CALCULADORA INTERVALAR PARA OPERAÇÕES ARITMÉTICAS EM JAVA

RENATO VIANA FERREIRA[1], MARCILIA ANDRADE CAMPOS[2], Bruno José Torres Fernandes[3]

[1] UFPE, CIn - Centro de Informática, rvf@cin.ufpe.br
[2] UFPE, CIn - Centro de Informática
[3] UFPE, CIn - Centro de Informática

RESUMO

Sistemas computacionais atuais são incapazes de representar todos os números reais, pois estes são contínuos e densos e a representação digital é discreta. A representação destes números é realizada em computadores por meio de ponto flutuante, que provê uma representação binária capaz de armazenar uma quantidade finita de valores em sua forma exata e todos os outros valores devem ser aproximados para o valor representável mais próximo. Java se tornou a linguagem mais utilizada no mundo devido ao seu paradigma de programação, portabilidade, simplicidade e robustez. Por isso, foi desenvolvida uma extensão para computação científica para Java, chamada Java-XSC (eXtension for Scientific Computation), no intuito de solucionar o problema de erro numérico em Java. Com relação à extensão intervalar para Maple, o intpakX, a biblioteca Java-XSC apresentou ótimos resultados após testes de validação, apresentando saídas semelhantes na grande maioria dos casos e também de melhor precisão em algumas operações. Uma calculadora intervalar para Java foi implementada como uma aplicação para a biblioteca, não apenas contendo as operações aritméticas, mas também lógicas.
Palavras-Chave: Intervalos; Java; Java-xsc

INTRODUÇÃO

Um problema enfrentado pelo modelo computacional atual é a representação numérica, devido à infinitude, continuidade e densidade dos números reais. Estes números não podem ser representados discretamente em sua totalidade. A representação destes números é realizada em computadores por meio de ponto flutuante, que provê uma representação binária capaz de armazenar uma quantidade finita de valores em sua forma exata e todos os outros valores devem ser aproximados para o valor representável mais próximo.

Muitos avanços da ciência e da engenharia das últimas décadas não seriam possíveis sem o processamento de números de ponto-flutuante pelos computadores digitais. Porém, alguns resultados de computação com pontos-flutuantes parecem estranhos, mesmo para experts na área.

Para representar ponto-flutuante, a linguagem de programação Java oferece dois tipos primitivos que são float e double que representam precisão simples de 32 bits e dupla precisão de 64 bits respectivamente, de acordo com o padrão IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985 (IEEE, New York).

A implementação de ponto flutuante em Java falha em dois aspectos em relação ao padrão IEEE, (i) a ausência de suporte a flags de exceções, e (ii) não implementação dos arredondamentos direcionados. Os flags são: (i) operação inválida; (ii) overflow; (iii) divisão por zero; (iv) underflow e (v) resultado inesato. Estes flags são fundamentais para implementação completa do padrão IEEE 754, e garantiriam melhor tratamento numérico.

Java se tornou a linguagem mais utilizada no mundo devido ao seu paradigma de programação, portabilidade, simplicidade e robustez. Por isso, foi desenvolvida uma extensão para computação científica para Java, chamada Java-XSC (eXtension for Scientific Computation), no intuito de solucionar o problema de erro numérico em Java, que para a maioria dos usuários, parece irrelevante, porém para computação científica, que necessita de alta precisão, o erro deve ser controlado para evitar resultados errados.

MÉTODOS

Este projeto tem a coordenação da professora Marcília Andrade Campos (www.cin.ufpe.br/~mac) no Centro de Informática da Universidade Federal de Pernambuco e conta com a participação de quatro alunos, sendo dois de mestrado e dois da graduação em Ciência da Computação.

O primeiro passo para o desenvolvimento de uma biblioteca intervalar para Java (Java-XSC) foi o estudo de arredondamentos direcionados, tanto para cima, como para baixo. Em seguida, uma classe Rounding foi modelada e implementada contendo os arredondamentos e foi validada utilizando a biblioteca intervalar para Maple, intpakX, desenvolvida no Departamento de Matemática da Universidade de Wuppertal na Alemanha, já consolidada como biblioteca intervalar.

Após a documentação da classe Rounding, foi discutida a criação do tipo intervalo, sobre duas possibilidades: a inserção de um tipo primitivo na linguagem Java ou a criação de uma classe. Verificando-se a impossibilidade de inserção de tipos primitivos na linguagem devido a restrições técnicas e legais, foi modelada e implementada a classe Interval para representar o tipo intervalo contendo seus limites superior e inferior e a precisão desejada.

Posteriormente foi feito um estudo sobre operações intervalares, tanto unárias, como lógicas e aritméticas, estes estudos foram realizados através de revisão bibliográfica de teses e artigos relacionados ao tópico e utilização da biblioteca intpakX.

As operações unárias foram implementadas na própria classe Interval, pois se referem unicamente a um intervalo. Para a implementação das operações aritméticas foi criada a classe IntervalMath e para as operações lógicas, a classe IntervalSet.

Toda a modelagem das classes foi realizada utilizando o Rational Rose Entreprise Edition da Rational/IBM. O

desenvolvimento da biblioteca utilizou as versões 1.4.2 e 5.0 de Java e a IDE (Ambiente Integrado de Desenvolvimento) Eclipse, nas versões 2.1, 3.0 e 3.1..

 A documentação da biblioteca seguiu o padrão Javadoc definido pela Sun Microsystems, empresa responsável por Java.

RESULTADOS

Durante este projeto foi modelada, implementada, documentada e validada uma extensão para computação científica para a linguagem de programação Java. Por fim, também foi implementada uma aplicação para esta biblioteca que foi uma calculadora intervalar para operações aritméticas e lógicas.

Os arredondamentos direcionados implementados na classe Rounding, quando comparados às funções de arredondamento do intpakX, apresentaram resultados extremamente satisfatórios, obtendo saídas semelhantes em 70% dos casos de teste. A diferença nos resultados, quando ocorreu, foi sempre apresentada na última casa decimal, onde em relação à qualidade do arredondamento, ou seja, a menor distância entre o número inicial a ser arredondado e o resultado da função de arredondamento, ocorreu um relativo empate entre o intpakX e a biblioteca Java-XSC.

 A definição do tipo intervalo foi feita considerando o limite inferior e superior e também a precisão desejada para os arredondamentos, no que Java-XSC difere do intpakX, pois a precisão deste último é invariável. Os limites foram representados pelo tipo primitivo double (ponto-flutuante) e a precisão pelo tipo primitivo int (inteiro).

Sobre o tipo intervalo, modelado como uma classe chamada Interval, foram implementadas as seguintes operações unárias: (i) Comprimento; (ii) Simétrico; (iii) Recíproco; (iv) Verificação do vazio. Algumas funções aritméticas e lógicas foram implementadas na própria classe Interval.

As operações aritméticas intervalares, algumas das quais, implementadas na classe IntervalMath, foram as seguintes: (i) Igualdade (implementada na classe Interval); (ii) Adição; (iii) Subtração; (iv) Multiplicação; (v) Divisão.

As operações lógicas intervalares, algumas das quais, implementadas na classe IntervalSet, foram: (i) Pertence (implementada na classe Interval); (ii) Interseção, (iii) União, (iv) Absoluto, (v) Distância.

A comparação dos resultados de Java-XSC com o intpakX, para as mesmas entradas, foi extremamente satisfatória, tendo a biblioteca Java-XSC apresentado um controle do erro bastante similar e se mostrando mais precisa em alguns casos, principalmente na operação de divisão. As operações de distância e absoluto não foram encontradas na biblioteca intpakX, impossibilitando a comparação.

A calculadora intervalar para Java foi implementada como uma aplicação para a biblioteca e será disponibilizada na web, não apenas contendo as operações aritméticas, mas também lógicas.

DISCUSSÃO E CONCLUSÃO

Sistemas computacionais atuais são incapazes de representar todos os números reais, pois estes são contínuos e densos e a representação digital é discreta. Java, sendo uma linguagem de programação, consequentemente, sofre esse problema, além disso, a implementação de ponto flutuante para Java desprezou fatores importantes como: (i) não suportar flags de exceções, definidos pelo padrão IEEE 754, (ii) não prover os arredondamentos direcionados também definidos pelo IEEE 754 e (iii) não prover intervalos como tipo primitivo. Portanto, Java apresenta vários erros em operações com ponto-flutuante.

 Como Java, atualmente, é linguagem mais popular do mundo, devido a um grande número de vantagens oferecidas, como: simplicidade, portabilidade e robustez, tende a ser mais utilizada em aplicações de computação científica. Visando suprir essa necessidade, foi implementada uma biblioteca chamada Java-XSC (eXtension for Scientific Computation) para controlar erros numéricos de ponto-flutuante utilizando métodos da matemática intervalar.

 Com relação à extensão intervalar para Maple, o intpakX, a biblioteca Java-XSC apresentou ótimos resultados após testes de validação, apresentando saídas semelhantes na grande maioria dos casos e também de melhor precisão em algumas operações.

 A aplicação implementada para uso da biblioteca foi uma calculadora intervalar para operações aritméticas e lógicas que será disponibilizada na internet através do site do projeto, http://www.cin.ufpe.br/~javaxsc/.

 A biblioteca Java-XSC será estendida para conter, além das operações aritméticas e lógicas, as operações logarítmicas, trigonométricas e estatísticas da matemática intervalar, além das relações de ordem.

REFERÊNCIAS

[1] Moore, R. E. (1996), ?Interval Analysis? Englewood Cliffs: Prentice-Hall.

[2] Campos, M. A., (1997), ?Uma Extensão Intervalar para a Probabilidade Real? Departamento de Informática ? UFPE.

[3] Kahan, W., Darci, J., (1998), ?How Java?s Floating-Point Hurts Everyone Everywhere?, http://cch.loria.fr/documentation/IEEE754/wkahan/JAVAhurt.pdf, visitado em 9 de Junho de 2005.

[4] IEEE Computer Society (1985), “IEEE Standard for Binary Floating-Point Arithmetic”, IEEE Std 754-1985.

[5] Sun Microsystems, Inc. (2005), ?Java Technology?, http://java.sun.com/, visitado em 9 de junho de 2005