atualizada em cada iteração (como na variação 1) e, também, em se atualizar o ponto onde a função real é calculada, ou seja, definimos $M_k = F'(X_k)$, onde $F'(X)$ é uma avaliação intervalar para a derivada $f'(x)$ da função $f(x)$ e, ao invés do ponto médio do intervalo X_k usamos

$$\begin{aligned} x_{k+1} &= N(x_k) \\ &= x_k - \frac{f(x_k)}{f'(x_k)} \quad \text{e } x_0 = med(X_0). \end{aligned}$$

Teorema 4 (Variação 2 para o método de Newton intervalar)

Seja $f(x)$ uma função real contínua em $X_0 = [a; b]$, de modo que $f(a)\cdot f(b) < 0$. Assim, existe $x_* \in X_0$, tal que $f(x_*) = 0$. Seja $F'(X)$ uma avaliação intervalar da derivada $f'(x)$ da função $f(x)$. Finalmente, definimos o **operador newtoniano intervalar** em duas partes:

$$X_0 = [a; b] \text{ e } x_0 = med(X_0) = \frac{a+b}{2}.$$

1. $n(x) = x - \dfrac{f(x)}{f'(x)} \in \mathbb{R}$, que corresponde a tomar uma nova aproximação para a raiz, usando o método de Newton real;

2. $N_2(X) = [n(x); n(x)] - \dfrac{[f(n(x))]; f(n(x))]}{F'(X)}$.

Finalmente, construímos a seqüência intervalar recursiva

$X_{k+1} = X_k \cap N_2(X_k)$ e $X_0 = [a; b]$.

Exemplo 4: Calcular a raiz da função $f(x) = x^2 - 2 = 0$ em $X_0 = [1; 2]$, usando este segundo operador newtoniano intervalar.

Solução : Como a derivada da função $f(x)$ é $f'(x) = 2x$, tomamos a avaliação intervalar $F'(X) = [2; 2] \cdot X$. Daí, o operador intervalar newtoniano fica:

1. $n(x) = x - \dfrac{x^2 - 2}{2x} \in \mathbb{R}$;

2. $N_2(X) = [n(x); n(x)] - \dfrac{[n(x)^2 - 2; n(x)^2 - 2]}{[2; 2] \cdot X}$

Finalmente, construímos a seqüência intervalar recursiva.

$X_{k+1} = X_k \cap N_2(X_k), X_0 = [1; 2]$ e $x_0 = \dfrac{3}{2}$.

k	X_k
0	$[1; 2]$
1	$[1.3; 1.5]$
2	$[1.414; 1.415]$
3	$[1.41421355; 1.41421357]$
4	$[1.414213562373094; 1.4142135623730096]$
5	$[1.414213562373094; 1.4142135623730096]$

Tabela 7.6: Seqüência $X_{k+1} = X_k \cap N_2(X_k)$.

k	X_k
0	$[1.0; 2.0]$
1	$[1.375; 1.4375]$
2	$[1.4141414141414141; 1.4142512077294687]$
3	$[1.4142135622631407; 1.4142135624280354]$
4	$[1.4142135623730951; 1.4142135623730951]$

Tabela 7.7: Seqüência X_k gerados pelo programa Newton2 em JAVA-XSC.

Computando os valores da seqüência, temos:

Logo, o valor da raiz da função $f(x) = x^2 - 2$ é um valor real $x_* \in [1.414213562373024; 1.414213562373096]$.

7.1.8 Teste em JAVA-XSC utilizando a variação 2 do método de Newton

Na variação 2 do método do Newton utiliza o operador newtoniano dividido em duas partes, sendo que a primeira parte não contém operações intervalares, e com algumas alterações na segunda parte, mas sem modificações substanciais. Baseado no programa Newton1 criamos um novo programa Newton2 com estas novas alterações, conforme programa e teste mostrados a seguir.

O programa Newton2

```
import ia_math.*;
public class  Newton2 {
          public static void main (String arguments[ ]) {
          RealInterval DERIV = new RealInterval(2.0,2.0);
          RealInterval X = new RealInterval(1.0,2.0);
          RealInterval ANTX = new RealInterval(0.0,0.0);
          double x;
          x=3/2;
          int k = 0;
          System.out.println("_________________________________________________");
          System.out.println("| k |                     X                      |");
          System.out.println("| "+k+" | "+X+" |");
          while((X.lo != X.hi) && (ANTX.lo != X.lo))
          {
          ANTX=X;
          x=x-(Math.pow(x,2)-2)/(2*x);
          X=IAMath.sub((new RealInterval(x,x)),
IAMath.div(new RealInterval((Math.pow(x,2)-2),
(Math.pow(x,2)-2)),IAMath.mul(DERIV,X)));
          k=k+1;
          System.out.println("| "+k+" | "+X+" |");
            }
          System.out.println("---------------------------------------------------");
         }
}
```

Capítulo 8

Conclusão

O uso da linguagem JAVA-XSC poderá trazer diversos benefícios aos usuários da computação cientifíca, como empresas e centros de pesquisas. Nos tempos atuais, devido à globalização da produção e da pesquisa científica, muitas empresas transnacionais, também conhecidas como multinacionais, têm seus centros de pesquisas situados em diversas áreas do planeta, sempre na busca de diminuírem os custos de pesquisa, desenvolvimento e produção de manufaturados e maximizarem seus resultados.

Estas empresas procuram especializar determinadas equipes de profissionais em áreas específicas. Para facilitar a compreensão, poderíamos citar o exemplo de uma indústria que deseja desenvolver um protótipo de avião e tem dois laboratórios de pesquisa e desenvolvimento de projetos: um nos EUA e outro na Alemanha. O laboratório dos EUA ficaria encarregado de projetar os modelos de aerodinâmica e segurança pelo fato de lá existirem especialistas nestas áreas, e o Laboratório na Alemanha faria o restante dos modelos. Imaginemos também que estes profissionais trabalhem em ambientes computacionais com diversas plataformas como Unix, Mac, Windows, NT, AIX, OS/2, com diferentes estações de trabalhos, fabricantes e sistemas operacionais.

As três perguntas básicas que ficariam para nós da computação científica:

1. Como unir estes diversos modelos de forma a terem um perfeito entrosamento em único projeto?

2. Como desenvolver e distribuir aplicações que executam em diferentes sistemas operacionais, versões e plataformas?

3. Como controlar e limitar os erros efetuados durante a execução do projeto?

A resposta seria o uso da linguagem JAVA com a extensão científica XSC para o desenvolvimento deste projeto. Como sabemos, os programas em JAVA compilam e rodam em qualquer plataforma, pois o programador escreve sua aplicação para uma máquina virtual, uma vez que qualquer aplicação escrita em JAVA é independente de plataforma. Somente a máquina virtual precisa ser transportada. Ela é a interface entre o computador do usuário e a aplicação JAVA, além de trabalhar sob a rede (internet), que é o veículo de distribuição das aplicações e de simulações interativas. Com a linguagem JAVA, o pesquisador do laboratório pode pedir aplicações através da internet ou intranet e processá-las numa máquina local. O emissor não precisa conhecer o seu ambiente hardware/software porque a aplicação é processada indistintamente em qualquer plataforma.

Ao mesmo tempo, a linguagem JAVA pode assegurar que as características e funções estão disponíveis para todos os utilizadores do projeto em desenvolvimento pela empresa. E as informações do projeto existentes na empresa podem ser rapidamente redirecionadas e implementadas na Web e na intranet, as aplicações existentes dos projetos dos diversos laboratórios podem ser facilmente redirecionadas para o Webtop (página da Web comum da empresa). O acesso aos aplicacionais faz-se através de um ambiente comum. E o que ainda é mais significativo é que nada do que existe nos correntes ambientes computacionais da empresa, no caso dos dois laboratórios do exemplo da indústria fabricante de aviões precisa ser eliminado para que todos estes benefícios comecem a se tornar visíveis. As aplicações JAVA podem ser introduzidas nas plataformas de computação já instaladas, podendo ser implementada uma migração faseada para as plataformas mais econômicas e, desta forma, diminuídos os custos.

À medida que os laboratórios de pesquisas e desenvolvimento podem moldar as suas próprias páginas Web, de acordo com as suas necessidades específicas, grandes conjuntos de ferramentas de desenvolvimento aplicacional podem facilitar um melhoramento das aplicações que são descentralizadas. A World Wide Web permite à empresa que, de uma forma seletiva, forneça acesso à informação na Internet e possibilite a troca de informações dos pesquisadores envolvidos no projeto. A inclusão nesta linguagem JAVA da biblioteca, com extensão científica conhecida como JAVA-XSC, irá suprir as deficiências computacionais relativas à qualidade dos resultados neste projeto. Esta extensão faz uso da matemática intervalar que é aplicada para representar valores desconhecidos e também valores contínuos; controlar o erro de arredondamento e apresentar dados inexatos, aproximações e erros de truncamento dos procedimentos, além da utilização de algoritmos numéricos autovalidáveis e com controle automático de erro. O desenvolvimento da linguagem JAVA-XSC poderá fornecer a solução aos três questionamentos feitos no exemplo do projeto da indústria de aviões.

A universalidade desta linguagem JAVA, única no seu gênero, foi confirmada pelo grande número de acordos de licenças, concluídos entre numerosas empresas de informática e a Sun Microsystems. Em perfeita simbiose com a filosofia da Sun, que preconiza uma tecnologia de rede aberta, o acordo de licenças JAVA permite às diversas empresas escolherem livremente as suas plataformas, materiais e logísticas. E com incremento desta extensão poderá tornar esta linguagem imbatível para desenvolvimento de aplicações pelos motivos já apresentados.

Concluímos haver uma necessidade imperiosa de estender JAVA para computações científicas pelos motivos acima apresentados. Neste sentido esta dissertação introduz uma biblioteca JAVA para computações científicas, como etapa fundamental da nossa proposta de construção da linguagem JAVA-XSC. Acreditamos que além da criação da biblioteca intervalar em JAVA esta dissertação teve como mérito o desenvolvimento da caracterização matemática das operações e funções intervalares em termos dos extremos de seus argumentos e que serviram de base na contrução dos métodos e implementação da biblioteca, conforme descrito no capítulo 5.

8.1 Trabalhos futuros

O processo de desenvolvimento de uma extensão científica é longo pois é necessario criar diversos módulos e tipos, operações e funções não existentes na linguagem de origem e que suportem a maioria das aplicações científicas. Nas diversas linguagens com extensões científicas, especialmente as estudadas, como PASCAL-XSC, INTLIB, C-XSC, verifica-se que todas estas são divididas em diversos módulos e contêm novos tipos de variáveis para a resolução dos diversos problemas de que trata a computação científica. O PASCAL-XSC tem sete módulos: o básico que trata da aritmética intervalar e outros módulos para as diversas formas de tipos da aritmética. A INTLIB utiliza várias rotinas entre as quais: rotinas aritméticas elementares, rotinas de funções padrão, rotinas de funções úteis, rotinas de impressões de erro, rotinas de constantes matemáticas de programas que são utilizadas para inicializar constantes globais e parâmetros. O C-XSC contém diversos tipos predefinidos para tratar as diversas rotinas aritméticas.

Nesta dissertação, desenvolvemos um novo tipo de dado em uma classe JAVA, definido como o nome de RealInterval, que armazena intervalos. Desenvolvemos um módulo que contém as operações e funções básicas da matemática intervalar baseada no I_ARI, da linguagem PASCAL-XSC. Estas operações e funções , que servem de base para a linguagem JAVA-XSC, estão descritas na dissertação nos Capítulo 2 e 3.

Definimos em matemática intervalar as funções arco co-tangente, arco secante, arco co-secante, mas estas não constam na biblioteca JAVA para números reais e conseqüentemente não foi possível desenvolver para a extensão JAVA-XSC. Como sugestão de pesquisa o desenvolvimento destas funções para a linguagem JAVA e em JAVA-XSC.

Como etapas futuras, existe a necessidade de se elaborarem módulos que venham a trabalhar com matrizes, vetores intervalares e aritmética intervalar complexa.

Uma outra sugestão de pesquisa seria a criação de applets, ou seja, programas especiais em JAVA que processam em página Web, voltados para a matemática intervalar utilizando a biblioteca que foi desenvolvida, de forma que o pesquisador ou aluno possa acessar, via Web, estas extensões da computação científica, em JAVA.

Uma última proposta ainda seria um estudo comparativo com outras linguagens XSC, procurando-se estabelecer uma comparação em termos da qualidade dos resultados (exatidão) e da diversidade das funções consideradas.

Anexos

Classe RealInterval

```
package ia_math;

/**
* A classe RealInterval representa um intervalo fechado de numeros reais
*/
public class RealInterval implements Cloneable{

        public double lo,hi;
        public String Err;

    public RealInterval(double lo, double hi)
{
if (lo <= hi)
{
this.lo = lo; this.hi = hi;
            this.Err="";
}
else
{

            System.out.println("Interval(Inferior="+lo+",Superior="+hi+"):
                                incorreto o extremo Inferior <= Superior");

}
}

    public RealInterval(double x)
{
if ((Double.NEGATIVE_INFINITY<x) && (x < Double.POSITIVE_INFINITY))
{
this.lo = x; this.hi = x;
            this.Err="";
}
else
{
```

```
            System.out.println("Intervalo(x): deve ser -inf<x<inf");
}

}

/**
* construct the interval [-infty,infty]
*/

public RealInterval()
{
this.lo = java.lang.Double.NEGATIVE_INFINITY;
this.hi = java.lang.Double.POSITIVE_INFINITY;
        this.Err= "";
}

public double lo()
{
return this.lo;
}

public double hi()
{
return this.hi;
}


public String toString() {

            String temp = new String();
            temp = "[" + this.lo + " ; " + this.hi +"]";
            return temp;
}

private String doubleToString(double x)
{
StringBuffer s = new StringBuffer((new Double(x)).toString());
int i = s.length();
int j;
for(j=1;j<20-i;i++) s.append(' ');
return(s.toString());
}

public Object clone()
{
return new
RealInterval(this.lo,this.hi);
}


}
```

Classe IAException

```
/*

IAException.java

package ia_math;

public class IAException extends RuntimeException
 {
  private String e;

  public IAException()
  {
    super();
  }

  public IAException(String s)
  {
super(s);
e = s;

  }

  public String toString()
  {
  return e;
  }

}
```

Classe IAMath

```
package ia_math;

public class IAMath
{
/*.................................................................\
.                                                                 .
.   BIBLIOTECA JAVA-XSC - CLASSE IA_MATH - METODOS INTERVALARES   .
.   -------------------------------------------------------       .
.   (BASEADO NAS DESCRICOES DE OPERACOES E FUNCOES INTERVALARES)  .
.              - CAPITULO 2 E 3 DA DISSERTACAO -                  .
\................................................................*\
```

```
/*..................................................................\
.                                                                   .
.            MODULO 1- Operacoes basicas                            .
.            --------------------------                             .
.                                                                   .
\..................................................................*\

/*******************************************************************\
*                                                                   *
*             1) Metodo add                                         *
*                Calcula a soma de Intervalos                       *
*                                                                   *
\*******************************************************************/

        public static RealInterval add(RealInterval x, RealInterval y)
        {
                RealInterval z = new RealInterval();
                z.lo = (x.lo+y.lo);
                z.hi = (x.hi+y.hi);
                return(z);
        }
/*******************************************************************\
*                                                                   *
*             2) Metodo uminus                                      *
*                Calcula o intervalo reciproco ou                   *
*                Pseudo inverso aditivo intervalar                  *
\*******************************************************************/

         public static RealInterval uminus(RealInterval x)
       {
                RealInterval z = new RealInterval();
                z.lo = -x.hi;
                z.hi = -x.lo;
                return(z);
        }
/*******************************************************************\
*                                                                   *
*             3) Metodo sub                                         *
*                Calcula a subtracao de Intervalos                  *
*                                                                   *
\*******************************************************************/

         public static RealInterval sub(RealInterval x, RealInterval y) {
                RealInterval z = new RealInterval();
                z.lo = (x.lo-y.hi);
                z.hi = (x.hi-y.lo);
                return(z);
        }
```

```
/*********************************************************************\
*                                                                     *
*             4) Metodo mul                                           *
*                Calcula a multiplicacao de intervalos                *
*                                                                     *
\*********************************************************************/
       public static RealInterval mul(RealInterval x, RealInterval y)
        {
               RealInterval z = new RealInterval();
               if (Math.min((x.lo*y.lo),(x.lo*y.hi)) >
                   Math.min((x.hi*y.lo),(x.hi*y.hi)))
                   {
                   z.lo=Math.min((x.hi*y.lo),(x.hi*y.hi));
                   }
                    else
                   {
                   z.lo=Math.min((x.hi*y.lo),(x.hi*y.hi));
                   }
              if (Math.max((x.lo*y.lo),(x.lo*y.hi)) >
                  Math.max((x.hi*y.lo),(x.hi*y.hi)))
                   {
                   z.hi=Math.max((x.lo*y.lo),(x.lo*y.hi));
                   }
                    else
                   {
                   z.hi=Math.max((x.hi*y.lo),(x.hi*y.hi));
                   }
              return(z);

       }
/*********************************************************************\
*                                                                     *
*             5) Metodo invmul                                        *
*                Calcula o intervalo reciproco ou                     *
*                Pseudo inverso multiplicativo intervalar             *
\*********************************************************************/
 public static RealInterval invmul(RealInterval x)
         {
               RealInterval z = new RealInterval();
               if ((x.lo <= 0.0) && (x.hi >= 0.0))
               {
  z.Err = "invmul: Os valores assumidos do intervalo nao deve conter zero";
               }
               else
               {
               z.lo = 1/x.hi;
               z.hi = 1/x.lo;
               }
               return(z);
        }
```

```
/********************************************************************\
*                                                                    *
*             6) Metodo sclmul                                       *
*                Calcula a multiplicacao de um escalar por um intervalo  *
*                                                                    *
\********************************************************************/
         public static RealInterval sclmul(double d , RealInterval x)
        // Multiplicao de um escalar
        {
                RealInterval z = new RealInterval();

                z.lo = Math.min(d*x.lo,d*x.hi);
                z.hi = Math.max(d*x.lo,d*x.hi);
                return z;
        }
/********************************************************************\
*                                                                    *
*             7) Metodo div                                          *
*                Calcula a divisao de intervalos                     *
*                                                                    *
\********************************************************************/
        public static RealInterval div(RealInterval x, RealInterval y)
        {
                RealInterval z = new RealInterval();
                if ((y.lo<=0.0) && (y.hi>=0.0))
                {
                z.Err =
         "div: A divisao(metodo div) nao aceita um intervalo contendo zero";
                }
                else
                if (Math.min((x.lo/y.hi),(x.lo/y.lo)) >
                    Math.min((x.hi/y.hi),(x.hi/y.lo)))
                    {
                    z.lo=Math.min((x.hi/y.hi),(x.hi/y.lo));
                    }
                     else
                    {
                    z.lo=Math.min((x.lo/y.hi),(x.lo/y.lo));
                    }
               if (Math.max((x.lo/y.hi),(x.lo/y.lo)) >
                   Math.max((x.hi/y.hi),(x.hi/y.lo)))
                    {
                    z.hi=Math.max((x.lo/y.hi),(x.lo/y.lo));
                    }
                     else
                    {
                    z.hi=Math.max((x.hi/y.hi),(x.hi/y.lo));
                    }
               return(z);
           }
```

```
/*..................................................................\
.                                                                   .
.             MODULO 2- Funcoes entre conjuntos                     .
.             ---------------------------------                     .
.                                                                   .
\.................................................................*\

*******************************************************************\
*                                                                  *
*              8) Metodo union                                     *
*                 Calcula a uniao de intervalos                    *
*                                                                  *
\*******************************************************************/
        public static RealInterval union(RealInterval x, RealInterval y)
                throws IAException
        {
                RealInterval z = new RealInterval();
                if ((x.hi < y.lo) || (y.hi < x.lo))
                {
                z.Err = "union: A intessecao e vazia!";
                }
                else if (x.lo <= y.lo)
                //se o extremo inferior x.lo for menor que o extermo inferior y.lo
                    {
                    z.lo = x.lo;
                    z.hi = Math.max(x.hi,y.hi);
                    }
                    else
                    {
                    z.lo = y.lo;
                    z.hi = Math.max(x.hi,y.hi);
                }
            return z;
        }
/*******************************************************************\
*                                                                  *
*              9) Metodo hull                                      *
*                 Uniao Convexa, faz a concatenacao de dois intervalos    *
*                                                                  *
\*******************************************************************/
public static RealInterval hull(RealInterval x, RealInterval y)
  {
      RealInterval z = new RealInterval();
      z.lo=Math.min(x.lo,y.lo);
      z.hi=Math.max(x.hi,y.hi);
      return z;
  }
```

```
/*********************************************************************\
*                                                                     *
*            10) Metodo intersect                                     *
*               Calcula a intersecao de intervalos                    *
*                                                                     *
\*********************************************************************/
        public static RealInterval intersect(RealInterval x, RealInterval y)
                throws IAException
        {
                RealInterval z = new RealInterval();
                if (((x.lo<=y.hi)&&(y.hi<=x.hi))||((y.lo<=x.hi)&&(x.hi<=y.hi)))
                   {
                   z.lo = Math.max(x.lo,y.lo);
                   z.hi = Math.min(x.hi,y.hi);
                   }
                else
                   z.Err = "intersect: Nao ha intersecao";
            return z;
        }
/*....................................................................\
.                                                                     .
.           MODULO 3- Funcoes geometricas                             .
.           ------------------------------                            .
.                                                                     .
\...................................................................*\

/*********************************************************************\
*                                                                     *
*            11) Metodo dist                                          *
*                Calcula a distancia entre intervalos                 *
*                                                                     *
\*********************************************************************/
        public static double dist(RealInterval x, RealInterval y)
        {
                return  (Math.max(Math.abs(x.lo-y.lo),Math.abs(x.hi-y.hi)));

        }
/*********************************************************************\
*                                                                     *
*           12) Metodo diam                                           *
*               Calcula a diametro de um intervalo                    *
*                                                                     *
\*********************************************************************/
        public static double diam(RealInterval x)
        {
              return  (x.hi-x.lo);
        }
```

```
/********************************************************************\
*                                                                    *
*             13) Metodo midpoint                                    *
*                 Calcula o ponto medio de um intervalo              *
*                                                                    *
\********************************************************************/
        public static double midpoint(RealInterval x)
        {
             return  (x.lo + x.hi)/2.0;
        }
/*...................................................................\
.                                                                    .
.             MODULO 4 - Funcoes elementares                         .
.             ------------------------------                         .
.                                                                    .
\..................................................................*\
/********************************************************************\
*                                                                    *
*             14) Metodo abs                                         *
*                  Calcula o valor absoluto de um intervalo          *
*                                                                    *
\********************************************************************/
public static RealInterval abs(RealInterval x) throws IAException
  {
      RealInterval z = new RealInterval();
      if ((x.lo <= 0.0) && (x.hi >= 0.0))
        z = new RealInterval(0.0,Math.max(Math.abs(x.lo),Math.abs(x.hi)));
      else
        z = new RealInterval(Math.min(Math.abs(x.lo),Math.abs(x.hi)),
        Math.max(Math.abs(x.lo),Math.abs(x.hi)));
      return z;
  }
/********************************************************************\
*                                                                    *
*             15) Metodo sqr                                         *
*                 Calcula o quadrado de um intervalo                 *
*                                                                    *
\********************************************************************/
        public static RealInterval sqr(RealInterval x)
        {
                RealInterval z = new RealInterval();
                if ((x.lo<0) && (x.hi>0)) {
                    z.lo=0;
                    z.hi=x.hi*x.hi;
                }
                else
                   {
                   z.lo = Math.min(x.lo*x.lo,x.hi*x.hi);
                   z.hi = Math.max(x.lo*x.lo,x.hi*x.hi);
                   }
                return (z);
        }
```

```
/*******************************************************************\
*                                                                   *
*            16) Metodo sqrt                                        *
*               Calcula a raiz quadrada de um intervalo             *
*                                                                   *
\*******************************************************************/
       public static RealInterval sqrt(RealInterval x)
       {
               RealInterval z = new RealInterval();
               String Erro;
               if (x.lo < 0.0)
          z.Err = "sqrt: O intervalo nao pode possuir raizes imaginarias";
               z.lo=Math.sqrt(x.lo);
               z.hi=Math.sqrt(x.hi);
               return (z);
        }
/*******************************************************************\
*                                                                   *
*            17) Metodo exp                                         *
*               Calcula a funao exponencial de um intervalo        *
*                                                                   *
\*******************************************************************/
       public static RealInterval exp(RealInterval x)
  {
               RealInterval z;
               z = new RealInterval();
               double hi ,lo;
               hi =lo = 0;
               z.lo = Math.pow(Math.E,x.lo);
               z.hi = Math.pow(Math.E,x.hi);
               return(z);

 }
/*******************************************************************\
*                                                                   *
*            18) Metodo exp2                                        *
*               Calcula a funcao exponencial                        *
*               de um intervalo elevado a 2                         *
*                                                                   *
\*******************************************************************/
       public static RealInterval exp2(RealInterval x)
       {
               RealInterval z;
               z = new RealInterval();
               double hi ,lo;
               hi =lo = 0;
               z.lo = Math.pow(2,x.lo);
               z.hi = Math.pow(2,x.hi);
               return(z);
       }
```

```
/*********************************************************************\
*                                                                     *
*            19) Metodo exp10                                         *
*                Calcula a funao exponencial                        *
*                de um intervalo elevado a 10                         *
*                                                                     *
\*********************************************************************/
        public static RealInterval exp10(RealInterval x)
        {
                RealInterval z;
                z = new RealInterval();
                String Erro;
                double hi, lo;
                hi =lo = 0;
                z.lo = Math.pow(10,x.lo);
                z.hi = Math.pow(10,x.hi);
                return z;
       }
/*********************************************************************\
*                                                                     *
*            20) Metodo power                                         *
*                Calcula a funcao potencia                            *
*                de um intervalo                                      *
\*********************************************************************/
        public static RealInterval power(RealInterval x, int k)
        {
               RealInterval a  = new RealInterval();
               double n = (double)k;
               if ((x.lo<0)&&(x.hi>=0)&&(n%2==0))
                  {
                  a.lo = 0;
                  a.hi = Math.max(Math.pow(Math.abs(x.lo),n),Math.pow(Math.abs(x.hi),n));
                  }
               else
               if ((x.hi<0)&&(n%2==0))
                 {
                  a.lo = Math.min(Math.pow(Math.abs(x.lo),n),Math.pow(Math.abs(x.hi),n));
                  a.hi = Math.max(Math.pow(Math.abs(x.lo),n),Math.pow(Math.abs(x.hi),n));
                 }
               else
               {
                 a.lo = Math.pow(x.lo,n);
                 a.hi = Math.pow(x.hi,n);
               }
               return(a);
       }
```

```
/********************************************************************\
*                                                                    *
*            21) Metodo ln                                           *
*               Calcula a funcao logaritmo natural de um intervalo   *
*                                                                    *
\********************************************************************/
        public static RealInterval ln(RealInterval x)
           {
                String Erro;
                RealInterval z;
                z =  new RealInterval();
                if (x.lo <= 0.0)
                {
                z.Err = "ln: O intervalo no pode ser menor ou igual zero";
                }
                else
                {
                z.lo = Math.log(x.lo);
                z.hi = Math.log(x.hi);
                }
                return z;
            }
/********************************************************************\
*                                                                    *
*            22) Metodo log2                                         *
*               Calcula a funcao logaritmo na base 2 de um intervalo *
*                                                                    *
\********************************************************************/
        public static RealInterval log2(RealInterval x)
        {
                RealInterval result = new RealInterval();
                RealInterval z = new RealInterval();
                z.lo = 2.0;
                z.hi = 2.0;
                if (x.lo <= 0.0)
                {
                z.Err = "log2: O intervalo nao pode ser menor ou igual zero";
                }
                else
                {
                result  = div(ln(x),ln(z));
                }
                return result;
        }
```

```
/*********************************************************************\
*                                                                     *
*            23) Metodo log10                                         *
*                Calcula a funcao logaritmo na base 10 de um intervalo   *
*                                                                     *
\*********************************************************************/
        public static RealInterval log10(RealInterval x)
        {
                RealInterval result = new RealInterval();
                RealInterval z = new RealInterval();
                z.lo = 10.0;
                z.hi = 10.0;
                if (x.lo <= 0.0)
                {
                z.Err = "log10: O intervalo nao pode ser menor ou igual zero";
                }
                else
                {
                result  = div(ln(x),ln(z));
                }
                result  = div(ln(x),ln(z));
                return result;
        }
/*...................................................................\
.                                                                   .
.            MODULO 5- Funcoes trigonometricas                      .
.            ---------------------------------                      .
.                                                                   .
\..................................................................*\

/*********************************************************************\
*                                                                     *
*            24) Metodo sin                                           *
*              Calcula a funcao seno de um intervalo                  *
*                                                                     *
\*********************************************************************/
        public static RealInterval sin(RealInterval x)
{
            RealInterval result;
            result = new RealInterval();
            double troc, minM, maxM,k;
            //
            // Se x2-x1 >= 2PI, a funcao ja passou pelos seus extremos -1 e 1
            //
            if ((x.hi-x.lo) > 2 * Math.PI)
                 {
                 result.lo=-1;
                 result.hi=1;
                                 }
            else
```

```
//
// Se x2-x1 estiver entre PI e 2PI
//
if ((x.hi-x.lo) > Math.PI)
{
      if (Math.sin(x.lo + ((x.hi-x.lo)/2)) == 0.0)
         {
          result.lo = -1;
          result.hi = 1;
         }
      else
      if (Math.sin (x.lo + ((x.hi-x.lo)/2)) > 0.0)
         {    // minM e 1
          result.lo=Math.min(Math.sin(x.lo), Math.sin(x.hi));
          result.hi=1;
          }
      else
          {   // -1 e maxM
          result.lo=-1 ;
          result.hi=Math.max(Math.sin(x.lo), Math.sin(x.hi));
          }
 }
 else
 {
 //
 // Se x2-x1 for menor que PI
 //
 k=((int)((2*(x.lo))/Math.PI))+1;
 // se x1 <= k(pi/2) <= x2 e
 if ((x.lo <= (k*(Math.PI/2))) && ((k*(Math.PI/2)) <= x.hi))
 {
     // para algum k, tal que k-1 seja multiplo de 4
     if ((k-1)%4==0.0)
     {         //minM e 1
     result.lo=Math.min(Math.sin(x.lo), Math.sin(x.hi));
     result.hi = 1;
     }
     else
     // para algum k, tal que k+1 seja multiplo de 4
     if ((k+1)%4==0.0)
     {        //-1 e maxM
     result.lo=-1;
     result.hi=Math.max(Math.sin(x.lo), Math.sin(x.hi));
     }
     else
     {        //minM e maxM
     result.lo=Math.min(Math.sin(x.lo), Math.sin(x.hi));
     result.hi=Math.max(Math.sin(x.lo), Math.sin(x.hi));
     }
 }
 else
```

```
            {       // minM e maxM
            result.lo = Math.min(Math.sin(x.lo), Math.sin(x.hi));
            result.hi = Math.max(Math.sin(x.lo), Math.sin(x.hi));
            }
            }
             return result;
        }

/*********************************************************************\
*                                                                     *
*            25) Metodo cos                                           *
*              Calcula a funcao co-seno de um intervalo               *
*                                                                     *
\*********************************************************************/
        public static RealInterval cos(RealInterval x)
            {
                RealInterval result,z;
                result = new RealInterval();
                x.lo=x.lo + Math.PI/2;
                x.hi=x.hi + Math.PI/2;
                result = sin(x);
                return result;
            }
/*********************************************************************\
*                                                                     *
*            26) Metodo tan                                           *
*                Calcula a funcao tangente de um intervalo            *
*                                                                     *
\*********************************************************************/
        public static RealInterval tan(RealInterval x)
        {
        RealInterval z = new RealInterval();
        int k,k1;
        if ((x.hi-x.lo) >= Math.PI)
           z.Err = "tan: Diverge, pois o intervalo nao pertence ao dominio da funcao";
        k1= (int) Math.floor(x.lo/(Math.PI/2)) + 1;
        if ((k1%2!=0)&&(k1*Math.PI/2>=x.lo)&&(k1*Math.PI/2<=x.hi))
           z.Err = "tan: Diverge, pois o intervalo nao pertence ao dominio da funcao";
        else
           {
           z.lo =Math.tan(x.lo);
           z.hi =Math.tan(x.hi);
           }
        return(z);
        }
```

```
/*******************************************************************\
*                                                                   *
*            27) Metodo cot                                         *
*                Calcula a funcao Co-tangente de um intervalo       *
*                                                                   *
\*******************************************************************/
        public static RealInterval cot(RealInterval x)
        {
        RealInterval z = new RealInterval();
        int k,k1;
        if ((x.hi-x.lo) >= Math.PI)
           z.Err = "cot: Diverge, pois o intervalo nao pertence ao dominio da funcao";
        k1= (int) Math.floor(x.lo/(Math.PI)) + 1;
        if ((k1*Math.PI>=x.lo)&&(k1*Math.PI<=x.hi))
           z.Err = "cot: Diverge, pois o intervalo nao pertence ao dominio da funcao";
        else
           {
           z.hi =(1/Math.tan(x.lo));
           z.lo =(1/Math.tan(x.hi));
           }
        return(z);

        }
/*******************************************************************\
*                                                                   *
*            28) Metodo sec                                         *
*                Calcula a funcao Secante de um intervalo           *
*                                                                   *
\*******************************************************************/
        public static RealInterval sec(RealInterval x)
        {
        RealInterval z = new RealInterval();
        int k,k1;
        if ((x.hi-x.lo) >= Math.PI)
           z.Err = "sec: Diverge, pois o intervalo nao pertence ao dominio da funcao";
        k1= (int) Math.floor(x.lo/(Math.PI/2)) + 1;
        k = (int) Math.floor(x.lo/Math.PI) + 1;
        if ((k1%2!=0)&&(k1*Math.PI/2>=x.lo)&&(k1*Math.PI/2<=x.hi))
              z.Err = " Diverge, pois o intervalo nao pertence ao dominio da funcao";
        if ((k%2!=0) && (k*Math.PI>=x.lo) && (k*Math.PI<=x.hi))
           {
           z.lo =Math.min(1/Math.cos(x.lo),1/Math.cos(x.hi));
           z.hi = -1;
           }
        else
           if ((k%2==0)&&(k*Math.PI>= x.lo) && (k*Math.PI<=x.hi)
              && (x.lo!=0))
              {
              z.hi = Math.max(1/Math.cos(x.lo),1/Math.cos(x.hi));
              z.lo = 1;
              }
           else
```

```
            {
            z.hi = Math.max(1/Math.cos(x.lo),1/Math.cos(x.hi));
            z.lo = Math.min(1/Math.cos(x.lo),1/Math.cos(x.hi));
            }
        return(z);

        }

/*********************************************************************\
*                                                                     *
*            29) Metodo csc                                           *
*                Calcula a funcao co-secante de um intervalo          *
*                                                                     *
\*********************************************************************/
        public static RealInterval csc(RealInterval x)
        {
        RealInterval z = new RealInterval();
        x.lo = x.lo - Math.PI/2;
        x.hi = x.hi - Math.PI/2;
        z = sec(x);
        return(z);

        }

/*********************************************************************\
*                                                                     *
*            30) Metodo arcsin                                        *
*                Calcula a funcao arco seno                           *
*                                                                     *
\*********************************************************************/
        public static RealInterval arcsin(RealInterval x)
                throws IAException
        {
                RealInterval z;
                z = new RealInterval();
                if ((x.lo < -1) || (x.hi > 1))
                    {
                z.Err = "arcsen: Neste intervalo a funcao nao tem inversa";
                    }
                z.lo = Math.asin(x.lo);
                z.hi = Math.asin(x.hi);
                return z;
        }
```

```
/*******************************************************************\
*                                                                   *
*            31) Metodo arccos                                      *
*               Calcula a funcao arco co-seno                       *
*                                                                   *
\*******************************************************************/
        public static RealInterval arccos(RealInterval x)
         {
                RealInterval z;
                z = new RealInterval();
                if ((x.lo < -1.0) || (x.hi > 1))
                    {
                z.Err = "arccos: Neste intervalo a funcao nao tem inversa";
                    }
                else
                {
                z.lo = Math.acos(x.hi);
                z.hi = Math.acos(x.lo);
                }
                return z;
        }
/*******************************************************************\
*                                                                   *
*            32) Metodo arctan                                      *
*               Calcula a funcao arco tangente de um intervalo      *
*                                                                   *
\*******************************************************************/
        public static RealInterval arctan(RealInterval x)
        {
                RealInterval y,z,result;
                z = new RealInterval();
                z.lo = Math.atan(x.lo);
                z.hi = Math.atan(x.hi);
                return z;
        }
/*******************************************************************\
*                                                                   *
*            33) Metodo sinh                                        *
*               Calcula a funcao seno hiperbolico                   *
*                                                                   *
\*******************************************************************/
        public static RealInterval sinh(RealInterval x)
        {
                RealInterval z = new RealInterval();
                z.lo = (Math.pow(Math.E,x.lo)-Math.pow(Math.E,-x.lo))/2;
                z.hi = (Math.pow(Math.E,x.hi)-Math.pow(Math.E,-x.hi))/2;

                return(z);
        }
```

```
/*******************************************************************\
*                                                                   *
*            34) Metodo cosh                                        *
*                Calcula a funcao co-seno hiperbolico               *
*                                                                   *
\*******************************************************************/
        public static RealInterval cosh(RealInterval x)
        {
         RealInterval z = new RealInterval();
         double a,b;
         a = (Math.pow(Math.E,x.lo) +  Math.pow(Math.E,-x.lo))/2;
         b = (Math.pow(Math.E,x.hi) +  Math.pow(Math.E,-x.hi))/2;
         if ((x.lo<=0)&&(x.hi>=0))
            {
            z.lo = 1;
            if (Math.abs(x.lo)>=x.hi)
               z.hi = a;
            else
               z.hi = b;
            }
         else
            {
            z.lo = Math.min(a,b);
            z.hi = Math.max(a,b);
            }
         return(z);
        }
/*******************************************************************\
*                                                                   *
*            35) Metodo tanh                                        *
*                Calcula a funcao tangente hiperbolico              *
*                                                                   *
\*******************************************************************/
        public static RealInterval tanh(RealInterval x)
        {
                RealInterval z = new RealInterval();
                z.hi =  (Math.pow(Math.E,x.lo)-Math.pow(Math.E,-x.lo))/
                        (Math.pow(Math.E,x.lo)+Math.pow(Math.E,-x.lo));
                z.lo =  (Math.pow(Math.E,x.hi)-Math.pow(Math.E,-x.hi))/
                        (Math.pow(Math.E,x.hi)+Math.pow(Math.E,-x.hi));
                return(z);
        }
```

```
/*******************************************************************\
*                                                                   *
*            36) Metodo coth                                        *
*                Calcula a funcao co-tangente hiperbolica           *
*                                                                   *
\*******************************************************************/
        public static RealInterval coth(RealInterval x)
        {
                RealInterval z = new RealInterval();
                double troc;
                if ((x.lo <= 0.0) && (x.hi >= 0.0))
                    z.Err = "coth: O intervalo nao pode conter o valor zero";
                else
                {
                z.hi =  (Math.pow(Math.E,x.lo)+Math.pow(Math.E,-x.lo))/
                        (Math.pow(Math.E,x.lo)-Math.pow(Math.E,-x.lo));
                z.lo =  (Math.pow(Math.E,x.hi)+Math.pow(Math.E,-x.hi))/
                        (Math.pow(Math.E,x.hi)-Math.pow(Math.E,-x.hi));
                }
                return(z);
        }
/*******************************************************************\
*                                                                   *
*            37) Metodo sech                                        *
*                Calcula a funcao secante hiperbolica               *
*                                                                   *
\*******************************************************************/
        public static RealInterval sech(RealInterval x)
        {
                RealInterval z = new RealInterval();
                double a,b,troc;
                a = 2.0/(Math.pow(Math.E,x.lo)+Math.pow(Math.E,-x.lo));
                b = 2.0/(Math.pow(Math.E,x.hi)+Math.pow(Math.E,-x.hi));
                if((x.lo<0)&&(x.hi>0)) {
                    z.hi = 1;
                    z.lo = Math.min(a,b);
                }
                else {
                    z.lo = Math.min(a,b);;
                    z.hi = Math.max(a,b);;
                }
                return(z);
        }
```

```
/*******************************************************************\
*                                                                   *
*           38) Metodo csch                                         *
*               Calcula a funcao co-secante hiperbolica             *
*                                                                   *
\*******************************************************************/
        public static RealInterval csch(RealInterval x)
        {
                RealInterval z = new RealInterval();
                double a,b,troc;
                a = 2.0/(Math.pow(Math.E,x.lo)-Math.pow(Math.E,-x.lo));
                b = 2.0/(Math.pow(Math.E,x.hi)-Math.pow(Math.E,-x.hi));
                if((x.lo<0)&&(x.hi>0))
                    z.Err = "csch: O intervalo nao deve conter zero";
                else
                    {
                    z.lo = Math.min(a,b);;
                    z.hi = Math.max(a,b);;
                    }
                return(z);
        }
/*******************************************************************\
*                                                                   *
*           39) Metodo senhI                                        *
*               Calcula o Seno Hiperbolico Inverso                  *
*                                                                   *
\*******************************************************************/
        public static RealInterval senhI(RealInterval x)
        {
                RealInterval z = new RealInterval();
                z.lo = Math.log(x.lo+Math.sqrt(Math.pow(x.lo,2)+1));
                z.hi = Math.log(x.hi+Math.sqrt(Math.pow(x.hi,2)+1));
                return(z);
/*******************************************************************\
*                                                                   *
*           40) Metodo coshI                                        *
*               Calcula o Cosseno Hiperbolico Inverso               *
*                                                                   *
\*******************************************************************/
        public static RealInterval coshI(RealInterval x)
        {
                RealInterval z = new RealInterval();
                if (x.lo<1)
                    z.Err = "coshI: O intervalo [a,b] deve satisfazer a
                condicao a,b >= 1";
                else
                {
                z.lo = Math.log(x.lo+Math.sqrt(Math.pow(x.lo,2)-1));
                z.hi = Math.log(x.hi+Math.sqrt(Math.pow(x.hi,2)-1));
                }
                return(z); }
```

```
/*********************************************************************\
*                                                                     *
*            41) Metodo tanhI                                         *
*                Calcula o Tangente Hiperbolica Inversa               *
*                                                                     *
\*********************************************************************/
        public static RealInterval tanhI(RealInterval x)
        {
                RealInterval z = new RealInterval();
                if (Math.abs(x.lo) >= 1)
                    z.Err = "tanhI: O intervalo deve estar  -1 e 1";
                else
                {
                z.lo = 0.5 * Math.log((1+x.lo)/(1-x.lo));
                z.hi = 0.5 * Math.log((1+x.hi)/(1-x.hi));
                }
                return(z);
         }
/*********************************************************************\
*                                                                     *
*            42) Metodo cothI                                         *
*                Calcula o Co-tangente Hiperbolica Inversa            *
*                                                                     *
\*********************************************************************/
        public static RealInterval cothI(RealInterval x)
        {
                RealInterval z = new RealInterval();
                if ((x.lo >= -1) && (x.hi <= 1))
                  z.Err = "cothI: O intervalo nao pode estar entre -1 e 1";
                else
                {
                z.lo =  0.5 * Math.log((x.hi+1)/(x.hi-1));
                z.hi =  0.5 * Math.log((x.lo+1)/(x.lo-1));
                }
                return(z);
         }
/*********************************************************************\
*                                                                     *
*            43) Metodo sechI                                         *
*                Calcula o Secante Hiperbolica Inversa                *
*                                                                     *
\*********************************************************************/

        public static RealInterval sechI(RealInterval x)
        {
                RealInterval z = new RealInterval();
                double troc;
                if ((x.lo <= 0)  || (x.hi > 1))
                    z.Err = "sechI: O intervalo tem que estar entre  0 e 1";
                else
```

```
                {
                z.hi = Math.log((1+Math.sqrt(1 - Math.pow(x.lo,2)))/(x.lo));
                z.lo = Math.log((1+Math.sqrt(1 - Math.pow(x.hi,2)))/(x.hi));
                }
                return(z);
      }
/*********************************************************************\
*                                                                     *
*            44) Metodo cschI                                         *
*                Calcula o Co-secante Hiperbolica Inversa             *
*                                                                     *
\*********************************************************************/
      public static RealInterval cschI(RealInterval x)
        {
                RealInterval z = new RealInterval();
                double troc;
                if (((x.lo < 0)&& (x.hi > 0)) || ((x.lo == 0)||(x.hi == 0)))
                    z.Err = "cschI: O intervalo nao deve conter zero";
                else
                {
        z.hi = Math.log(1/x.lo + (Math.sqrt(1 + Math.pow(x.lo,2))/Math.abs(x.lo)));
        z.lo = Math.log(1/x.hi + (Math.sqrt(1 + Math.pow(x.hi,2))/Math.abs(x.hi)));
                }
                return(z);
        }
/*....................................................................\
.                                                                     .
.            MODULO 6 - Funcoes de transferencia e                    .
.                       Constantes intervalares                       .
.                       ---------------------------                   .          .
.                                                                     .
\....................................................................*\

/*********************************************************************\
*                                                                     *
*            45) Metodo int_interv                                    *
*                Converte um inteiro em um intervalo                  *
*                                                                     *
\*********************************************************************/

        public static RealInterval int_interv(int i)
        // return Interval N;
        {
                RealInterval z = new RealInterval();
                double x;
                x = (double)i;
                z = new RealInterval(x,x);
                return z;
        }
```

```
/**********************************************************************\
*                                                                      *
*            46) Metodo real_interv                                    *
*                Converte um real em um intervalo                      *
*                                                                      *
\**********************************************************************/
        public static RealInterval real_interv(double i)
        {
                RealInterval z = new RealInterval();
                z = new RealInterval(i,i);
                return z;
        }
/**********************************************************************\
*                                                                      *
*            47) Metodo inf                                            *
*                Retorna o extremo inferior de um intervalo            *
*                                                                      *
\**********************************************************************/
        public static double inf(RealInterval x)
        {
                double y;
                y = x.lo;
                return y;
        }
/**********************************************************************\
*                                                                      *
*            48) Metodo sup                                            *
*                Retorna o extremo superior de um intervalo            *
*                                                                      *
\**********************************************************************/
        public static double sup(RealInterval x)
        {
                double y;
                y = x.hi;
                return y;
        }
/**********************************************************************\
*                                                                      *
*            49) Metodo PI                                             *
*                Calcula o intervalo de PI                             *
*                                                                      *
\**********************************************************************/
        public static RealInterval PI()
        {
                RealInterval z = new RealInterval();
                z.lo = Math.PI;
                z.hi = nextfp(Math.PI);
                return (z);
        }
```

```
/*********************************************************************\
*                                                                     *
*            50) Metodo PI/n                                          *
*               Calcula o intervalo de PI dividido por n              *
*                                                                     *
\*********************************************************************/
        public static RealInterval pi_n(int i)
        {
                RealInterval z = new RealInterval();
                z.lo = PI().lo / i;
                z.hi = PI().hi / i;
                return (z);
        }
/*********************************************************************\
*                                                                     *
*            51) Metodo A                                             *
*               Calcula o intervalo de 1/PI                           *
*                                                                     *
\*********************************************************************/
        public static RealInterval A()
        {
                RealInterval z = new RealInterval();
                z.lo = (1.0 / PI().hi);
                z.hi = (1.0 / PI().lo);
                return (z);
        }
/*********************************************************************\
*                                                                     *
*            52) Metodo blow - retorna um novo intervalo com o        *
*               o predecessor do extremo inferior do intervalo        *
*               e o sucessor do extremo superior do intervalo         *
*                                                                     *
\*********************************************************************/
        public static RealInterval blow(RealInterval x)
        {
                RealInterval z = new RealInterval();
                z.lo = prevfp(x.lo);
                z.hi = nextfp(x.hi);
                return (z);
        }
  public static double nextfp(double x) {
    double y;
    if (x < Double.POSITIVE_INFINITY) {
      long xx = Double.doubleToLongBits(x);
      if (x >= 0)
         y = Double.longBitsToDouble(xx+1);
      else if (x==0)
         y = Double.longBitsToDouble(1);
      else
         y = Double.longBitsToDouble(xx-1);
      return(y);
    }
```

```
    else
      return(x);
  }
  public static double prevfp(double x) {
    if (x==0)
      return(-nextfp(0.0));
    else
      return(-nextfp(-x));
  }

}
```

O Programa Testebib

```
//("Este programa testa a biblioteca Ia_Math,
// que implementa operacoes e funcoes com intervalos");
import java.io.*;
import ia_math.*;
class Testebib {
     public static void main(String arguments[]) {
//      Teste2 t = new Teste2();
        Testebib t = new Testebib();
        String temp;
        temp = "UFRN/DIMAP/LABLIC\nRelatorio do teste na Biblioteca JAVA-XSC,
                \ncom todos";
        temp = temp+" os exemplos descritos na dissertacao nos capitulos 2 e 3 \n";
        StringBuffer texto = new StringBuffer(temp);
        texto.append(t.Soma());
        texto.append(t.Pseudo_aditivo());
        texto.append(t.Subtracao());
        texto.append(t.Multiplicacao());
        texto.append(t.Inverso_multiplicativo());
        texto.append(t.Escalar_multiplicacao());
        texto.append(t.Divisao());
        texto.append(t.Uniao());
        texto.append(t.Uniao_convexa());
        texto.append(t.Intersecao());
        texto.append(t.Distancia());
        texto.append(t.Diametro());
        texto.append(t.Ponto_medio());
        texto.append(t.Absoluto());
        texto.append(t.Quadrado());
        texto.append(t.Raiz_quadrada());
        texto.append(t.Exponencial());
        texto.append(t.Exponencial_2());
        texto.append(t.Exponencial_10());
        texto.append(t.Potencia());
        texto.append(t.Logaritmo_natural());
        texto.append(t.Logaritmo_2());
        texto.append(t.Logaritmo_10());
```

```
        texto.append(t.Seno());
        texto.append(t.Co_seno());
        texto.append(t.Tangente());
        texto.append(t.Co_tangente());
        texto.append(t.Secante());
        texto.append(t.Co_secante());
        texto.append(t.Arco_seno());
        texto.append(t.Arco_co_seno());
        texto.append(t.Arco_tangente());
        texto.append(t.Seno_h());
        texto.append(t.Co_seno_h());
        texto.append(t.Tangente_h());
        texto.append(t.Co_tangente_h());
        texto.append(t.Secante_h());
        texto.append(t.Co_secante_h());
        texto.append(t.Seno_h_I());
        texto.append(t.Co_seno_h_I());
        texto.append(t.Tangente_h_I());
        texto.append(t.Co_tangente_h_I());
        texto.append(t.Secante_h_I());
        texto.append(t.Co_secante_h_I());
        texto.append(t.Inteirop_intervalo());
        texto.append(t.Realp_intervalo());
        texto.append(t.Extremos_inferior_e_superior());
        texto.append(t.Constantes_pi());
        texto.append(t.Inflacionamento());
        temp = texto.toString();
        try {
            FileWriter temp2 = new FileWriter("teste.txt");
            int a;
            a = (temp.length() -1);
            for (int i=0;i<a;i++)
                temp2.write(temp.charAt(i));
            temp2.close();

       }catch(IOException e) {
            System.out.println("Error -" +e.toString());
        }
}
/**************************************************************************/
        public String Soma () {
          String temp;
          temp= "-----------------------------------------------------------";
            temp= temp + "\nMODULO-1 OPERACOES BASICAS - TESTE - JAVA-XSC";
          RealInterval A = new RealInterval(2.0,3.0);
          RealInterval B = new RealInterval(4.0,5.0);
          RealInterval c;
          c= IAMath.add(A,B);
          temp = temp+"\n-----------------------------------------------------------";
          temp =temp + "\n1)Soma\n\tA=[2.0;3.0] B=[4.0;5.0]\n\tadd(A,B)= ";
          temp = temp + c +"\n";
          return(temp); }
```

```
/**********************************************************************/
        public String Pseudo_aditivo() {
          String temp;
          RealInterval c;
          RealInterval A = new RealInterval(2.0,3.0);
          c= IAMath.uminus(A);
          temp = "-----------------------------------------------------------";
          temp =temp +"\n2)Pseudo aditivo\n\tA=[2.0;3.0]\n\tuminus(A,B)= ";
          temp = temp + c+"\n";
          return(temp);
        }
/**********************************************************************/
        public String Subtracao() {
          String temp;
          RealInterval A = new RealInterval(2.0,3.0);
          RealInterval B = new RealInterval(4.0,5.0);
          RealInterval c;
          c= IAMath.sub(A,B);
          temp = "-----------------------------------------------------------";
          temp =temp +"\n3)Subtracao\n\tA=[2.0;3.0] B=[4.0;5.0]\n\tsub(A,B)= ";
          temp = temp + c+"\n";
          return(temp);
        }
/**********************************************************************/
        public String Multiplicacao() {
          String temp;
          RealInterval A = new RealInterval(-2.0,3.0);
          RealInterval B = new RealInterval(-4.0,5.0);
          RealInterval c;
          c= IAMath.mul(A,B);
          temp = "-----------------------------------------------------------";
          temp =temp +"\n4)Multiplicacao\n\tA=[-2.0;3.0] B=[-4.0;5.0]\n
                \tmul(A,B)= ";
          temp = temp + c+"\n";
          return(temp);
        }
/**********************************************************************/
        public String Inverso_multiplicativo () {
          String temp;
          RealInterval A = new RealInterval(-3.0,-2.0);
          RealInterval c;
          c= IAMath.invmul(A);
          temp = "-----------------------------------------------------------";
          temp =temp +"\n5)Pseudo inverso multiplicativo\n\tA=[-3.0;-2.0]
                \n\tinvmul(A)= ";
          temp = temp + c+"\n";
          return(temp);
        }
```

```
/*********************************************************************/
         public String Escalar_multiplicacao () {
          String temp;
          double b = -2;
          RealInterval A = new RealInterval(-2.0,1.0);
          RealInterval c;
          c= IAMath.sclmul(b,A);
          temp = "-------------------------------------------------------";
          temp =temp +"\n6)Multiplicacao por um escalar\n\tA=[-2.0;1.0] e
                x = -2.0\n\tsclmul(A)= ";
          temp = temp + c+"\n";
          return(temp);
        }
/*********************************************************************/
        public String Divisao () {
          String temp;
          RealInterval A = new RealInterval(2.0,3.0);
          RealInterval B = new RealInterval(4.0,5.0);
          RealInterval c;
          c= IAMath.div(A,B);
          temp = "-------------------------------------------------------";
          temp =temp +"\n7)Divisao \n\tA=[2.0;3.0] B=[4.0;5.0]\n\tdiv(A,B)= ";
          temp = temp + c+"\n";
          return(temp);
        }
/*********************************************************************/
        public String Uniao () {
          String temp;
          RealInterval A = new RealInterval(-3.0,2.0);
          RealInterval B = new RealInterval(0.0,5.0);
          RealInterval c;
          temp= "-------------------------------------------------------";
          temp= temp + "\nMODULO-2 FUNCOES ENTRE CONJUNTOS - TESTE - JAVA-XSC";
          c= IAMath.union(A,B);
          temp = temp+"\n-------------------------------------------------------";
          temp =temp +"\n8)Uniao\n\tA=[-3.0;2.0] B=[0.0;5.0]\n\tunion(A,B)= ";
          temp = temp + c+"\n";
          A = new RealInterval(2.0,3.0);
          B = new RealInterval(4.0,5.0);
          c= IAMath.union(A,B);
          if (c.Err=="") {
                temp =temp +"\n8)Uniao\n\tA=[2.0;3.0] B=[4.0;5.0]\n\tunion(A,B)= ";
                temp = temp + c+"\n";
          }
          else
                temp =temp +"\n\tC=[2;3] D=[4;5]\n\tunion(C,D)= ";
                temp = temp + c.Err+"\n";
          return(temp);
        }
```

```
/**************************************************************************/
        public String Uniao_convexa () {
          String temp;
          RealInterval A = new RealInterval(-3.0,2.0);
          RealInterval B = new RealInterval(0.0,5.0);
          RealInterval C = new RealInterval(2.0,3.0);
          RealInterval D = new RealInterval(4.0,5.0);
          RealInterval E = new RealInterval(3.0,8.0);
          RealInterval F = new RealInterval(-1.0,10.0);
          A= IAMath.hull(A,B);
          temp = "------------------------------------------------------------";
          temp =temp +"\n9)Uniao convexa\n\tA=[-3.0;2.0] B=[0.0;5.0]
                \n\thull(A,B)= ";
          temp = temp + A +"\n";
          A= IAMath.hull(C,D);
          temp =temp + "\n\tC=[2.0;3.0] D=[4.0;5.0]\n\thull(C,D)= ";
          temp = temp + A + "\n";
          A= IAMath.hull(E,F);
          temp =temp +"\n\tE=[3.0;8.0] F=[-1;10.0]\n\thull(E,F)= ";
          temp = temp + A + "\n";
          return(temp);
        }
/**************************************************************************/
        public String Intersecao () {
          String temp;
          RealInterval A = new RealInterval(2.0,5.0);
          RealInterval B = new RealInterval(3.0,8.0);
          RealInterval c;
          c= IAMath.intersect(A,B);
          temp = "------------------------------------------------------------";
          temp =temp +"\n10)Intersecao\n\tA=[2.0;5.0] B=[3.0;8.0]
                \n\tintersect(A,B)= ";
          if (c.Err!="")
               temp = temp + c.Err+"\n";
          else
               temp = temp + c+"\n";
          A = new RealInterval(2.0,5.0);
          B = new RealInterval(6.0,8.0);
          c= IAMath.intersect(A,B);
          temp =temp +"\n\tC=[2.0;5.0] D=[6.0;8.0]\n\tintersect(C,D)= ";
          if (c.Err!="")
               temp = temp + c.Err+"\n";
          else
               temp = temp + c+"\n";

          return(temp);
        }
```

```
/**********************************************************************/
        public String Distancia () {
          String temp;
          RealInterval A = new RealInterval(2.0,5.0);
          RealInterval B = new RealInterval(-2.0,-1.0);
          double c;
          c= IAMath.dist(A,A);
          temp= "-------------------------------------------------------";
          temp= temp + "\nMODULO-3 FUNCOES GEOMETRICAS - TESTE - JAVA-XSC";
          temp =temp + "\n-------------------------------------------------------";
          temp =temp +"\n11)Distancia\n\tA=[2.0;5.0] B=[-2.0;-1.0]\n\n\tdist(A,A)= ";
          temp = temp + c +"\n";
          c= IAMath.dist(A,B);
          temp =temp +"\tdist(A,B)= ";
          temp = temp + c +"\n";
          return(temp);
        }
/**********************************************************************/
        public String Diametro () {
          String temp;
          RealInterval A = new RealInterval(2.0,5.0);
          double c;
          c= IAMath.diam(A);
          temp = "-------------------------------------------------------";
          temp =temp +"\n12)Diametro\n\tA=[2.0;5.0]\n\tdiam(A)= ";
          temp = temp + c +"\n";
          return(temp);
        }
/**********************************************************************/
        public String Ponto_medio () {
          String temp;
          RealInterval A = new RealInterval(2.0,6.0);
          double c;
          c= IAMath.midpoint(A);
          temp = "-------------------------------------------------------";
          temp =temp +"\n13)Ponto medio\n\tA=[2.0;6.0]\n\tmidpoint(A)= ";
          temp = temp + c +"\n";
          return(temp);
        }
/**********************************************************************/
        public String Absoluto () {
          String temp;
          temp= "-------------------------------------------------------";
          temp= temp + "\nMODULO-4 FUNCOES ELEMENTARES - TESTE - JAVA-XSC";
          RealInterval A = new RealInterval(-4.0,0.0);
          RealInterval B = new RealInterval(-3.0,-1.0);
          RealInterval C = new RealInterval(-2.0,3.0);
```

```
    A= IAMath.abs(A);
    B= IAMath.abs(B);
    C= IAMath.abs(C);
    temp = temp + "\n-----------------------------------------------------------";
    temp =temp +"\n14)Valor absoluto\n\tA=[-4.0;0.0] B=[-3.0;-1.0]
    C=[-2.0;3.0]";
    temp = temp +"\n\tabs(A)= " +A + "\n\tabs(B)= " + B +
    "\n\tabs(C)= " + C+"\n";
    return(temp);
  }
/***************************************************************************/
  public String Quadrado () {
    String temp;
    RealInterval A = new RealInterval(-1.0,2.0);
    A= IAMath.sqr(A);
    temp= "------------------------------------------------------------";
    temp= temp + "\nMODULO-4 FUNCOES ELEMENTARES - TESTE - JAVA-XSC";
    temp = "------------------------------------------------------------";
    temp =temp +"\n15)Quadrado\n\tA=[-1;2]\n\tsqr(A)= ";
    temp = temp + A +"\n";
    A = new RealInterval(-2.0,-1.0);
    A= IAMath.sqr(A);
    temp =temp +"\n\tB=[-2;-1]\n\tsqr(B)= ";
    temp = temp + A +"\n";
    A = new RealInterval(1.0,2.0);
    A= IAMath.sqr(A);
    temp =temp +"\n\tC=[1;2]\n\tsqr(C)= ";
    temp = temp + A +"\n";

    return(temp);
  }
/***************************************************************************/

  public String Raiz_quadrada () {
    String temp;
    RealInterval A = new RealInterval(9.0,25.0);
    A= IAMath.sqrt(A);
    temp = "------------------------------------------------------------";
    temp =temp +"\n16)Raiz quadrada\n\tA=[9.0;25.0]\n\tsqrt(A)= ";
    temp = temp + A +"\n";
    A = new RealInterval(-4.0,25.0);
    A= IAMath.sqrt(A);
    temp =temp +"\n\tA=[-4.0;25.0]\n\tsqrt(A)= ";
    if (A.Err!="")
    temp = temp + A.Err+"\n";
    else
    temp = temp + A +"\n";
    return(temp);
  }
```

```
/*********************************************************************/
        public String Exponencial () {
          String temp;
          RealInterval A = new RealInterval(0.0,1.0);
          A= IAMath.exp(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n17)Exponencial\n\tA=[0.0;1.0]\n\texp(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(-1.0,1.0);
          A= IAMath.exp(A);
          temp =temp +"\n\tA=[-1.0;1.0]\n\texp(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/*********************************************************************/
         public String Exponencial_2 () {
          String temp;
          RealInterval A = new RealInterval(-1.0,2.0);
          A= IAMath.exp2(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n18)Exponencial com base 2\n\tA=[-1.0;2.0]\n\texp(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/*********************************************************************/
        public String Exponencial_10 () {
          String temp;
          RealInterval A = new RealInterval(-2.0,1.0);
          A= IAMath.exp10(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n19)Exponencial com base 10\n\tA=[-2.0;1.0]\n\texp(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/*********************************************************************/
        public String Potencia () {
          String temp;
          RealInterval A = new RealInterval(-2.0,3.0);
          int n=3;
          A= IAMath.power(A,n);
          temp = "------------------------------------------------------------";
          temp =temp +"\n20)Potencia\n\tA=[-2.0;3.0] n = 3\n\tpower(A,n)= ";
          temp = temp + A +"\n";
          n=2;
          A = new RealInterval(-3.0,2.0);
          A= IAMath.power(A,n);
          temp =temp +"\n\tB=[-3.0;2.0] n = 2\n\tpower(B,n)= ";
          temp = temp + A +"\n";
          A = new RealInterval(-5.0,-3.0);
          n=2;
          A= IAMath.power(A,n);
```

```
          temp =temp +"\n\tC=[-5.0;-3.0] n = 2\n\tpower(C,n)= ";
          temp = temp + A +"\n";
          A = new RealInterval(2.0,4.0);
          n=2;
          A= IAMath.power(A,n);
          temp =temp +"\n\tD=[2.0;4.0] n = 2\n\tpower(D,n)= ";
          temp = temp + A +"\n";
          return(temp);

        }
/**************************************************************************/
        public String Logaritmo_natural () {
          String temp;
          RealInterval A = new RealInterval(1.0,Math.E);
          A= IAMath.ln(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n21)Logaritmo \n\tA=[1.0;e]\n\tln(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/**************************************************************************/
        public String Logaritmo_2 () {
          String temp;
          RealInterval A = new RealInterval(1.0,4.0);
          A= IAMath.log2(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n22)Logaritmo de base 2\n\tA=[1.0;4.0]\n\tlog2(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/**************************************************************************/
        public String Logaritmo_10 () {
          String temp;
          RealInterval A = new RealInterval(2.0,3.0);
          A= IAMath.log10(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n23)Logaritmo de base 10\n\tA=[2.0;3.0]\n\tlog2(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/**************************************************************************/
        public String Seno () {
          String temp;
          RealInterval A = new RealInterval(0.0,1.0);
          A= IAMath.sin(A);
          temp= "------------------------------------------------------------";
          temp= temp + "\nMODULO-5 FUNCOES TRIGONOMETRICAS  TESTE - JAVA-XSC";
          temp = temp+"\n------------------------------------------------------------";
          temp =temp +"\n24)Seno\n\tA=[0,1]\n\tsin(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(1.0,9.0);
          A= IAMath.sin(A);
```

```
          temp =temp +"\n\n\tA=[1.0;9.0]\n\tsin(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(2.0,6.0);
          A= IAMath.sin(A);
          temp =temp +"\n\n\tA=[2.0;6.0]\n\tsin(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(4.0,5.0);
          A= IAMath.sin(A);
          temp =temp +"\n\n\tA=[4.0;5.0]\n\tsin(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/***************************************************************************/
        public String Co_seno () {
          String temp;
          RealInterval A = new RealInterval(0.0,Math.PI/2);
          A= IAMath.cos(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n25)Co-seno\n\tA=[0.0;PI/2]\n\tcos(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(1.0,9.0);
          A= IAMath.cos(A);
          temp =temp +"\n\n\tA=[1;9]\n\tcos(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(2.0,5.0);
          A= IAMath.cos(A);
          temp =temp +"\n\n\tA=[2.0;5.0]\n\tcos(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(4.0,5.0);
          A= IAMath.cos(A);
          temp =temp +"\n\n\tA=[4.0;5.0]\n\tcos(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/***************************************************************************/
        public String Tangente () {
          String temp;
          RealInterval A = new RealInterval(0.0,1.0);
          A= IAMath.tan(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n26)Tangente\n\tA=[0.0;1.0]\n\ttan(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A = new RealInterval(1.0,9.0);
          A= IAMath.tan(A);
          temp =temp +"\n\n\tA=[1.0;9.0]\n\ttan(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
```

```
          A = new RealInterval(2.0,4.0);
          A= IAMath.tan(A);
          temp =temp +"\n\n\tA=[2.0;4.0]\n\ttan(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A = new RealInterval(4.0,5.0);
          A= IAMath.tan(A);
          temp =temp +"\n\n\tA=[4.0;5.0]\n\ttan(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          return(temp);
        }
/**************************************************************************/
        public String Co_tangente () {
          String temp;
          RealInterval A = new RealInterval(1.0,2.0);
          A= IAMath.cot(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n27)Co-tangente\n\tA=[1.0;2.0]\n\tcot(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A = new RealInterval(2.5,3.5);
          A= IAMath.cot(A);
          temp =temp +"\n\tB=[2.5;3.5]\n\tcot(B)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A = new RealInterval(-0.5,0.5);
          A= IAMath.cot(A);
          temp =temp +"\n\tC=[-0.5;0.5]\n\tcot(C)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";

          return(temp);
        }
/**************************************************************************/
        public String Secante () {
          String temp;
          RealInterval A = new RealInterval(-1.0,1.5);
          A= IAMath.sec(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n28)Secante\n\tA=[-1.0;1.5]\n\tsec(A)= ";
          if (A.Err!="")
```

```
          temp = temp + A.Err+"\n";
        else
          temp = temp + A +"\n";
        RealInterval B= new RealInterval(0.0,Math.PI);
        B= IAMath.sec(B);
        temp =temp +"\n\tB=[0.0;PI]\n\tsec(B)= ";
        if (B.Err!="")
          temp = temp + B.Err+"\n";
        else
          temp = temp + B +"\n";
        RealInterval C= new RealInterval(1.7,4.5);
        C= IAMath.sec(C);
        temp =temp +"\n\tC=[1.7;4.5]\n\tsec(C)= ";
        if (C.Err!="")
          temp = temp + C.Err+"\n";
        else
          temp = temp + C +"\n";
        RealInterval D= new RealInterval(3.5,6.0);
        D= IAMath.sec(D);
        temp =temp +"\n\tD=[3.5;6.0]\n\tsec(D)= ";
        if (D.Err!="")
          temp = temp + D.Err+"\n";
        else
          temp = temp + D +"\n";
        RealInterval E= new RealInterval(1.4,4.0);
        E= IAMath.sec(E);
        temp =temp +"\n\tE=[1.4;4.0]\n\tsec(E)= ";
        if (E.Err!="")
          temp = temp + E.Err+"\n";
        else
          temp = temp + E +"\n";
        RealInterval F = new RealInterval(5.5,6.0);
        F= IAMath.sec(F);
        temp =temp +"\n\tF=[5.5;6.0]\n\tsec(F)= ";
        if (F.Err!="")
          temp = temp + F.Err+"\n";
        else
          temp = temp + F +"\n";
        return(temp);
        }
/***************************************************************************/
      public String Co_secante () {
        String temp;
        RealInterval A = new RealInterval(-1.0,1.0);
        A= IAMath.csc(A);
        temp = "-------------------------------------------------------------";
        temp =temp +"\n29)Co-secante\n\tA=[-1.0;1.0]\n\tcsc(A)= ";
        if (A.Err!="")
          temp = temp + A.Err+"\n";
        else
          temp = temp + A +"\n";
        RealInterval B= new RealInterval(-5.0,-4.0);
```

```
        B= IAMath.csc(B);
        temp =temp +"\n\tB=[-5.0;-4.0]\n\tcsc(B)= ";
        if (B.Err!="")
          temp = temp + B.Err+"\n";
        else
          temp = temp + B +"\n";
        RealInterval C= new RealInterval(4.0,6.0);
        C= IAMath.csc(C);
        temp =temp +"\n\tC=[4.0;6.0]\n\tcsc(C)= ";
        if (C.Err!="")
          temp = temp + C.Err+"\n";
        else
          temp = temp + C +"\n";
        return(temp);
      }
/***************************************************************************/
      public String Arco_seno () {
        String temp;
        RealInterval A = new RealInterval(-0.5,1.0);
        A= IAMath.arcsin(A);
        temp = "-----------------------------------------------------------";
        temp =temp +"\n30)Arco_seno\n\tA=[-0.5;1.0]\n\tarcsin(A)= ";
        if (A.Err!="")
          temp = temp + A.Err+"\n";
        else
          temp = temp + A +"\n";
        RealInterval B= new RealInterval(-0.5,Math.PI/2);
        B= IAMath.arcsin(B);
        temp =temp +"\n\tB=[0.5;PI/2]\n\tarcsin(B)= ";
        if (B.Err!="")
          temp = temp + B.Err+"\n";
        else
          temp = temp + B +"\n";
        return(temp);
      }
/***************************************************************************/
      public String Arco_co_seno () {
        String temp;
        RealInterval A = new RealInterval(0.5,1.0);
        A= IAMath.arccos(A);
        temp = "-----------------------------------------------------------";
        temp =temp +"\n31)Arco co-seno\n\tA=[0.5;1.0]\n\tarccos(A)= ";
        if (A.Err!="")
          temp = temp + A.Err+"\n";
        else
          temp = temp + A +"\n";
        RealInterval B= new RealInterval(-0.5,1.5);
        B= IAMath.arccos(B);
        temp =temp +"\n\tB=[-0.5;1.5]\n\tarccos(B)= ";
        if (B.Err!="")
          temp = temp + B.Err+"\n";
        else
```

```
          temp = temp + B +"\n";
        return(temp);
      }
/*********************************************************************/
      public String Arco_tangente () {
        String temp;
        RealInterval A = new RealInterval(-1.0,5.0);
        A= IAMath.arctan(A);
        temp = "----------------------------------------------------------";
        temp =temp +"\n32)Arco-tangente\n\tA=[-1.0;5.0]\n\tarctan(A)= ";
        if (A.Err!="")
          temp = temp + A.Err+"\n";
        else
          temp = temp + A +"\n";
        return(temp);
      }
/*********************************************************************/
      public String Seno_h () {
        String temp;
        RealInterval A = new RealInterval(-Math.PI/2,Math.PI/2);
        A= IAMath.sinh(A);
        temp = "----------------------------------------------------------";
        temp =temp +"\n33)Seno hiperbolico\n\tA=[-PI/2;PI/2]\n\tsinh(A)= ";
        temp = temp + A +"\n";
        return(temp);
      }
/*********************************************************************/
      public String Co_seno_h () {
        String temp;
        RealInterval A = new RealInterval(-1,Math.PI/2);
        A= IAMath.cosh(A);
        temp = "----------------------------------------------------------";
        temp =temp +"\n34)Co-seno hiperbolico\n\tA=[-1;PI/2]\n\tcosh(A)= ";
        temp = temp + A +"\n";
        A = new RealInterval(-3,-2);
        A= IAMath.cosh(A);
        temp =temp +"\n\tB=[-3;-2]\n\tcosh(B)= ";
        temp = temp + A +"\n";
        A = new RealInterval(1,2);
        A= IAMath.cosh(A);
        temp =temp +"\n\tC=[1;2]\n\tcosh(C)= ";
        temp = temp + A +"\n";
        return(temp);
      }
/*********************************************************************/
      public String Tangente_h () {
        String temp;
        RealInterval A = new RealInterval(-1.0,1.0);
        A= IAMath.tanh(A);
        temp = "----------------------------------------------------------";
        temp =temp +"\n35)Tangente hiperbolica\n\tA=[-1.0;1.0]\n\ttanh(A)= ";
        temp = temp + A +"\n";
```

```
          return(temp);
        }
/**********************************************************************/
        public String Co_tangente_h () {
          String temp;
          RealInterval A = new RealInterval(-1,Math.PI);
          A= IAMath.coth(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n36)Co-tangente hiperbolica\n\tA=[-1.0;PI]
          \n\tcoth(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A = new RealInterval(1,Math.PI);
          A= IAMath.coth(A);
          temp =temp +"\n\tB=[1;PI]\n\tcoth(B)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A = new RealInterval(-2,-1);
          A= IAMath.coth(A);
          temp =temp +"\n\tC=[-2;-1]\n\tcoth(C)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";

          return(temp);
        }
/**********************************************************************/
        public String Secante_h () {
          String temp;
          RealInterval A = new RealInterval(-1,Math.PI/2);
          A= IAMath.sech(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n37)Secante hiperbolica\n\tA=[-1.0;PI/2]\n\tsech(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(-2,-1);
          A= IAMath.sech(A);
          temp =temp +"\n\tB=[-2.0;-1]\n\tsech(B)= ";
          temp = temp + A +"\n";
          A = new RealInterval(1,2);
          A= IAMath.sech(A);
          temp =temp +"\n\tB=[1;2]\n\tsech(B)= ";
          temp = temp + A +"\n";

          return(temp);
        }
```

```
/*********************************************************************/
        public String Co_secante_h () {
          String temp;
          RealInterval A = new RealInterval(1.0,3.0);
          A= IAMath.csch(A);
          temp = "----------------------------------------------------------";
          temp =temp +"\n38)Co-secante hiperbolica\n\tA=[1.0;3.0]\n\tcsch(A)= ";
          temp = temp + A +"\n";
          A = new RealInterval(-1.0,1.0);
          A= IAMath.csch(A);
          temp =temp +"\n\tB=[-1.0;1.0]\n\tcsch(B)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          return(temp);
        }
/*********************************************************************/
        public String Seno_h_I () {
          String temp;
          RealInterval A = new RealInterval(-Math.PI/2,Math.PI/2);
          A= IAMath.senhI(A);
          temp = "----------------------------------------------------------";
          temp =temp +"\n39)Seno hiperbolico inverso\n\tA=[-PI/2;PI/2]\n\tsenhI(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }
/*********************************************************************/
        public String Co_seno_h_I () {
          String temp;
          RealInterval A = new RealInterval(2,3);
          A= IAMath.coshI(A);
          temp = "----------------------------------------------------------";
          temp =temp +"\n40)Co-seno hiperbolico inverso\n\tA=[2;3]
          \n\tcoshI(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A= new RealInterval(-Math.PI/2,Math.PI/2);
          A= IAMath.coshI(A);
          temp =temp +"\n\tB=[-PI/2;PI/2]\n\tcoshI(B)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          return(temp);
        }
```

```
/*********************************************************************/
        public String Tangente_h_I () {
          String temp;
          RealInterval A = new RealInterval(-0.5,0.5);
          A= IAMath.tanhI(A);
          temp = "-----------------------------------------------------------";
          temp =temp +"\n41)Tangente hiperbolica inversa\n\tA=[-1/2;1/2]
          \n\ttanhI(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          return(temp);
        }
/*********************************************************************/
        public String Co_tangente_h_I () {
          String temp;
          RealInterval A = new RealInterval(-3.0,-2.0);
          A= IAMath.cothI(A);
          temp = "-----------------------------------------------------------";
          temp =temp +"\n42)Co-tangente hiperbolica inversa\n\tA=[-3.0;-2.0]
          \n\tcothI(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A= new RealInterval(-1.0,1.0);
          A= IAMath.cothI(A);
          temp =temp +"\n\tB=[-1.0;1.0]\n\tcothI(B)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          return(temp);
        }
/*********************************************************************/
        public String Secante_h_I () {
          String temp;
          RealInterval A = new RealInterval(0.5,1.0);
          A= IAMath.sechI(A);
          temp = "-----------------------------------------------------------";
          temp =temp +"\n43)Secante hiperbolica inversa\n\tA=[1/2;1.0]\
          n\tsechI(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A= new RealInterval(-2,-1);
          A= IAMath.sechI(A);
          temp =temp +"\n\tB=[-2.0;-1.0]\n\tsechI(B)= ";
          if (A.Err!="")
```

```
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A= new RealInterval(10,20);
          A= IAMath.sechI(A);
          temp =temp +"\n\tC=[10;20]\n\tsechI(C)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A= new RealInterval(0.3,0.5);
          A= IAMath.sechI(A);
          temp =temp +"\n\tD=[0.3;0.5]\n\tsechI(D)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          return(temp);
        }
/*************************************************************************/
        public String Co_secante_h_I () {
          String temp;
          RealInterval A = new RealInterval(1.0,Math.PI);
          A= IAMath.cschI(A);
          temp = "------------------------------------------------------------";
          temp =temp +"\n44)Co-secante hiperbolica inversa\n\tA=[1.0;PI]\n\tcschI(A)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          A= new RealInterval(-Math.PI/2,3 * Math.PI/2);
          A= IAMath.cschI(A);
          temp =temp +"\n\tB=[-PI/2;3PI/2]\n\tcschI(B)= ";
          if (A.Err!="")
            temp = temp + A.Err+"\n";
          else
            temp = temp + A +"\n";
          return(temp);
        }
/*************************************************************************/
       public String Inteirop_intervalo() {
          String temp=   "------------------------------------------------------------------";
          temp= temp + "\nMODULO-6 FUNCOES DE TRANSFERENCIA E CONSTANTES - TESTE - JAVA-XSC";
          temp = temp +"\n-----------------------------------------------------------------";
          temp =temp + "\n45)Inteiro para Intervalo\n\ta=3 \n\tint_interv(a)= ";
          int a = 3;
          RealInterval A= new RealInterval();
          A = IAMath.int_interv(a);
          temp = temp + A+ "\n";
          return(temp);
       }
```

```
/**********************************************************************/
       public String Realp_intervalo() {
          String temp = "----------------------------------------------------------";
          temp =temp + "\n46)Real para Intervalo\n\ta=3.0 \n\treal_interv(a)= ";
          double a = 3.0;
          RealInterval A= new RealInterval();
          A = IAMath.real_interv(a);
          temp = temp + A + "\n";
          return(temp);
       }
/**********************************************************************/
       public String Extremos_inferior_e_superior() {
          String temp = "----------------------------------------------------------";
          temp =temp + "\n47)Extremo inferior e 48)Extremo superior
                \n\tA=[1.0;4.0]\n";
          double a,b;
          RealInterval A = new RealInterval(1,4);
          a = IAMath.inf(A);
          b = IAMath.sup(A);
          temp = temp + "\n\tinf(A) = " + a;
          temp = temp + "\n\tsup(A) = " + b+"\n";
          return(temp);
       }
/**********************************************************************/
       public String Constantes_pi() {
          String temp = "----------------------------------------------------------\n";
          temp =temp + "Constantes PI \n49)PI 50)PI/n 51)1/PI, n=2 \n";
          int i = 2;
          RealInterval A;
          A = IAMath.PI();
          temp = temp + "\n\tPI() = " + A;
          A = IAMath.pi_n(2);
          temp = temp + "\n\tSeja n=2 PI/n= " + A;
          A = IAMath.A();
          temp = temp + "\n\t1/PI= " + A +"\n";
          return(temp);
       }
/**********************************************************************/
       public String Inflacionamento() {
          String temp;
          RealInterval A = new RealInterval(3.0,4.0);
          A= IAMath.blow(A);
          temp = "----------------------------------------------------------";
          temp =temp +"\n52)Inflacionamento\n\tA=[3.0;4.0]\n\tblow(A)= ";
          temp = temp + A +"\n";
          return(temp);
        }

}
```

Bibliografia

[Aci91] Benedito Melo Acióly, *Fundamentação Computacional da Matemática Intervalar.* Tese de doutorado, Porto Alegre: CPGCC da UFRGS, 1991, 253 p.

[Bar83] Robert G. Bartle, *Elementos de análise real*; tradução de Alfredo A. de Farias. - Rio de Janeiro : Campus, 1983.

[BeA97] Benjamín Callejas Bedregal and Benedito Melo Acióly, *Computability on the Interval Space: A Domain Approach.* Proceeding of XXIII Conferencia Latinoamericana de Informática - CLEI'97, Tomo I, pages 107-116, Valparaíso-Chile, 10-15, november 1997.

[CGI99] Marc. C. Chan, Steven W. Griffith e Anthony F. Iasi. *JAVA 1001 dicas de programação*, tradução, Editora Makron books, São Paulo, 1999.

[Cra76] F. Crary. *The AUGMENT precompiler.* Technical Report 1470, MRC, University of Wisconsin, Madison, 1976.

[Ely93] J. S. Ely. The VPI software package for variable precision interval arithmetic. *Interval Computations,* 1993(2):135-153, 1993.

[FGK+92] C. Falco-Korn, S. Gutzwiller, S. Künig and Ch. Ullrich. Modula-SC, motivation, language definition and implementation. In E. Kaucher, S. Markov, and G. Mayer, editors, *Computer Arithmetic - Scientific Computation and Mathematical Modelling*, IMACS Annals on Computing and Applied Math. 12, pages 161-180, Basel, 1992. J. C. Balzer AG.

[HHD97] Charles Leandro Höher, Carlos Amaral Hölbig, Tiarajú Asmuz Diverio. *Programando em PASCAL-XSC.* 1ª. edição, Instituto de Informática da UFRGS : SAGRA-Luzzatto. 1997.

[HNR93] R. Hammer, M. Neaga, and D. Ratz. PASCAL-XSC, New concepts for scientific Computabity and numerical data processing. In E. Adams and U. Kulish, editors, *Scientific Computing with Automatic Result Verification*, pages 15-44, New York, etc., 1993. Academic Press.

[JaW86] P. Jansen and P. Weidner. *High-accuracy arithmetic software-some tests of the ACRITH problem solving routines.* ACM Trans. Math. Software, 12(1):62-71,1986.

[JeS84] J. P. Jeter and B. D. Shriver. *Variable precision and interval arithmetic: A portable enhancemente to FORTRAN.* Adv. Eng. Software, 6(1):45-50, 1984.

[Jul91] H.-P.Jüllig. *Algorithmen mit Ergebnisverication mit C++/2.0.* Technical report, Techical University Hamburg-Harburg, 1991.

[Kea95] R. Baker Kearfott. *Usage notes for INTLIB and Fortran 90 access to it.* Department of Mathematics, University of Southwestern Louisiana, Lafayette, USA.

[Kea96] R. Baker Kearfott. *Rigorous Global Search: Continuos Problems.* Kluwer Academic Publishers, 1996. USA.

[KKW+93] R. Klatte, U. Kulisch, A. Wiethoff, C. Lawo, and M. Rauch. *C-XSC; A C++ Class Library for Extended Scientific Computing.* Springer-Verlag, New York, 1993.

[Kok90] J. Kok. The embedding of accurate arithmetic in Ada. In P. J. L. Wallis, editor, *Improving Floating Point Programming*, pages 99-120, New York, 1990. Wiley.

[Kra93] W. Krämer. Multiple precision Computabity s with result verification. In E. Adams and U. Kulisch, editors, *Scientific Computing with Automatic Result Verification*, Mathematics in Science and Engineering, volume 189, pages 325-256, New York, 1993. Academic Press.

[KuM81] U. W. Kulish and W.L Miranker. *Computer arithmetic in theory and pratice* 1981.

[KuM86] U. W. Kulish and W.L Miranker. *The arithmetic of the digital computer: A new approach.* SIAM Rev., 1(28):1-40, March 1986.

[Lec93] A. Leclerc. *Parallel interval global optimization in C++.* Interval Computations, 1993(3):148-163, 1993.

[LeC99] Laura Lemay, Rogers Candenhead. *Aprenda em 21 dias JAVA 1.2* tradução, editora Campus, Rio de Janeiro, 1999

[Lis97] Úrsula Adriane Lisbôa. *Núcleo de Aritmética de Alta Exatidão da Biblioteca Intervalar libavi.a* Dissertação de mestrado, Porto Alegre: CPGCC da UFRGS, 1997, 121 p.

[Lyr99] Aarão Lyra. *Computabilidade no Espaço dos Intervalos Reais: Um Modelo BBS Intervalar.* Dissertação de mestrado, Natal: DIMAP-UFRN, 126 p.

[MLF87] Magali, Leônidas, Frederico Ferreira Campos Filho. *Cálculo Numérico com aplicações*, Editora Harbara, 1987.

[Moo66] Ramon E. Moore. *Interval Analysis.* Englewood Cliffs: Prentice-Hall, 1966.

[Moo79] Ramon E. Moore. *Methods and Applications of Interval Analysis.* SIAM, Philadelphia, 1979. 190p.

[MuF82] Mustafa A. Munem e David J. Foulis. *Cálculo.* tradução, Editora LTC-Livros Técnicos e Científicos Editora S.A., 1982.

[ODC97] Paulo Werlang de Oliveira, Tiarajú Asmuz Diverio e Dalcidio Moraes Claudio. *Fundamentos da Matemática Intervalar.* Instituto de Informática da UFRGS : SAGRA-Luzzatto. 1997.

[Ral87] L. B. Rall. *An introduction to the scientific computing language Pascal-SC.* Computers and Mathematics with Applications, 14(1):53-59, 1987.

[Roc96] Helder L. S. Rocha, *JAVA*, 1996, cite contendo tutorias e artigos sobre JAVA publicadas na revista LAN times, http://www.dsc.ufpb.br/ helder/java/index.html.

[Rum85] S. M. Rump. ACRITH - high accuracy arithmetic subroutine library. In B. Buchberger, editor, EUROCAL '85: *European Conference on Computer Algebra*, New York, 1985. Springer-Verlag.

[Sag97] Rafael Linden Sagula. *Avaliação de Desempenho de Bibliotecas Intervalares.* Trabalho de diplomação, Porto Alegre: Instituto de Infomática, 1997, 52p.

[San99] Regivan Hugo Nunes Santiago. *Teoria das Equações Intervalares Locais.* Tese de doutorado Recife, DI-UFPE; 1999.

[Spi75] Michael Spivak. *Cálculo Infinitesimal*, Editora Reverté, S.A., Barcelona, 1975.

[Sun58] T. Sunaga. *Theory of an interval algebra and its applications to numerical analysis.* RAAG Memoirs, 2:29-46,1958.

[UMC83] University of Minnesota Computer Center. *M77 Reference Manual*: 1977 Standards Version Edition 1. University of Minnesota, 1983.

[Wal93] W. V. Walter. *FORTRAN-XSC: A portable Fortran90 module library for accurate and reliable scientific computing.* Computing (Suppl.), 9:265-286,1993.

[Wie96] A. Wiethoff. *C-XSC: A C++ Class Library for Extended Scientific Computing.* Institute for Applied Mathematics, University of Karlsruhe, Karlsruhe, Germany, July 1996, http://www.uni-Karlsruhe.de/ iam/lehrstuh12-e.html.