JAVA-XSC: Estado da arte

Benjamín Callejas Bedregal
Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada
Natal-RN, Brasil
bedregal@dimap.ufrn.br

José E. Montenegro Dutra
Universidade Federal do Rio Grande do Norte
Departamento de Ciências Exatas e Aplicada
Caicó-RN, Brasil.
eneas@ufrnet.br

Abstract

The Language E**x**tensions **S**cientific **C**omputation, XSC in short, were introduced in order to provide an appropriate environment programming to support scientific and interval computing. This class of languages provide important tools to development of numerical software. For example, provide an automatic control of numerical error, libraries with the main mathematical routines, dynamic arrays, data type not existent in traditional programming languages, such as complex numbers, real interval, etc. beyond of several other characteristic.

The present work has as main goal to present the basic characteristic of our proposal of XSC library for JAVA programming language.

Palavras Chaves: Matemática intervalar, linguagens XSC, erros numéricos, JAVA

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 mais poderosa e precisa que a aritmética empregada na maioria das linguagens modernas de programação.

A aritmética intervalar proposta por Moore em 1959 [14], trata com dados na forma de intervalos numéricos e tem por objetivo automatizar a análise de erro computacional. Serve para controlar o erro de arredondamento e para representar dados inexatos, e 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 aritmética de alta exatidão que é uma qualidade necessária nos ambientes computacionais usados na resolução de problemas da computação científica e das engenharias. Ela permite que o resultado calculado nas operações aritméticas, mesmo considerando 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 softwares numéricos modernos e para aplicações científicas tais como: controle de arredondamento, tipos de dados com exatidão após a vírgula; 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 os números complexos, intervalos reais, matrizes intervalares, etc.; 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 tem-se mostrado ideal para aplicações na rede, tornando-se de fato a mais bem aceita e a mais usada pelos desenvolvedores de produtos e aplicações na internet. Trata-se de uma linguagem de programação orientada a objetos, independente da plataforma, que está revolucionando a World Wide Web. 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 no desenvolvimento de sistemas distribuídos para aplicações computacionais e científicas na solução de problemas físicos, químicos e das engenharias que necessitam de alta exatidão. Assim, resulta razoável adicionar computações intervalares a JAVA

No inicio de 1999 a SUN tornou público, através da homepage da comunidade científica internacional na área de computação intervalar (http://cs.utep.edu/interval-comp/java.html) que está realizando esforços internos para adicionar intervalos a JAVA e pede sugestões ou propostas à comunidade intervalar neste sentido. Isto nos motivou a desenvolvermos, através da dissertação de mestrado [1], uma biblioteca que implementasse os métodos intervalares (reais) usualmente disponíveis nas linguagens XSC assim como alguns testes básicos, nascendo desta forma o embrião do JAVA-XSC. Posteriormente, com a ajuda de outros alunos, foi-se melhorando a qualidade das implementações e estenderam-se e incorporaram-se novas bibliotecas para lidar com números complexos, complexos intervalares, matrizes e matrizes intervalares (reais e complexas), foi também implementada uma calculadora intervalar, desenvolveram-se as API dos métodos e lançou-se uma página do projeto (http://www.dimap.ufrn.br/~java-xsc) a qual desafortunadamente encontra-se desatualizada. O presente trabalho visa apresentar o estado da arte em que se encontra o projeto JAVA-XSC.

2 Matemática Intervalar

Os números racionais, do ponto de vista algébrico, 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 — e.g. $3.14x + 1.2 = 2.3$ — 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. No entanto, esse não é o principal problema dessa abordagem. O problema maior reside no fato de que esse erro de aproximação não obedece 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 que nem sempre são viáveis.

A matemática intervalar busca resolver problemas que se concentram fundamentalmente em dois aspectos: na criação de um modelo computacional que reflita com fidelidade o controle e análise dos erros que ocorrem no processo computacional e na escolha de técnicas de programação adequadas para desenvolvimento de software científico buscando minimizar os erros nos resultados. Assim, os algoritmos trabalham sobre intervalos em vez de números racionais, tendo como resultados da computação, intervalos que contém as soluções reais desejadas, cujas amplitudes dão uma medida da qualidade dos resultados e, portanto, o tamanho do erro na aproximação (intervalo) de cada número real.

Um *intervalo real* $[r, s]$ foi definido como um subconjunto de $\mathbb{R}$ dado por: $[r, s] = \{x \in \mathbb{R} \ / \ r \leq x \geq s\}$, e o conjunto de todos os intervalos reais é denotado por $\mathbb{I}(\mathbb{R})$, isto é:

$$\mathbb{I}(\mathbb{R}) = \{[r, s] \ / \ r, s \in \mathbb{R} \text{ e } r \leq s\}.$$

Embora um intervalo seja um conjunto de números reais, ele pode ser tratado como um par de números reais, uma vez que as operações aritméticas entre intervalos, assim como a maioria das funções usuais, podem ser caracterizadas em termos dos extremos. Por exemplo, no caso da adição e da multiplicação intervalar:

$$
\begin{aligned}
[a,b]+[c,d] &= \{r \in \mathbb{R} \,/\, r = s+t \text{ para algum } s \in [a,b] \text{ e } t \in [c,d]\} \\
&= [a+c, b+d]
\end{aligned}
$$

$$
\begin{aligned}
[a,b]\cdot[c,d] &= \{r \in \mathbb{R} \,/\, r = s\cdot t \text{ para algum } s \in [a,b] \text{ e } t \in [c,d]\} \\
&= [min\{a\cdot c, b\cdot c, a\cdot d, b\cdot d\}, max\{a\cdot c, b\cdot c, a\cdot d, b\cdot d\}]
\end{aligned}
$$

Observe que a finalidade principal em se considerar um intervalo seria o fato deste intervalo constituir uma aproximação dos reais que ele contém. Mas, também pode ser encarado como uma generalização do conjunto dos números reais, uma vez que cada número real r pode ser identificado com o intervalo $[r, r]$, chamado *intervalo degenerado*. Neste sentido, o conjunto $\mathbb{I}(\mathbb{R})$ constitui uma estrutura algébrica que generaliza a estrutura dos reais [22, 14, 15, 17].

3 Linguagens XSC

A velocidade de computadores digitais vem aumentando com o avanço da tecnologia, isto porque historicamente a ênfase da computação estava tradicionalmente direcionada à velocidade, porém uma vez que já temos atingido um nível aceitável de velocidade podemos dar agora uma ênfase maior à precisão e confiança de resultados. A Matemática Intervalar desenvolveu 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 sejam satisfatórios e ferramentas de programação poderosas que não estão geralmente disponíveis, esta exigência fez com que fosse necessário a criação das linguagens com extensões para computações científicas.

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 estão disponíveis em FORTRAN, PASCAL e C++. Rotinas que resolveram problemas com verificação automática de resultado foram desenvolvidos para muitos problemas padrão da análise numérica linear, sistemas de equações não lineares, equações diferenciais, integrais, etc, assim como rotinas para um grande número de aplicações na engenharia e nas ciências naturais.

Os pacotes de matemática intervalar podem ser classificados da seguinte forma [9]:

1. **Primeiros pacotes:** A maioria era em FORTRAN ou baseado em ALGOL, e foram construídos com o princípio da pre-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:** Criados no modelo "SC ", estas são linguagens portáteis fundados principalmente no principio de sobrecarga de operadores.

4. **Outros modernos pacotes:** Estes incluem INTLIB_90 e vários pacotes em C++, e outras linguagens que suportam sobrecarga de operadores.

5. **Pacotes para manipuladores simbólicos:** Estes incluem bibliotecas para matemática intervalar em Mathematica e MAPLE.

O surgimento na década dos 80 e 90 de linguagens de programação como C++, FORTRAN90 e JAVA facilitaram a portabilidade das linguagens, isto é a capacidade que tem uma linguagem de se adaptar a diversos equipamentos e softwares. Baixo a liderança do Prof. Dr. Kulisch da Universidade de Karlsruhe, começou uma segunda geração de linguagens, as linguagens XSC.

As linguagens XSC têm como característica principal a maximização da precisão e operações intervalares. A portabilidade das linguagens XSC é herdada da portabilidade das linguagens em que são implementadas. Por exemplo, o compilador PASCAL-XSC [2, 6] foi escrito originalmente em C e adere às especificações do PASCAL-SC. Estão disponíveis em PASCAL-XSC um pacote de ferramentas numérica bem-documentado e com rotinas para avaliação de polinômios, diferenciação automática, raízes únicas de equações, precisão, verificação de sistemas de equações lineares, sistemas de equações não-linear, otimização global, etc.

O C-XSC [11, 24] é uma extensão da linguagem de programação C++ que consiste em um conjunto de classes de bibliotecas em C++. C-XSC é um das várias implementações de matemática intervalar baseados em C++.

FORTRAN-XSC [23] e INTLIB foram ambos baseado no FORTRAN90. Porém, a ênfase no projeto do FORTRAN-XSC estava em um produto de boa precisão e em módulos para aritmética de precisão, como também para intervalos de vetor e operações de matrizes.

O objetivo principal do projeto JAVA-XSC é desenvolver para a linguagem de programação JAVA uma extensão cuja característica principal seja facilitar aplicações da computação científica. Assim, o JAVA-XSC, isto é a extensão de JAVA com bibliotecas para os tipos de dados intervalos reais, números complexos, intervalos complexos, matrizes e vetores intervalares que foi feita com uma garantia de corretude de seus métodos e, no possível, de otimalidade [4, 20].

São várias as vantagens de acrescentar bibliotecas intervalares à linguagem de programação JAVA. Uma especialmente importante é ser uma linguagem de programação independente de plataforma. Processando o mesmo programa com uma linguagem de programação dependente da plataforma, em diferentes plataformas com distintas precisões e tipos de arredondamentos, freqüentemente conduziram 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. Isto poderia tornar um sistema distribuído inviável.

4 A Biblioteca IntMath-JAVA

A biblioteca para JAVA que implementa o tipo de dados intervalos reais, junto com todas as funções usuais da matemática intervalar, denominado por nós como IntMath-JAVA, foi desenvolvida como um pacote, que é a forma em JAVA de se agrupar classes e interfaces relacionadas. O nome da biblioteca IntMath-JAVA é o acrônimo de *Interval Mathematics for JAVA*, baseado na classe que trata os conjuntos dos números reais que é a JAVA.lang.Math. Esta biblioteca contém todas as funções intervalares implementadas no módulo I_ARI do PASCAL-XSC, mais algumas outras.

A metodologia usada para desenvolver esta biblioteca foi identificar as funções intervalares que seriam implementadas, dar uma caracterização matemática delas em termo dos extremos de seus argumentos e de forma ótima (a melhor representação intervalar usando a terminologia de [20]), e escrever o código JAVA para estas funções baseados fortemente em estas caracterizações. Esta biblioteca contém três classes básicas:

4.1 Classe IAException

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

4.2 Classe RealInterval

Esta classe representa o intervalo de números reais. Ou seja aqui se define a estrutura básica do intervalo. Esta classe contém as seguintes variáveis, construtores e métodos:

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 uma única variável double.
3. RealInterval(): Constrói um intervalo contendo como extremo inferior - infinito e + infinito, que é uma forma de criar e preencher este intervalo.

Métodos da Classe IAMath			
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

Table 1: Operações aritméticas, conjuntistas e geométricas.

Método:

Na classe RealInterval temos o método toString que faz o layout do intervalo colocando os colchetes e o ponto e virgula no intervalo para que seja impresso desta forma.

4.3 Classe IAMath

Esta Classe contém 52 métodos para executar operações e funções intervalares, os quais são agrupados 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 [13], 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 das operações intersecção, união e 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 1, 2, 3 e 4 informam sobre os métodos desenvolvidos dentro da classe IAMath.

Métodos da Classe IAMath			
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

Table 2: Funções elementares.

Métodos da Classe IAMath			
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

Table 3: Funções trigonométricas.

Métodos da Classe IAMath			
Módulo 6-Funções de transferência e constantes			
N^0	Método	Tipo do Resultado	Nome genérico
45.	int_interv	RealInterval	Converte inteiro → intervalo
46.	real_interv	RealInterval	Converte real → 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

Table 4: Funções transferência e constantes.

4.4 Corretude dos Métodos

Quando definimos uma representação intervalar de uma função real, usualmente o fazemos usando a aplicação da função real nos valores extremos do intervalos e eventualmente algumas constantes. Os métodos usados em bibliotecas XSC também são definidos desta maneira. Porém, uma vez que o modo de arredondamento do JAVA é o "mais próximo´´, computações intervalares que implementam em uma maneira natural (sem se preocupar com arredondamentos) as versões intervalares ótimas de uma função real podem resultar em intervalos que não tenham a propriedade de corretude, isto é que não garantam que o resultado ideal esteja no intervalo de saída [4, 20]. Por exemplo, a versão intervalar ótima da função real $f(x) = \frac{x}{3}$ é $F(X) = X/3$. O resultado ideal de uma implementação de F quando o intervalo $X = [2, 4]$ é o intervalo $[0.\overline{6}, 1.\overline{3}]$, no entanto calculando os extremos (2/3 e 4/3) em JAVA, o resultado seria $[0.6666666666666667, 1.3333333333333333]$ o que não é correto no sentido de [4, 20].

Para garantir que o intervalo de saída de uma computação intervalar contenha a saída ideal usualmente são utilizados dois métodos:

1. Inflacionamento: O intervalo de saída de cada operação que possa vir a ter este problema é inflacionado, isto é, o extremo esquerdo é diminuído até o ponto flutuante mais próximo pela esquerda dele e o da direita é aumentado para o ponto flutuante mais próximo pela direita dele.

2. Arredondamento direcionado: a cada operação são usados arredondamentos para baixo no caso do extremo esquerdo e arredondamento para acima no caso do extremo direito. intervalarmente de uma maneira natural

O inflacionamento é usado nas bibliotecas XSC: INTLIB [8] e **libivi.a** [13] enquanto o arredondamento direcionado é usado nas bibliotecas XSC: PASCAL-XSC [10] and C++-XSC [11, 5]. O inflacionamento de um intervalo é feito via duas funções, uma retorna o ponto flutuante imediatamente inferior ao extremo esquerdo do intervalo enquanto a outra retorna o ponto flutuante imediatamente superior ao extremo direito do intervalo. Claramente esta solução é mais simples que a de arredondamento direcionado, mas tem a desvantagem de perda de precisão retornando as vezes intervalos excessivamente grandes. Então porque não usar sempre arredondamento direcionado? simplesmente porque nem toda linguagem permite manipular a forma de arredondamento a usar em cada operação. Um exemplo de linguagem que não permite arredondamento direcionado é o JAVA. Isto motivo inicialmente a usarmos inflacionamento. Mas, já está em fase de desenvolvimento a implementação a baixo nível do arredondamento direcionado [21].

5 Outras Bibliotecas

A inclusão das bibliotecas intervalares da seção anterior, ainda não são suficientes para chamar esta versão de JAVA de JAVA-XSC. Para isso, necessitamos considerar, também, bibliotecas que lidem com números complexos, intervalos complexos e matrizes intervalares.

Os métodos da classe ComplexInterval são análogos aos métodos da classe Complex.

As outras classes de matrizes são análogas à classe MatrixInterval.

Métodos da Classe Complex			
N^0	Método	Tipo do Resultado	Descrição
1	add	Complex	Adiciona dois números complexos
2	conjugate	Complex	Retorna a conjugada de um número complexo
3	div	Complex	Divide um número complexo por outro número complexo
4	getReal	Real	Retorna a parte real de um número complexo
5	minus	Complex	Subtrai de um número complexo outro número complexo
¨6	mod	Real	retorna o módulo de um número complexo
7	mult	Complex	Multiplica dois números complexos
8	multEscalar	Complex	Multiplica um escalar por um número complexo
9	setImage	void	Modifica a parte imaginária de um número complexo
10	setReal	void	Modifica a parte real de um número compelxo
11	toString	String	Transforma um número complexo num string

Methods of MatrixInterval Class **Métodos da Classe MatrixInterval**			
N^0	Método	Tipo do Resultado	Descrição
1	add	MatrixInterval	Adiciona duas matrizes de intervalos reais
2	dist	Real	Retorna a distância entre duas matrizes de intervalos reais
3	exhib	void	Mostra na tela uma matriz de intervalos reais
4	getInterval	RealInterval	Retorna o intervalo real que esta em uma dada posição de uma matriz de intervalos reais
5	getM	Integer	Retorna a quantidade de linhas de uma matriz de intervalos reais
6	getN	Integer	Retorna a quantidade de colunas de uma matriz de intervalos reais
7	inclussion	Boolean	verifica se uma matriz de intervalos reais está incluida em outra
8	inf	RealInterval	Retorna o ínfimo de uma matriz de intervalos reais
9	intersection	MatrixInterval	Retorna a interseção de duas matrizes de intervalos reais
10	matrixEquals	Boolean	Verifica se duas matrizes de intervalos reais são iguais
11	matrixDiameter	MatrixReal	Retorna uma matriz real com os diámetros dos intervalos na matriz de intervalos reais
12	matrixMiddlePoint	MatrixReal	Retorna a matriz real com os pontos medios dos intervalos da matriz de intervalos reais
13	matrixModule	MatrixReal	Retorna a matriz módulo de uma matriz de intervalos reais
14	minus	MatrixInterval	Subtrai de uma matriz de intervalos reais outra matriz de intervalos reais
15	mult	MatrixInterval	Multiplica duas matrizes de intervalos reais
16	multInterval	MatrixInterval	Multiplica um intervalo com uma matriz de intervalos reais
17	put_in	void	Coloca um mintervalo numa determinada posição da matriz de intervalos reais
18	sup	RealInterval	Retorna o supremo intervalar de uma matriz de intervalos reais
19	Union	MatrixInterval	Retorna a matriz com a união dos intervalos que estão na mesma posição em duas matrizes de intervalos reais

6 Método de Newton Interval em JAVA-XSC

Com é bem conhecido pelos pesquisadores em matemática intervalar, existem diversas extensões intervalares do tradicional método de Newton, veja por exemplo [16, 9, 3]. Aqui a modo de ilustrar como pode ser usada a biblioteca JAVA-XSC apresentaremos a segunda variação deste método apresentada em [1].

Esta variação consiste em tomar a inclusão intervalar da derivada, sendo atualizada em cada iteração 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

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

onde $x_0 = med(X_0)$.

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 $f(x)$. Definimos o **operador newtoniano intervalar** em duas partes:

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. $N_2(X) = \mathcal{I}(n(\underline{X}), n(\overline{X})) - \dfrac{\mathcal{I}(f(n_i(X)); f(n(X)))}{F'(X)}$.

$\mathcal{I}(a, b) = [\min\{a, b\}, \max\{a, b\}]$ e $\underline{X}$ e $\overline{X}$ são os extremos esquerdo e direito do intervalo X, respectivamente.

Finalmente, construímos a seqüência intervalar recursiva

$$X_{k+1} = X_k \cap N_2(X_k)$$

começando com $X_0 = [a; b]$.

Exemplo: 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. $N_2(X) = \mathcal{I}(n(\underline{X}), n(\overline{X})) - \dfrac{\mathcal{I}(n(\underline{X})^2 - 2, n(\overline{X})^2 - 2)}{[2; 2] \cdot X}$

Finalmente, construímos a seqüência intervalar recursiva

$$X_{k+1} = X_k \cap N_2(X_k)$$

a partir de $X_0 = [1; 2]$.

A tabela 5 apresenta os valores dessa seqüência calculados usando o programa Java-XSC a seguir.

O programa Newton2

```
import ia_math.*; public class  Newton2
{
   public static void main (String arguments[ ])
   {
```

k	X_k
0	$[1.0; 2.0]$
1	$[1.375; 1.4375]$
2	$[1.414141414141414; 1.4142512077294688]$
3	$[1.4142135622631408; 1.4142135624280355]$
4	$[1.4142135623730950; 1.4142135623730952]$
5	$[1.4142135623730950; 1.4142135623730952]$

Table 5: Valores da seqüência X_k obtidos pelo programa Newton2 em JAVA-XSC.

```
        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("-----------------------------------------------------");
    }
}
```

Logo, o valor da raiz da função $f(x) = x^2 - 2$, no intervalo $[1, 2]$, é um valor real $x_* \in [1.4142135623730950; 1.4142135623730952]$. De fato $x_* = \sqrt{2}$.

7 Conclusão

Aqui propomos uma extensão da linguagem de programação JAVA, a qual considera algumas bibliotecas intervalares, o qual é de fundamental importância para as computações científicas com uma garantia do erro. Desenvolvemos alguns métodos elementares, rotinas e funções padrão. Como este intuito, primeiro fizemos a definição precisa de cada função intervalar considerada na nossa biblioteca, em termos dos extremos dos intervalos envolvidos[1] e depois implementamos em JAVA cada uma dessas funções seguindo, o mais fiel possível, essas definições.

São várias as vantagens como vimos acima, de acrescentar computações de intervalo a JAVA. Uma característica específica de JAVA que faz esta extensão especialmente importante é que ela é uma linguagem independente de plataforma. Rodando o mesmo programa em plataformas diferentes com diferentes precisões do computador, etc., freqüentemente conduz a resultados diferentes. Se nós 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 bug de certo tipo, ou há pouco estabilidade numérica. Assim, um potencial uso seria em sistemas distribuídos em que alguns módulos necessitem de computações numéricas.

A linguagem JAVA tem o suporte da empresa Sun Microsystems, que a distribui livremente. De fato, foi a própria SUN que tornou público, através da homepage da comunidade científica internacional na área de computação intervalar (http://cs.utep.edu/interval-comp/java.html) que está realizando esforços internos para incorporar intervalos a JAVA e pede sugestões ou propostas à comunidade intervalar neste sentido.

[1] Para a maioria das funções intervalares usadas na biblioteca não se dispõe, na literatura tradicional de teoria dos intervalos, uma caracterização delas em termos dos extremos de seus argumentos.

Esperamos que este trabalho satisfaça as expectativas da comunidade internacional de intervalos e que contribua a concretizar os anseios da SUN Microsystem de desenvolver a linguagem JAVA-XSC.

Como trabalhos futuros temos nos proposto implementar novas bibliotecas para pontos flutuantes nas quais possamos administrar o tipo de arredondamento desejado. Isto é fundamental para termos como resultados intervalos que além de conter o resultado ideal também sejam o mais estreitos possíveis.

Também pretendemos adicionar dinamicidade na precisão dos resultados numéricos, ou seja definir novas bibliotecas intervalares que se utilizem do conceito de pontos flutuantes dinâmicos e que já foram implementados em JAVA pelo grupo [18].

Além disso, pretendemos testar e verificar as bibliotecas intervalares, usando alguns métodos intervalares, como por exemplo o método de Runge-Kutta Intervalar em suas diversas versões. Também pretendemos fazer uma comparação, em termos de abrangência, qualidade dos resultados e eficiência da nossa proposta com outras linguagens XSC, e em diversas plataformas (tais como UNIX, WINDOW NT e AIX).

Agradecimentos

Diversos estudantes tem participado do projeto Java-XSC em algum momento, pelo que gostaria de agradecer a todos eles, em especial a Clístenes Oliveira Simonetti, Rodrigo da Câmara Varela, João Paulo de Araújo Bezerra e Daniel Carvalho Sodré Duarte.

References

[1] J.E.M. Dutra. Java-XSC: Uma biblioteca Java para computações intervalares. Dissertação de Mestrado, PPgSCC-UFRN, Natal, 2000.

[2] R. Hammer, M. Neaga, and D. Ratz. PASCAL-XSC, New concepts for scientific computation 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.

[3] E. Hansen and G.W. Walster. *Global Optimization Using Interval Analysis.*CRC, second edition, 2003.

[4] T. Hickey, Q. Ju and M.H. Van Emdem. Interval arithmetic: From principles to implementation. *Journal of the ACM*, 48(5):1038–1068, 2001.

[5] W. Hofschuster, W. Krämer, S. Wedner and A. Wiethoff. *C-XSC 2.0: A C++ class library for extended scientific computing.* Wissenschaftliches Rechnen/Softwaretechnologie, 2001 (preprint available in http://www.xsc.de).

[6] C.L. Höher, C.A. Hölbig, T.A. Diverio. *Programando em PASCAL-XSC.* 1^a edição, Instituto de Informática da UFRGS: SAGRA-Luzzatto. 1997.

[7] 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.

[8] R. Baker Kearfott. *Usage Notes for INTILIB and Fortran 90 Access to it.* Department of Mathematics, University of Southwestern Louisiana, Lafayette, USA, 1995.

[9] R. Baker Kearfott. *Rigorous Global Search: Continuos Problems.* Kluwer Academic Publishers, 1996. USA.

[10] R. Klatte, U. Kulisch, M. Neaga, D. Ratz and Ch. Ullrich. *PASCAL-XSC; Language reference with examples.* Springer-Verlag, Berlin, 1991. Version march 3, 1999 in http:
www.xsc.de.

[11] 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.

[12] W. Krämer. Multiple precision computations 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.

[13] U. A. Lisbôa. *Núcleo de Aritmética de Alta Exatidão da Biblioteca Intervalar libavi.a.* Dissertação de mestrado, Porto Alegre: CPGCC da UFRGS, 1997, 121 p.

[14] R.E. Moore. Automatic error analysis in digital computation. Technical Report LMSD84821, Lockheed Missiles and Space Co., 1959.

[15] R.E. Moore. *Methods and Applications of Interval Analysis.* SIAM, 1979.

[16] A. Neumaier. *Interval Methods for System of Equations.* Cambridge University Press, Cambridge, England, 1990.

[17] P. W. de Oliveira, T. A. Diverio e D. M. Claudio. *Fundamentos da Matemática Intervalar.* Instituto de Informática da UFRGS: SAGRA-Luzzatto. 1997.

[18] O. F. Oliveira Jr. Números Dinâmicos: Uma abordagem computacional orientada a objetos com implementações na linguagem Java. Dissertação de Mestrado, PPgSC, UFRN, Natal, 2004.

[19] 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.

[20] R.H.N. Santiago, B.R.C. Bedregal and B.M. Acióly. Formal Aspects of Correctness and Optimality of Interval Computations. *Formal Aspect of Computing*, 18:231–243, 2006.

[21] J.F.V. da Silva, B.R.C. Bedregal and R.H.N. Santiago. A Java Floating Point Library with Directed Rounding. To appear in Proceeding of SCAN'06 International Symposion, Duisburg, Germany, September 26-29, 2006.

[22] T. Sunaga. *Theory of an interval algebra and its applications to numerical analysis.* RAAG Memoirs, 2:29-46,1958.

[23] W. V. Walter. *FORTRAN-XSC: A portable FORTRAN90 module library for accurate and reliable scientific computing.* Computing (Suppl.), 9:265-286,1993.

[24] 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)