Critérios para Análise e Escolha de Ambientes Intervalares

Mauricio Dorneles Caldeira Balboni, Lucas Mendes Tortelli, Mariline Lorini,
Vinícius Signori Furlan, Alice Fonseca Finger, Aline Brum Loreto

***Resumo*—Quando se trabalha com números de ponto flutuante o resultado é apenas uma aproximação de um valor real e erros gerados por arredondamentos ou por instabilidade dos algoritmos, podem levar a resultados incorretos. Não se pode afirmar a exatidão da resposta estimada sem auxílio de uma análise de erro. Utilizando-se intervalos para representação dos números reais, é possível controlar a propagação desses erros. Para obtenção de resultados mais exatos, as implementações em computadores devem ser efetuadas através de linguagens, ambientes ou bibliotecas que tenham definido o tipo intervalo e suas operações. Como existem diversos ambientes computacionais intervalares, o presente trabalho consiste em analisar tais ambientes quanto aos critérios de qualidade de software, critérios de linguagens de programação e critérios de qualidade do intervalo solução, a fim de escolher qual o melhor ou mais adequado para ser utilizado. Pelos critérios de qualidade de software verifica-se que o pacote IntLab satisfaz todas as características de qualidade, já pelos critérios para avaliação de linguagens de programação o ambiente C-XSC se destaca e, segundo critérios de qualidade do intervalo, Python apresentou melhores resultados.**

***Palavras-chave*—Ambientes intervalares, aritmética intervalar, qualidade do software, avaliação de linguagens de programação, qualidade do intervalo.**

I. Introdução

Problemas numéricos na computação científica originam-se primordialmente da impossibilidade de se operar com os números reais diretamente, pois tem-se que representar uma grandeza contínua (a reta real) de forma discreta (palavras de máquina). O sistema de ponto flutuante [1], [2] é uma aproximação prática dos números reais. Infelizmente, como um sistema algébrico, suas características são extremamente pobres quando comparadas com o dos números reais.

Os intervalos foram definidos com o objetivo inicial de automatizar a análise do erro computacional. Através da utilização de intervalos, tem-se um controle automático de erros com limites confiáveis, além de provas de existência e não existência de solução de diversos problemas [3].

M.D.C. Balboni é aluno de graduação em Ciência da Computação na Universidade Federal de Pelotas. E-mail: (baalbis@gmail.com).

L.M. Tortelli é aluno de graduação em Ciência da Computação na Universidade Federal de Pelotas. E-mail: (lmtortelli@inf.ufpel.edu.br).

M. Lorini é aluna de graduação em Ciência da Computação na Universidade Federal de Pelotas. E-mail: (mlorini@inf.ufpel.edu.br).

V.S. Furlan é aluno de graduação em Ciência da Computação na Universidade Federal de Pelotas. E-mail: (vsfurlan@inf.ufpel.edu.br).

A.F. Finger é aluna de pós-graduação em Ciência da Computação na Universidade Federal de Pelotas. E-mail: (affinger@inf.ufpel.edu.br).

A.B. Loreto é professora Doutora do Centro de Desenvolvimento Tecnológico na Universidade Federal de Pelotas. E-mail: (aline.loreto@inf.ufpel.edu.br).

A aritmética intervalar utiliza intervalos reais para representar valores infinitos, valores desconhecidos ou para representar valores contínuos que podem ser conhecidos ou não. Os intervalos servem para representar dados inexatos, aproximações e erros de truncamento de procedimentos.

Na matemática intervalar, o valor real x é aproximado por um intervalo $\mathbf{x}$, que possui como limites inferior e superior números de máquina de forma que o intervalo contenha x [3]. O tamanho deste intervalo pode ser usado como medida para avaliar a qualidade de aproximação [4]. Os cálculos reais são substituídos por cálculos que utilizam a aritmética intervalar [3].

No processo de resolução de problemas podem ser constatadas fontes de erros, tais como: propagação dos erros nos dados iniciais, arredondamento e erros de truncamento, causados ao se truncar sequências infinitas de operações aritméticas após um número finito de etapas. Neste contexto percebe-se a importância de técnicas intervalares. Ressalta-se que uma resposta intervalar carrega com ela a garantia de sua incerteza. Um valor pontual não carrega medidas de sua incerteza. Mesmo quando uma análise de sondagem do erro é executada, o número resultante é somente uma estimativa do erro que pode estar presente [5].

O método para implementação de operações e algoritmos intervalares em máquinas é realizado por meio do critério de semimorfismo proposto por Kulisch [6], [7]. Considerando que o controle do erro numérico é feito através do uso de intervalos ao invés de números reais, Kulisch [7] e Kulisch e Miranker [6] propuseram que a implementação da aritmética intervalar seja realizada através da chamada aritmética de exatidão máxima, o que significa a busca para que resultados numéricos ou sejam um número de ponto flutuante ou estejam entre dois números de ponto flutuantes consecutivos.

Neste contexto, para obtenção de resultados com maior exatidão os cálculos numéricos devem ser suportados pela matemática intervalar e pela aritmética de exatidão máxima, o que implica que em computadores sejam realizados por meio das linguagens ou bibliotecas que tenham definidos o tipo intervalo e as operações sobre o tipo, usualmente denominadas de linguagens XSC (*eXtended Scientific Computation*) [8].

Existem diversos ambientes computacionais para matemática intervalar, dentre as quais o presente trabalho analisa: Maple Intervalar, IntLab, IntPy, C-XSC, Fortan-XSC. Pascal-XSC e Java-XSC. Destes, dois são softwares que fazem o uso da metodologia de aritmética intervalar, é o caso dos pacotes Maple Intervalar e o IntLab, os demais são linguagens de programação que suportam o tipo intervalo.

Diante de várias opções de ambientes intervalares e a dificuldade em escolher qual o melhor ou mais adequado para utilizar em futuros trabalhos, o presente trabalho analisa tais ambientes considerando critérios de qualidade de software, critérios de avaliação de linguagens de programação e critérios de qualidade do intervalo.

O presente trabalho organiza-se da seguinte maneira: na Seção 2 descrevem-se as características de cada ambiente intervalar e o estado da arte; na Seção 3 apresentam-se os critérios de avaliação de software, critérios de linguagens de programação e critérios de qualidade do intervalo; na Seção 4 encontram-se as avaliações dos ambientes e na Seção 5 expõem-se as conclusões. Por fim, as principais referências.

II. Ambientes Intervalares

O uso de ambientes de programação que suportem representação intervalar para as operações de cálculo científico favorece o controle automático de erros através de métodos auto-validáveis (métodos que se encarregam de verificar e garantir a exatidão dos cálculos efetuados) [9].

Descrevem-se brevemente os ambientes intervalares a serem analisados segundo critérios de qualidade de software [10], critérios de linguagens de programação [11] e critérios de qualidade do intervalo [4]. São eles:

- **C-XSC**: C++ *for eXtended Scientific Computing* é uma linguagem de programação de fácil uso, especialmente para aplicações científicas e de engenharia. Trata-se de uma biblioteca numérica para a Computação Científica baseada na linguagem C++. Possui alta geração de resultados com exatidão verificados automaticamente [8]. Fornece um grande número de tipos de dados numéricos e operadores predefinidos. Estes tipos são implementados como classes da linguagem C++. Assim, o C-XSC permite a programação de alto nível de aplicações numéricas em C++. Disponível para muitos ambientes computacionais que possuem um compilador C-XSC. Suas características mais importantes são [12]:
  - Aritmética intervalar para números reais, complexos, intervalares e intervalares complexos com propriedades definidas matematicamente;
  - Tipos de dados com alta exatidão;
  - Operadores aritméticos predefinidos com alta exatidão;
  - Aritmética de múltipla precisão dinâmica e funções padrão;
  - Controle de arredondamento para os dados de entrada e saída;
  - Bibliotecas de rotinas para resolução de problemas numéricos;
  - Resultados numéricos com rigor matemático.

  C-XSC é particularmente adequado para o desenvolvimento de algoritmos numéricos que proporcionam resultados precisos verificados automaticamente. Todas as operações básicas (intervalo) são de exatidão máxima [8].
- **Maple Intervalar**: É um pacote do Maple que possui operações aritméticas intervalares. O software matemático Maple é um ambiente interativo com uma interface amigável que, para muitas finalidades, dispensa a programação. As bibliotecas do Maple, uma vez carregadas, disponibilizam os comandos e operadores necessários para cálculos específicos. Possui uma linguagem de programação fundamentada no conceito de linguagem interpretada e um mecanismo de construção e distribuição de pacotes de programas e funções. É largamente utilizado em computação científica e também na implementação de protótipos de sistemas de grande porte que, após testados no Maple, são posteriormente implementados em outras linguagens que proporcionem um processamento mais rápido do que sistemas de computação algébricos. Software proprietário [13].
- **IntLab**: IntLab (*INTerval LABoratory*) é um pacote desenvolvido para o software Matlab® [14]. Contém tipos de dados básicos e operadores para aritmética intervalar, bem como uma variedade de métodos numéricos usando intervalos. O foco principal é produzir resultados confiáveis. Qualquer resultado prova ser verdadeiro sob quaisquer circunstâncias, nomeadamente em matéria de erros de arredondamento e todos os termos de erro. A filosofia do IntLab é que tudo é escrito em código Matlab para garantir melhor portabilidade, usa extensivamente rotinas BLAS, o que garante tempos de computação rápida, comparáveis à aritmética pura de ponto flutuante. Vetor intervalo e operações de matriz são muito rápidos em IntLab, no entanto, os cálculos não lineares e *loops* podem retardar o sistema de forma significativa, devido à sobrecarga de interpretação e uso extensivo do conceito de operador. Software proprietário [14].
- **IntPy**: É um pacote intervalar desenvolvido na linguagem de programação Python. É composto de 2 subpacotes, *support* e *irreal*, além de um módulo que encapsula as classes de exceção. O subpacote *support* agrupa toda a funcionalidade de suporte ao IntPy e é composto de três módulos: *roundingmodule.c*, *general.py*, *stdfunc.py*. O primeiro módulo é uma extensão C para Python que manipula os modos de arredondamento do processador. Este módulo é dependência para toda funcionalidade do IntPy. O segundo módulo organiza todas as peças de software pontuais. Uma dessas peças é a função *rational2fraction* (racional) responsável por converter uma representação em *string* de números racionais em outra representação mais facilmente manipulável por computador.

  O subpacote *irreal* compreende 2 módulos: *irreal.py* e *irmath.py*. O primeiro implementa a classe IReal que representa o tipo Intervalo Real. Todas as operações aritméticas e de conjuntos, relações de ordem e funções auxiliares são implementadas nessa classe. O segundo módulo implementa extensões intervalares das funções padrão. Todos são implementados na linguagem Python e reconhecidos como software livre GPL [15].
- **Pascal-XSC**: Por ser uma extensão da linguagem de programação Pascal, o Pascal-XSC contém as mesmas características do Pascal padrão. Pelo uso dos módulos matemáticos do Pascal-XSC, algoritmos numéricos são facilmente programados e provêm alta exatidão e verificação automática de resultados [12]. Programas escritos nesta

linguagem são de fácil leitura. Para o Pascal-XSC já foram desenvolvidas bibliotecas que possibilitam a resolução de sistemas de equações lineares utilizando verificação automática do resultado.

- **Fortran-XSC**: A linguagem Fortran, apesar de ser antiga, foi projetada para otimização do uso de memória, pois na época a memória era um recurso escasso e caro tornando as aplicações mais complexas em termos de implementação [11]. O Fortran-XSC surgiu como um módulo do Fortran 77 para cálculos que demandavam de grande precisão aritmética. Quando houve a evolução para o Fortran 90, os engenheiros decidiram que não haveria uma reescrita do compilador, tornando assim o módulo XSC totalmente compatível com o Fortran 90. Fortran é uma linguagem altamente legível, uma vez que este critério foi o pilar de sua criação e seu módulo XSC foi desenvolvido para facilitar o desenvolvimento de algoritmos científicos com verificação automática de resultados. As principais características do Fortran-XSC são:
  - Técnicas de auto-validação numérica para as mais diversas aplicações;
  - Para cálculo de números existem uma série de pacotes com todas as propriedades intervalares e complexas;
  - Alocação dinâmica de memória.

  Fortran foi especialmente projetado para cálculos científicos e ainda é fortemente utilizado em diversos campos de pesquisa científica e engenharia [16].
- **Java-XSC**: O ambiente de programação Java é uma linguagem amplamente utilizada devido ao seu paradigma de programação, portabilidade, simplicidade e robustez [17]. Devido a estes quesitos, o Java-XSC foi implementado como uma biblioteca para cálculos científicos com o objetivo de conter o erro criado pela máquina ao representar os dados numéricos em toda sua extensão. A ferramenta Java é extremamente potente em se tratando de portabilidade, uma vez que seu sistema de implementação é híbrido (compilado e interpretado). O interessante no Java é sua forte tipagem e declarações de operações, uma vez que a precisão é um fator importante da linguagem. Suas características são:
  - Extinta a existência de ponteiros, sua substituição foi definida pela passagem por referência;
  - Alta exatidão através da forte tipagem e pela forma que a linguagem é implementada;
  - Grande número de operadores artiméticos;
  - Altamente difundida entre os programadores;
  - Alocação dinâmica da memória, aumentando assim a otimização e a eficiência dos algoritmos;
  - *Garbage Colletor*, elimina qualquer área não usada da memória;
  - Intervalos representados no tipo *double*.

A. Trabalhos Relacionados

Segundo Hölbig [12], o estudo das técnicas da aritmética intervalar é importante devido a sua aplicabilidade em uma grande variedade de problemas de engenharia e de outras ciências, tais como: a determinação de potenciais em certas redes elétricas, o cálculo do *stresse* numa armação de construção ou de uma estrutura de ponte, o cálculo do padrão de escoamento num sistema hidráulico com ramos interconectados ou o cálculo das estimativas das concentrações de reagentes sujeitos a simultâneas reações químicas, entre outras. No trabalho em questão, foram desenvolvidos estudos sobre a linguagem de programação para computação científica Pascal-XSC, a qual é uma linguagem de programação de propósito geral, que proporciona condições especiais à implementação de algoritmos numéricos sofisticados, que verificam matematicamente os resultados.

Os resultados obtidos pelos métodos intervalares implementados, utilizando a linguagem Pascal-XSC, mostraram-se, em sua grande maioria, com boa qualidade. Hölbig [12] comparou com os resultados pontuais dos sistemas testados e verificou a mesma grandeza de exatidão e, ainda, que a solução pontual estava contida no intervalo solução. Mostrou-se que a ordem do diâmetro dos intervalos soluções foi de 10^{-15}.

O trabalho devido a Mesquita [9] descreve os principais conceitos da teoria intervalar e apresenta uma ferramenta computacional, C-XSC, utilizada na implementação de algoritmos intervalares. Afirma que a ferramenta C-XSC demonstrou ser uma boa opção na computação de cálculos matemáticos, porém não justificou a escolha por este ambiente e também não comparou com demais ambientes intervalares.

Salienta-se que trabalhos relacionados a estudo e análise de ferramentas intervalares ou ambientes intervalares são raros na literatura.

Ambientes intervalares ou ferramentas intervalares, conforme Hölbig [12], suportam o tipo intervalo, possuem operações aritméticas sobre o tipo [3] e contemplam a aritmética de exatidão máxima [6], [7]. Além das implementações de operações e algoritmos intervalares serem realizadas por meio do critério de semimorfismo proposto por Kulisch e Miranker [6].

III. Critérios de avaliação

Como não existem na literatura critérios para avaliar ambientes intervalares e existem ambientes que são softwares e outros linguagens de programação, o presente trabalho adota como critérios: Avaliação para Linguagens de Programação devido a Sebesta [11], Qualidade de Software definido por Rocha [10] e Qualidade do Resultado Intervalar elaborado por Ratschek [4].

A. Métodos de Avaliação para as Linguagens de Programação

Dentre as mais diversas linguagens de programação existentes, é necessário avaliá-las através de critérios que as diferenciam uma das outras. As linguagens de programação são classificadas de acordo como são escritas e como são executadas. Estas peculiaridades de cada aplicação são ditas como critérios para avaliação de linguagens de programação. De acordo com Sebesta [11], os critérios são:

- Legibilidade: consiste em como os programas podem ser lidos e entendidos por qualquer indivíduo, uma vez

que um código melhor entendido torna-se mais eficiente, suscetível a otimizações e a manutenção;

- Manutenção: a manutenabilidade de uma linguagem de programação está diretamente ligada aos ajustes que podem ser realizados durante ou depois do projeto ou ligada a uma nova funcionalidade do código;
- Simplicidade: o programa simples contém comandos bem definidos e com um número de componentes básicos para a criação do código;
- Ortogonalidade: visa em dado um conjunto relativamente pequeno de instruções primitivas da linguagem, estas podem se compor para poder fornecer instruções complexas, sem ter a necessidade de comandos padrões;
- Suporte a abstração: consiste na análise da escrita do código fonte, se ele pode ser entendido e utilizado ignorando alguns detalhes. Porém, um programa que consiste em apresentar esse tipo de usabilidade durante a execução pode tornar-se um algoritmo difícil de identificar os possíveis erros presentes durante a execução. Está diretamente ligado a legibilidade e a forma em que ela está organizada internamente;
- Verificação de tipos: é o processo de testar se existem erros de tipo no programa fonte, ou por meio do compilador ou durante a execução, esse último chamado de interpretador. Ao se determinar os tipos previamente, a transição entre eles se torna mais difícil durante a execução do programa, acarretando assim uma maior confiabilidade aos resultados obtidos. A falta de verificação de tipos em um algoritmo que demande de cálculos precisos, pode dar liberdade de introduzir um erro dentro da operação, acarretando perda de precisão;
- Manipulação de excessões: é a capacidade de um programa interceptar erros em tempo de execução, colocar em prática medidas corretas e, depois, prosseguir.

Os critérios, definidos anteriormente, afetam de forma direta para o fim que a linguagem será projetada.

1) Linguagens Compiladas e Interpretadas: Este tópico visa apresentar o conceito de uma linguagem interpretada ou compilada e com isto determinar qual é mais aceitável em quesito de eficiência em cálculos numéricos. As linguagens de programação podem ser classificadas e avaliadas quanto ao seu critério de implementação. Porém, outro fator importante é a forma como a linguagem de programação atua sobre a CPU (*central processing unit*) do computador, como esta é executada, alocada na memória, que tipos de processamentos ela pode fazer, entre outros aspectos. Uma linguagem compilada tem como principal característica a velocidade de execução e a geração de um arquivo objeto. Este arquivo é criado no processo chamado de compilação, o qual basicamente é a tradução do programa escrito em uma linguagem de programação para uma linguagem de máquina. Este método de compilação atravessa várias etapas conforme a Figura 1.

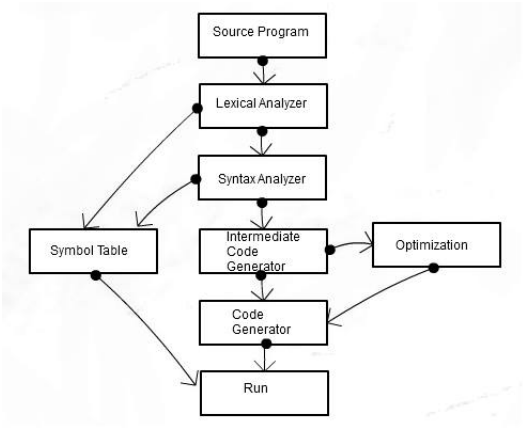

Figura 1: Processo de etapas de um compilador.

O analisador léxico é quem determina quem são os identificadores, as palavras reservadas, os símbolos dos operadores e os símbolos de pontuação. O analisador semântico tem como objetivo arranjar todas as unidades do analisador sintático e criar uma árvore de análise, que é caracterizada pela forma em que o programa será executado. Outra etapa do processo de compilação consiste em, a partir das análises anteriores, criar um código intermediário para serem feitas atribuições de tipos de dados e otimizações antes do código final ser gerado. Uma vantagem da utilização da compilação ao invés da interpretação é que, apesar da demora do processo de compilar um algoritmo, o código final obtido é mais otimizado e será executado mais rapidamente, uma vez que todas as instruções presentes já foram pré-executadas anteriormente. As linguagens compiladas são C, Fortran e Java.

Uma linguagem caracterizada como interpretada tem como principal vantagem a existência de uma camada adicional de software, chamado interpretador, o qual irá "ler" este algoritmo, cada instrução por vez, sem a necessidade de gerar um arquivo a ser executado, conforme apresenta a Figura 2. Este software que fica entre o código e o sistema operacional (SO) fornece uma máquina virtual para a linguagem, tornando assim o processo de manutenção do código facilitado, uma vez que o código dará problema quando a instrução errônea for executada.

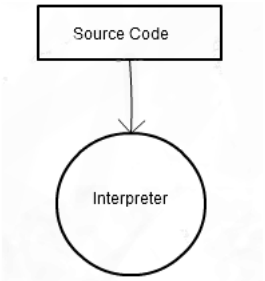

Figura 2: Processo de interpretação.

Diferentemente do compilador, o interpretador não faz qualquer tradução do programa fonte para uma linguagem de máquina, ele simplesmente executa cada instrução do código seguindo a ordem que estas estão aninhadas e estruturadas, tornando esta forma de processamento entre 10 a 100 vezes mais lenta do que sistemas compilados. Uma das principais linguagens de programação que executam seus algoritmos deste modo é Python [11].

Uma exceção a ambas as regras é a linguagem de programação Java. Utiliza um sistema híbrido que unifica os principais pontos de ambas formas de implementação. Este processo consiste em traduzir como compilador o código fonte, criando assim o *bytecode*, e executar este em uma MVJ (Máquina Virtual Java).

Através de todos os conceitos abordados é possível avaliar qualquer linguagem de programação quanto aos seus critérios e suas formas de execução.

B. Fatores De Qualidade Do Software

Segundo Rocha [9], os objetivos de qualidade de um software determinam as propriedades gerais que o produto deve possuir. Os fatores de qualidade do produto determinam a qualidade do ponto de vista dos diferentes usuários do produto (usuário final, alunos e professores).

Os objetivos de qualidade são atingidos através de fatores e subfatores. O objetivo confiabilidade da representação é atingido através de dois fatores: legibilidade e manipulabilidade. O fator legibilidade avalia a possibilidade de diferentes pessoas entenderem o programa com relativa facilidade para que possam utilizá-lo. Relacionados a este fator têm-se os subfatores: clareza e concisão. O fator manipulabilidade avalia a possibilidade de diferentes pessoas manipularem o programa com facilidade. É atingido através de três subfatores: disponibilidade, estrutura e rastreabilidade.

O objetivo confiabilidade conceitual é caracterizado pela implementação satisfatória do que foi especificado e projetado correspondendo, desta forma, às necessidades que geraram seu desenvolvimento. Dois fatores contribuem para o alcance desse objetivo: fidedignidade e integridade. O fator fidedignidade avalia a correspondência do programa às especificações e ao projeto. Este fator é atingido através de três subfatores: precisão, completeza e necessidade. O fator integridade está relacionado à capacidade de o programa resistir a situações hostis (dados errados, agressões, etc). Dois subfatores permitem o alcance desse fator: robustez e segurança.

Utilizabilidade é a característica de qualidade de programas que determina a conveniência e a viabilidade de sua utilização ao longo do tempo. Este objetivo exige a existência dos outros dois: confiabilidade da representação e confiabilidade conceitual.

O objetivo desse trabalho é escolher qual ambiente é o melhor (ou mais adequado) para utilizar em trabalhos futuros (em pesquisas que requerem controle de erros e garantia de exatidão nas soluções de problemas), considerando suas características enquanto softwares de boa qualidade, as quais são: manutenibilidade, operacionalidade, portatilidade, reutilizabilidade, eficiência, rentabilidade, avaliabilidade.

Manutenabilidade: avalia a facilidade com que o programa pode ser adaptado a fim de atender às necessidades de modificação que surgem depois de seu desenvolvimento.

Operacionalidade: avalia a facilidade de comunicação com o usuário. Este fator possui dois subfatores: oportunidade e amenidade ao uso.

Portatilidade: é a característica de um programa poder ser operado de maneira fácil e adequado em diferentes configurações de equipamentos além da original.

Reutilizabilidade: é a característica que avalia a possibilidade do reaproveitamento, total ou parcial, de funções desenvolvidas em um programa em outras aplicações.

Eficiência: é a característica de o programa realizar suas funções sem desperdício de recursos (memória e periféricos entre outros).

Rentabilidade: é a característica de o programa ter uma relação custo-benefício aceitável.

Avaliabilidade: é a característica que avalia a facilidade com que um programa pode ser avaliado. Este fator possui dois subfatores: verificabilidade e validabilidade.

Segundo Gonçalves [18], ao realizar a escolha do software é preciso fazer algumas considerações:

- Quanto ao conteúdo, se o mesmo atende às necessidades de seu objetivo curricular ou pesquisa;
- Se o software permite modificações a fim de atender às necessidades individuais;
- Na operação do programa, como é tratado o erro dos usuários, qual o controle que o usuário tem da operação do programa, se existe um bom manual.

Estas considerações podem ser subdivididas em aspectos pedagógicos e técnicos. Como o objetivo do trabalho é analisar ambientes intervalares para computação científica, acredita-se na relevância dos aspectos técnicos, os quais são:

- Apresentação clara de objetivos e indicação das possibilidades de uso;
- Adequação ao equipamento disponível nos respectivos ambientes de ensino;
- Facilidade de instalação e desinstalação;
- Fornecimento do manual de utilização;
- Compatibilidade e integração com outros softwares e hardwares;
- Layout que facilite a utilização do programa;
- Atualização de conteúdo via Internet.

Destaca-se ainda, como critérios técnicos, aqueles softwares que se enquadram nas categorias de código aberto, gratuitos, proprietário ou livres e tipos de licença. Salienta-se que não é objetivo do trabalho analisar os ambientes intervalares segundo esses critérios técnicos, uma vez que os mesmos são suportados por todos os ambientes objetos da pesquisa.

C. Qualidade do resultado

Quando se trabalha com números de ponto flutuante o resultado obtido é apenas uma aproximação de um valor real e erros são gerados por arredondamentos ou por algoritmos instáveis, levando algumas vezes a resultados incorretos.

A fim de completar o objetivo do presente trabalho, faz-se uma análise da qualidade dos intervalos obtidos na linguagem

C-XSC, no pacote IntPy, na biblioteca IntLab e no software Maple Intervalar, os quais possibilitam a programação utilizando operações definidas na matemática intervalar [3]. A análise dá-se através da aplicação dos indicadores estatísticos descritivos intervalares: Média, Mediana, Variância, Amplitude Total, Desvio Padrão, Coeficiente de Variação, Covariância e Coeficiente de Correlação. Com esta análise verifica-se qual ambiente retorna o intervalo solução com melhor qualidade, ou seja, menor diâmetro (comprimento).

Segundo Ratschek [4], os computadores utilizam aritmética de ponto flutuante. Nesta aritmética, números reais são aproximados por um subconjunto de números reais chamados representação numérica da máquina. Devido a esta representação, são gerados dois tipos de erros: quando uma entrada de valor real é aproximada por um número de máquina e quando o erro é causado por resultados intermediários aproximados pelos números de máquina. A aritmética intervalar fornece uma ferramenta para estimar e controlar esses erros automaticamente.

Quando se realiza a computação com número de máquina $\tilde{x}$ não existe estimativa do erro $|\tilde{x}-x|$. A computação com utilização de intervalos fornece as seguintes estimativas para o erro [4]:

- Erro Absoluto: $|x-m(\mathbf{x})| < w(\mathbf{x})/2$ onde $m(\mathbf{x})$ é o ponto médio do intervalo $\mathbf{x}$ e $w(\mathbf{x}) = |\bar{x}-\underline{x}|$ é o diâmetro do intervalo $\mathbf{x}$;
- Erro Relativo: $\left|\frac{x-m(\mathbf{x})}{x}\right| \leq \frac{w(\mathbf{x})}{2min|\mathbf{x}|}$ se $0 \notin \mathbf{x}$, onde $|\mathbf{x}| = |\mathbf{x} : x \in \mathbf{x}|$.

O tamanho desse intervalo pode ser usado como medida para avaliar a sua qualidade [4].

Aplicam-se estas medidas de erros com o objetivo de verificar a qualidade do intervalo solução em diferentes ambientes intervalares, verificando qual ambiente retorna o intervalo com melhor qualidade, obtido para os indicadores estatísticos intervalares conforme definidos em Loreto [19].

IV. Avaliações

As linguagens definidas para trabalhar com entradas intervalares recebem uma nomenclatura diferente, chamada XSC (*eXtended Scientific Computing*). Dentre as linguagens que se enquadram em tal definição, cita-se: C-XSC, Python (biblioteca IntPy), Fortran-XSC e Java-XSC. Avalia-se a sua forma de execução e processamento de dados, a fim de verificar qual linguagem obtém os melhores resultados.

A. Análise de linguagem de programação para cálculo científico

Apesar das linguagens de programação terem suas principais diferenças quanto aos critérios estabelecidos e quanto sua forma de execução, é possível avaliar todas sobre os critérios definidos e, deste modo, determinar a linguagem mais aplicável para cálculos científicos.

Dentre as linguagens compiladas, C-XSC é a que mais se destaca, uma vez que esta é uma linguagem mais difundida, por ser derivada da linguagem C/C++, e por apresentar bons conceitos de abstração, manipulação de exceções, manutenabilidade e legibilidade. Diferentemente do Java, que por ser uma linguagem com grande portabilidade, seus códigos podem se tornar menos legíveis e, consequentemente, com menor número de abstrações. A linguagem Fortran não se torna uma grande concorrente, uma vez que suas aplicações são limitadas, por não poder ser atribuído conceitos matemáticos recursivos e por esta não apresentar uma separação da memória para este fim.

Em contrapartida, a linguagem Python é altamente abstrata e expressiva, uma vez que suas instruções são simples de serem entendidas e a combinação delas definem instruções mais complexas. Afirmando uma boa caracterização para o conceito de ortogonalidade, porém esta linguagem perde em desempenho, uma vez que o processador precisa de mais tempo para executar o algoritmo com toda sua extensão.

Foi desenvolvido um pacote denominado de PyPy, o qual supre a necessidade de Python ser previamente compilada e otimizada antes de ser executada. Deste modo, em relação ao desempenho que Python perdia por ser interpretada, agora há um ganho quanto aos critérios de vinculação de tipos e tempo de execução.

A linguagem Pascal funciona no sistema de implementação compilado, ou seja, seu processo de criação de código executável passa pelos processos de análise léxica, criação da árvore de análise, entre outras. Em contrapartida a linguagem Pascal foi criada para fins didáticos, não ocorrendo assim uma real exploração dos recursos disponíveis pelo hardware.

B. Características dos Softwares

Neste caso, foram avaliados dois pacotes de softwares, são eles: Maple Intervalar e IntLab (MatLab), ambos proprietários, de boa atualização, fácil instalação e possuem manual de utilização.

Faz-se então a análise das características dos softwares, conforme apresenta a Tabela I.

Tabela I: Características dos softwares

Características	Maple Intervalar	IntLab
Manutenibilidade	Bom	Bom
Operacionalidade	Bom	Bom
Portatilidade	Médio	Médio
Reutilizabilidade	Bom	Bom
Eficiência	Ruim	Bom
Rentabilidade	Ruim	Bom
Avaliabilidade	Bom	Bom

Analisando a Tabela I observa-se que o pacote IntLab, do software MatLab, satisfaz praticamente todas as características enquanto software de boa qualidade. Sobre o critério de Portabilidade ressalta-se que ambos os pacotes (IntLab e Maple Intervalar) não satisfazem totalmente, porque possuem versões apenas para o sistema operacional Windows.

Softwares são desenvolvidos para atender às necessidades de seus usuários e devem ter uma vida útil, produtiva e longa [18]. Neste contexto, o presente trabalho analisou os ambientes intervalares Maple Intervalar e IntLab (MatLab) considerando suas características enquanto softwares de boa

qualidade. Enfatiza-se que a qualidade de software pode ser definida como um conjunto de propriedades a serem satisfeitas em determinado grau, de modo que o software satisfaça as necessidades de seus usuários.

C. Qualidade do resultado

Para a realização das simulações utilizou-se o computador com as seguintes configurações: processador Intel® Core™ i5 CPU 760 @ 2.80GHz Quad-Core, L1 Cache 64Kb, L2 Cache 512Kb, L3 Cache 8Mb, Memória RAM de 16GB DDR3 1333MHz, armazenamento HD Sata 500GB modelo ATA Samsung HD502HJ, sistema operacional Linux Ubuntu 11.10 (IntPy e C-XSC), Windows 7 Ultimate (IntLab e Maple).

Para os cálculos, consideram-se os dados de entrada

$$\{x_1,\ldots,x_n\}, x_1 \in \mathbf{x_1},\ldots,x_n \in \mathbf{x_n},$$

ou seja, domínios intervalares onde $\mathbf{x_1} = [\underline{x}_1, \overline{x}_1], \ldots, \mathbf{x_n} = [\underline{x}_n, \overline{x}_n]$.

Representa-se cada um dos n valores reais $\{x_1,\ldots,x_n\}$ de uma amostra aleatória de uma população por intervalos $\mathbf{x_i} = [x_i - \delta, x_i + \delta]$, onde δ é a margem de precisão escolhida para estes valores.

Na Tabela II apresentam-se os resultados obtidos através de testes utilizando os seguintes valores reais para os indicadores: 14.2687, 16.0697, 16.8786, 18.2961, 19.8179, 20.7834, 21.0517, 21.8146, 22.0711, 22.6562, 24.6094, 25.1095, 26.9127 e 27.0258.

A partir da Tabela II é possível verificar que o valor real de cada indicador está contido no intervalo solução gerado pelos indicadores descritivos com entradas intervalares.

Com a finalidade de verificar a qualidade dos intervalos obtidos e apresentados na Tabela II, as Tabelas III e IV, apresentadas na página 8, trazem as estimativas para o Erro Absoluto (EA) e para o Erro Relativo (ER), bem como o diâmetro (w($\mathbf{x}$)) calculado para cada intervalo.

Observa-se, analisando as Tabelas III e IV, que o ambiente Maple Intervalar, em alguns indicadores estatísticos, apresentou maiores erros Absoluto e Relativo que os demais ambientes. Já o IntPy retornou intervalos solução mais exatos em todos os indicadores calculados.

Até o momento, foram pesquisados e definidos os ambientes intervalares, bem como as medidas de erros para verificação da qualidade dos intervalos solução, obtidos para os indicadores estatísticos intervalares da média, variância, desvio padrão, coeficiente de variação, covariância e coeficiente de correlação.

Com o sistema de ponto flutuante F(10, 14, -10, 10) (ou com quatorze casas decimais) verifica-se, através das medidas de erros (absoluto e relativo), qual ambiente retorna intervalo solução com mais exatidão.

V. Conclusão

Com o objetivo de escolher um ambiente intervalar para utilizar em futuras pesquisas, o presente trabalho analisa, segundo critérios de: qualidade de software, linguagens de programação e qualidade do resultado intervalar os ambientes Maple Intervalar – pacote do software Maple, IntLab – pacote do software Matlab, C-XSC, Fortran-XSC, Pascal-XSC e Java-XSC- linguagens de programação.

Considerando as características de qualidade de software, o pacote IntLab possui qualidade por satisfazer praticamente todas as características de manutenibilidade, operacionalidade, reutilizabilidade, eficiência, rentabilidade e avaliabilidade.

Em relação aos critérios para avaliação de linguagens de programação, considerando linguagens compiladas, o C-XSC se destaca por apresentar bons conceitos de abstração, manipulação de exceções, manutenabilidade e legibilidade. Porém, a linguagem interpretada Python é altamente abstrata e expressiva, uma vez que suas instruções são simples e a combinação destas definem instruções mais complexas. Em se tratando de exatidão, verificada com os critérios de qualidade do intervalo, Python apresentou os melhores resultados em comparação a C-XSC e aos pacotes Maple Intervalar e IntLab.

Sobre os critérios técnicos, verifica-se que os softwares que são proprietários, como o IntLab e o Maple Intervalar, são de boa atualização, fácil instalação e possuem manual de utilização. Já os que são gratuitos, como C-XSC e IntPy, também possuem manual de utilização, mas suas atualizações não são tão boas quanto dos softwares proprietários. Salienta-se que o IntPy é de fácil instalação porém o C-XSC não. Não deixando de citar que no IntLab existe uma grande exatidão nos resultados e que o Maple Intervalar é o de mais fácil utilização.

Assim, pelos resultados, conclui-se que os melhores ambientes nos critérios citados acima são: C-XSC e IntPy. Salienta-se que o ambiente a ser adotado em futuras pesquisas será Intpy, devido a exatidão dos resultados e por possuir um pacote Pypy que supre a necessidade do Python ser previamente compilada e otimizada antes de ser executada. Garantindo a qualidade do intervalo solução e conhecendo o ambiente intervalar que retorna o intervalo com melhor qualidade, tem-se uma ferramenta intervalar confiável para ser utilizada em diversas aplicações que utilizem os indicadores estatísticos descritivos intervalares.

Agradecimentos

Os autores agradecem à UFPel e CAPES pelo suporte financeiro na realização do presente trabalho.

Tabela II: Valores Reais e Intervalares

Indicador	Valores Reais	C-XSC	IntLab	Maple	IntPy
Média	21.2404	[21.235428,21.245429]	[21.2354,21. 2455]	[21.2344,21.2464]	[21.2403757142, 21.2403957142]
Mediana	21.4331	[21.428499, 21.438501]	[21.4284, 21.4386]	[21.4281,21.4381]	[21.4331399999, 21.4331600000]
Ampl. Total	12.7571	[12.746999, 12.767001]	[12.7469, 12.7671]	[12.7471,12.7671]	[12.7570799999, 12.7571200000]
Variância	14.3066	[14.2964, 14.3168]	[14.2442, 14.3677]	[14.2226,14.3906]	[14.3065269110, 14.306772815]
Desvio Padrão	3.78241	[3.774184, 3.790438]	[3.7741, 3.7904]	[3.77361, 3.79121]	[3.78239697956, 3.78242948580]
Coef. Variação	0.178076	[0.177646, 0.178496]	[0.1776, 0.1785]	[0.17764, 0.17850]	[0.17807563618, 0.17807733425]
Covariância	5.151523	[5.105631, 5.197465]	[5.0946, 5.2040]	[5.08052, 5.22252]	[5.15140819363, 5.15163820792]
Coef. Correlação	0.645051	[0.644621, 0.645481]	[0.6345, 0.6556]	[0.63763,0.65247]	[0.64502883893, 0.64507392065]

Tabela III: Comparativo Linguagens

Indicador	C-XSC			IntPy		
	EA	ER	w(x)	EA	ER	w(x)
Média	$0.00004 < 0.005$	$0.000002 \leq 0.0002$	0.010001	$0.0 < 0.00001$	$0.0 \leq 4 \times 10^{7}$	$2.00000000028 \times 10^{-05}$
Mediana	$0.00035 < 0.005$	$0.000016 \leq 0.0002$	0.010002	$0.0 < 0.00001$	$0.0 \leq 4 \times 10^{7}$	$2.0000100001 \times 10^{-05}$
Ampl. Total	$0.0001 < 0.01$	$0.000007 \leq 0.0007$	0.0204	$0.0 < 0.00002$	$0.0 \leq 1 \times 10^{6}$	0.000245904
Variância	$0.00076 < 0.0662$	$0.000049 \leq 0.0043$	0.020002	$4 \times 10^{10} < 0.0001$	$1 \leq 8 \times 10^{6}$	$4.00001000003 \times 10^{-05}$
Desvio Padrão	$0.0001 < 0.00812$	$0.000026 \leq 0.0021$	0.016254	$2 \times 10^{10} < 0.00006$	$2 \times 10^{11} \leq 8 \times 10^{6}$	$3.25062399997 \times 10^{-05}$
Coef. Variação	$0.000005 < 0.0004$	$0.000028 \leq 0.0023$	0.00085	$1 \times 10^{11} < 3 \times 10^{6}$	$3 \times 10^{11} \leq 9 \times 10^{6}$	1.69807×10^{-06}
Covariância	$0.000025 < 0.0459$	$0.000004 \leq 0.0089$	0.091834	$1 \times 10^{10} < 0.0001$	$2 \times 10^{11} \leq 1 \times 10^{5}$	0.00023001429
Coef. Correlação	$0.00017 < 0.01732$	$0.000165 \leq 0.0170$	0.00086	$2 \times 10^{10} < 1 \times 10^{5}$	$8 \times 10^{10} \leq 4 \times 10^{5}$	$4.50817200001 \times 10^{-05}$

Tabela IV: Comparativo Ambientes

Indicador	IntLab			Maple		
	EA	ER	w(x)	EA	ER	w(x)
Média	$0.00005 < 0.0050$	$0.000002 \leq 0.0002$	0.0101	$0.00004 < 0.0050$	$0.000002 \leq 0.0002$	0.012
Mediana	$0.0003 < 0.0102$	$0.000013 \leq 0.0002$	0.0102	$0.7633 < 0.0050$	$0.036250 \leq 0.0002$	0.01
Ampl. Total	$0.00005 < 0101$	$0.000003 \leq 0.0004$	0.1235	$0 < 0.01$	$0. \leq 0.02$	0.168
Variância	$0.00065 < 0.0617$	$0.000045 \leq 0.0043$	0.0202	$0.0007 < 0.0662$	$0.000049 \leq 0.0043$	0.02
Desvio Padrão	$0.0001 < 0.0082$	$0.000026 \leq 0.0021$	0.0163	$0.0001 < 0.0084$	$0.000027 \leq 0.0021$	0.0176
Coef. Variação	$0.00005 < 0.0004$	$0.000280 \leq 0.0025$	0.0009	$0.000005 < 0.0004$	$0.000028 \leq 0.0023$	0.00086
Covariância	$0.0022 < 0.0547$	$0.000427 \leq 0.0107$	0.1094	$0.5039 < 0.0581$	$0.097830 \leq 0.0103$	0.142
Coef. Correlação	$0.00005 < 0.010$	$0.000077 \leq 0.0166$	0.0211	$0.0563 < 0.0104$	$0.097950 \leq 0.0168$	0.01484

REFERÊNCIAS

[1] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, 1991.

[2] ANSI/IEEE STD 754, "Ieee standard for binary floating - point arithmetic," *ACM GIGPLAN*, vol. 22, pp. 5–48, 1987.

[3] R. E. Moore, *Interval Analysis*. Englewood Cliffs: Prentice Hall, 1966.

[4] H. Ratschek and J. Rokne, *New Computer Methods for Global Optimization*. Horwood Publishing Limited, 1988.

[5] A. F. Finger, A. B. Loreto, M. A. Campos, F. Varjão, and M. das Graças dos Santos, "A complexidade computacional dos problemas de computar intervalos encapsuladores para as variáveis aleatórias uniforme, exponencial e pareto," *XXXVIII Conferencia Latinoamericana en Informática*, 2012.

[6] U. Kulisch and L. Miranker, *Computer Arithmetic in Theory and Practice*, 1st ed. Academic Press, 1981.

[7] U. W. Kulisch. (2008, apr) Complete interval arithmetic and its implementation on the computer. Disponível em http://www.math.kit.edu/iwrmm/seite/preprints/media/preprint%20nr.

[8] R. Klatte, U. Kulisch, A. Wiethoff, C. Lawo, and M. Rauch, *C-XSC - A C++ Class Library for Extended Scientic Computing*. Springer-Verlag, 1993.

[9] M. P. de Mesquita, "Matemática intervalar: Princípios e a ferramenta c-xsc," Lavras, 2002.

[10] A. R. Rocha and G. H. B. de Campos, "Avaliação da qualidade de software educacional," *Em Aberto*, no. 57, jan./mar. 1993.

[11] R. W. Sebesta, *Conceps of Programming Languages*, 4th ed. Colorado Spring: Addison Wesley Longman, 1999, vol. 1.

[12] C. A. Hölbig, "Ambiente de alto desempenho com alta exatidão para resolução de problemas," Ph.D. dissertation, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2005.

[13] M. A. Campos, G. P. Dimuro, J. F. F. Araújo, and A. M. Dias, "Probabilidade intervalar e cadeias de markov intervalares no maple," *Tendências em Matemática Aplicada e Computacional*, vol. 3, no. 2, pp. 53–62, 2002.

[14] MATLAB. (2013, apr) Mathworks. Disponível em http://www.mathworks.com.

[15] IntPy. (2013, apr) Interval arithmetic package. Disponível em http://pypi.python.org/pypi/intpy/0.1.3.

[16] W. V. Walter, "Fortran-xsc a portable fortran 90 module library for accurate and reliable scientific computing," *Springer Vienna*, vol. 9, 1993.

[17] R. V. Ferreira, B. J. T. Fernandes, E. S. R. Bezerra, and M. A. Campos, "Interval computation with java-xsc," *XXVIII CNMAC*, 2005.

[18] I. Gonçalves. (2012, mar) Análise dos diferentes tipos de softwares usados na educação. Disponível em http://http://geocities.ws ivanete20032002/aval-softword_iva.html.

[19] A. B. Loreto, "Análise da complexidade computacional de problemas de estatística descritiva com entradas intervalares," Ph.D. dissertation, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2006.