VIII ERMAC-R3

5º Encontro Regional de Matemática Aplicada e Computacional
20 a 22-Novembro-2008
Universidade Federal do Rio Grande do Norte - Natal/RN

IntPy: Um Framework Intervalar em Python

Rafael Menezes Barreto **Marcília Andrade Campos**,
Centro de Informática,
Universidade Federal de Pernambuco, UFPE,
Recife - Pernambuco - Brasil,
E-mail: {rmb3, mac}@cin.ufpe.br

Resumo

A biblioteca IntPy, apresentada neste trabalho, fornece um framework para a matemática intervalar com exatidão máxima desenvolvido como software livre usando a linguagem de programação Python. Esta linguagem foi escolhida por ser especialmente adaptada ao trabalho científico devido a sua sintaxe simples aliada com a grande expressividade possibilitada pela alta abstração e pluralidade de paradigmas que a caracteriza. Além disso, por também ser software livre e gozar de grande popularidade na comunidade de computação científica, a escolha de Python torna a IntPy uma alternativa viável e de baixo custo aos grandes pacotes intervalares baseados em software proprietário (INTLAB, intpakX), permitindo assim maior popularização da técnica intervalar.

Palavras-chave

Ponto flutuante; matemática intervalar; exatidão máxima; Python; software livre.

Introdução

Problemas numéricos em circuitos digitais originam-se, primordialmente, da impossibilidade de se operar com o conjunto dos números reais de forma direta uma vez que é necessário discretizá-lo. O sistema de ponto flutuante [6] é uma aproximação prática dos números reais. Infelizmente, como sistema algébrico, é extremamente pobre quando comparado com os reais.

Atualmente, uma abordagem bastante difundida para a solução de problemas numéricos com foco na exatidão é a matemática intervalar [8, 9]. Além dos erros associados à discretização (arredondamento, truncamento), a matemática intervalar também é adaptada para tratar imprecisões associadas às grandezas de trabalho (altura, peso, tempo, temperatura, etc.), o que a torna especialmente atraente na modelagem computacional de eventos físicos.

A matemática intervalar fornece uma ferramenta para estimar e controlar erros numéricos automaticamente. Ao invés de aproximar um número real x por um número de máquina, x é aproximado por um intervalo $X = [x_1, x_2]$ tal que $x \in X$ e x_1, x_2 são números de máquina. A amplitude de X, $x_2 - x_1$, pode ser utilizada como uma medida de qualidade da aproximação. Dessa forma, X provê informação acerca da exatidão do valor computado possibilitando manter controle sobre o erro.

Este trabalho apresenta o framework IntPy para a matemática intervalar escrito como software livre usando a linguagem de programação Python [16]. Python incorpora os principais atributos de uma linguagem computacional de cunho científico como sintaxe clara, alta abstração de dados e procedimentos, construções apropriadas ao manejo matemático, sobrecarga de operadores, pluralidade de paradigmas e interatividade, o que justifica sua escolha. Além disso, as características de multi-plataforma e livre distribuição implicam em alta acessibilidade, tornando o IntPy uma alternativa viável e de baixo custo aos grandes pacotes intervalares baseados em software proprietário (INTLAB, intpakX), permitindo assim maior popularização da técnica intervalar.

Este texto está organizado em quatro seções sumarizadas como segue. A segunda seção introduz a matemática intervalar definindo o espaço dos intervalos, a aritmética, a exatidão máxima, bem como funções auxiliares como ponto médio, diâmetro, distância de Hausdorff, etc. A terceira seção descreve os detalhes de implementação do framework. A quarta seção mostra exemplos de uso do IntPy e o compara com o pacote intervalar PyInterval [4]. A quinta seção conclui com as considerações finais.

A Matemática Intervalar

No que se segue, todas as estruturas da matemática intervalar implementadas pelo framework IntPy são definidas.

O Espaço dos Intervalos

O conjunto de todos os $x \in \mathbb{R}$ satisfazendo a condição $a_1 \leq x \leq a_2$ é chamado um *intervalo* [8, 9] e é denotado por:

$$[a_1, a_2] = \{x \in \mathbb{R} \mid a_1 \leq x \leq a_2\}$$

O espaço dos intervalos, $\mathbb{IR}$, é, então:

$$\mathbb{IR} = \{[a_1, a_2] \mid a_1 \leq a_2 \wedge a_1, a_2 \in \mathbb{R}\}$$

Sejam $X = [x_1, x_2]$, $Y = [y_1, y_2]$, $Z = [z_1, z_2]$ e $W = [w_1, w_2]$.

Igualdade

Os intervalos X e Y são iguais se e só se $x_1 = y_1$ e $x_2 = y_2$.

Relações de Ordem

Existem várias relações de ordem intervalares descritas na literatura. Entre elas, tem-se a extensão intervalar da relação de ordem real $<$ definida por Moore [8, 9] como segue:

$$X < Y \Leftrightarrow x_2 < y_1$$

Outra relação de ordem é dada por:

$$X \leq Y \Leftrightarrow x_1 \leq y_1 \wedge x_2 \leq y_2$$

Uma vez que intervalos também são conjuntos, as relações usuais se aplicam.

Aritmética

As operações aritméticas no espaço dos intervalos $\mathbb{IR}$ [11, 8, 9] são definidas por:

(i) $X + Y = [x_1 + y_1, x_2 + y_2]$

(ii) $X \times Y = [min(\sigma), max(\sigma)]$

onde $\sigma = \{x_1y_1, x_1y_2, x_2y_1, x_2y_2\}$.

Complementarmente, são definidos os elementos *negativo* e *recíproco*, respectivamente:

(iii) $-X = [-x_2, -x_1] = \{-x \mid x \in X\}$

(iv) $\frac{1}{X} = [\frac{1}{x_2}, \frac{1}{x_1}] = [\frac{1}{x} \mid x \in X \wedge 0 \notin X]$

Assim:

(v) $X - Y = X + (-Y)$

(vi) $X \div Y = X \times \frac{1}{Y}$

Exatidão Máxima

A primeira questão que se impõe a qualquer tentativa de implementação da aritmética intervalar em computadores é o fato de que o espaço de intervalos não é mais definido sobre os reais, mas sim sobre o sistema de ponto flutuante. É necessário, portanto, utilizar-se de método que garanta a inclusão do resultado verdadeiro no resultado intervalar. Esse método é a aritmética de exatidão máxima [5, 6, 7].

Aplicada à matemática intervalar, a aritmética de exatidão máxima redefine a aritmética intervalar de modo que:

(i) $X + Y = [\bigtriangledown(x_1 + y_1), \bigtriangleup(x_2 + y_2)]$

(ii) $X \times Y = [min(\bigtriangledown\sigma), max(\bigtriangleup\sigma)]$

onde $\bigtriangledown$ e $\bigtriangleup$ constituem os arredondamentos direcionados do sistema de ponto flutuante.

Outras Definições

O valor absoluto de um intervalo X é definido por:

$$|X| = max\{|x_1|, |x_2|\}$$

e seu diâmetro por:

$$w(X) = x_2 - x_1$$

Define-se também o ponto médio de X:

$$mid(X) = \frac{1}{2}(x_1 + x_2)$$

Objetivando transformar $\mathbb{IR}$ em *espaço métrico* define-se a distância de Hausdorff entre X e Y:

$$d(X, Y) = max\{|y_1 - x_1|, |y_2 - x_2|\}$$

As operações de união e interseção são definidas da maneira usual. Adicionalmente, define-se a união convexa (*hull*):

$$X \sqcup Y = [min\{x_1, y_1\}, max\{x_2, y_2\}]$$

O Framework IntPy

No estágio atual de desenvolvimento, o framework IntPy suporta todas as definições precedentes inclusive as relações e operações usuais de conjuntos como teste de pertinência, continência, união e interseção. Além disso, os arredondamentos direcionados foram implementados de forma nativa, através do chaveamento dos modos de arredondamento do processador. Algumas implementações como a PyInterval [4] e Java-XSC – tanto a versão da UFRN [2] como a da UFPE [3] – emulam os arredondamentos direcionados utilizando a técnica de inflacionamento em que os extremos do intervalo são somados ou subtraídos ao *epsilon* da máquina. Apesar da vantagem da simplicidade na implementação, esta técnica em geral leva a intervalos excessivamente grandes perdendo em precisão. Outra funcionalidade do IntPy não encontrada nessas implementações é a possibilidade da entrada de dados através da representação em string de números racionais, o que impede o truncamento prematuro da entrada, i.e., `"1/3"` ao invés de `"0.3333"`.

Desde que o IntPy é um pacote primordialmente científico, a confiabilidade deve ser um dos principais requisitos. Por esta razão, adotou-se uma metodologia de desenvolvimento baseada em iterações curtas e dirigidas por testes no que é conhecido em engenharia de software como *Test-Driven Development* [1]. A ferramenta utilizada para os testes foi o módulo *doctest* pertencente a biblioteca padrão de Python.

O IntPy foi desenvolvido sob a licença livre GPL [13] utilizando a versão 2.5 de Python [16] e o pacote GCC [15].

A seguir, uma descrição da arquitetura e dos detalhes de implementação dos arredondamentos direcionados é dada.

Arquitetura

O pacote IntPy é composto de 2 subpacotes, *support* e *ireal*, além de um módulo que encapsula as classes de exceção, *errors.py*.

O subpacote *support* agrupa toda a funcionalidade de suporte ao framework IntPy e é composto de 3 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 a funcionalidade do IntPy. O segundo módulo organiza todas as peças de software pontuais. Uma dessas peças é a função `rational2fraction(rational)` responsável por converter uma representação em string de números racionais em outra representação mais facilmente manipulável por computador. O módulo *stdfunc.py* merece atenção especial. O padrão de representação de sistemas de ponto flutuante IEEE 754 [14] obriga a implementação de todos os métodos de arredondamento especificados por ele somente nas operações de soma, subtração, multiplicação, divisão e na função raiz quadrada. Todas as outras funções padrão (`log`, `exp`, `sin`, `cos`, etc.) têm comportamento inconsistente com respeito aos arredondamentos direcionados. O correto funcionamento das funções padrão sob os arredondamentos direcionados é condição necessária para a realização das extensões intervalares básicas. O módulo *stdfunc.py* implementa esse requisito. Atualmente, entretanto, o estágio de desenvolvimento deste módulo ainda é bastante primitivo.

O subpacote *ireal* compreende 2 módulos: *ireal.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 nesta classe. O segundo módulo implementa extensões intervalares das funções padrão. No momento, dado o estágio primário de desenvolvimento do módulo *stdfunc.py* somente as extensões das funções de potenciação com expoente inteiro e raiz quadrada estão implementadas.

O módulo *errors.py* concentra todas as classes de exceção utilizadas no pacote IntPy.

Uma vez que os papéis dos módulos estão bem definidos e há pouco acoplamento, é fácil estender as funcionalidades do framework ou utilizar seus módulos de forma unitária. Por exemplo, é possível utilizar o módulo *roundingmodule.c* de maneira isolada em alguma aplicação que precise controlar os modos de arredondamento do processador. Por outro lado, é fácil implementar um tipo intervalar complexo simplesmente compondo instâncias de `IReal`.

Arredondamentos Direcionados

Por tratar-se de um interpretador, Python não permite acesso de baixo nível direto como o necessário para alterar os modos de arredondamento do processador. Por outro lado, é possível construir extensões para Python [12] utilizando a linguagem de programação C, que permite esse nível de acesso. Além disso, a padronização C99 de C introduziu na biblioteca padrão a possiblidade de acesso e controle do ambiente de ponto flutuante através do arquivo de cabeçalho `<fenv.h>` [10]. Anteriormente, qualquer manipulação no sistema de ponto flutuante deveria ser feita com o uso de linguagem *Assembly*, manipulando diretamente o processador da máquina, tornando a tarefa complexa e descartando qualquer chance de portabilidade. A maior parte dos compiladores C atuais são compatíveis com esse aspecto do C99, de modo que alterações no sistema de ponto flutuante podem ser feitas de forma simples e portável.

A abordagem escolhida no IntPy, portanto, foi a criação da extensão C para Python *roundingmodule.c*, parte do subpacote *support*, que implementa as funções `get_mode()` e `set_mode(mode)`, interfaces para as respectivas `void fegetround(void)` e `int fesetround(int round)` do cabeçalho `<fenv.h>`.

Um exemplo de utilização dessa extensão é mostrado a seguir:

```
>>> from intpy.support import *
>>> bak = rounding.get_mode()
>>> rounding.set_mode(-1)
0
>>> 1.0/10.0
0.099999999999999992
>>> rounding.set_mode(1)
0
>>> 1.0/10.0
0.10000000000000001
>>> rounding.set_mode(bak)
0
```

A atribuição `bak = rounding.get_mode()` faz um backup do modo de arredondamento atual da máquina. O comando `rounding.set_mode(-1)` seta o arredondamento para baixo[1]. Como reflexo disso, a operação `1.0/10.0` resulta no valor arredondado para baixo. Os dois comandos seguintes ilustram o caso do arredondamento para cima. Por fim, o último comando restaura o modo de arredondamento inicial.

Exemplos

Nesta seção, todas as funcionalidades implementadas no framework IntPy são demonstradas e comparadas com as respectivas na PyInterval [4]. A motivação por trás dos testes comparativos com a PyInterval é que esta é uma biblioteca intervalar alternativa também desenvolvida em Python que utiliza a técnica de inflacionamento para os arredondamentos direcionados. Os testes descritos em [4] serão utilizados sempre que aplicável.

[1] Vale notar o retorno de `set_mode(mode)`. Sempre que o modo de arredondamento é setado corretamente, o valor de retorno é zero. Caso contrário, o retorno é diferente de zero.

As Tabelas 1 e 2 comparam a criação de intervalos no IntPy e na PyInterval e mostram que não há diferença entre as duas bibliotecas nesse ponto.

A comparação das operações aritméticas é mostrada nas Tabelas 3 e 4. Como pode-se observar, o IntPy gerou em todos os casos resultados de menor diâmetro que a PyInterval. Isso justifica-se pelo uso do inflacionamento por parte da PyInterval. Apesar da diferença pequena, após reiteradas operações o acúmulo dessa diferença tende a tornar-se notável.

Tabela 1: Criação de intervalos com IntPy

Operação	Resultado
$X = [1.1, 1.2]$	[1.0999999999999999, 1.2000000000000002]
$Y = [3.2, 3.4]$	[3.1999999999999997, 3.4000000000000004]

Tabela 2: Criação de intervalos com PyInterval

Operação	Resultado
$X = [1.1, 1.2]$	[1.0999999999999999, 1.2000000000000002]
$Y = [3.2, 3.4]$	[3.1999999999999997, 3.4000000000000004]

Tabela 3: Operações aritméticas com IntPy

Operação	Resultado
$X + Y$	[4.2999999999999989, 4.6000000000000005]
$X - Y$	[-2.3000000000000007, -1.9999999999999996]
$X \times Y$	[3.5199999999999991, 4.0800000000000018]
$X \div Y$	[0.32352941176470579, 0.37500000000000011]

Tabela 4: Operações aritméticas com PyInterval

Operação	Resultado
$X + Y$	[4.2999999999999989, 4.6000000000000014]
$X - Y$	[-2.3000000000000012, -1.9999999999999993]
$X \times Y$	[3.5199999999999987, 4.0800000000000018]
$X \div Y$	[0.32352941176470568 0.37500000000000017]

O IntPy implementa funcionalidades não descritas em [4], de tal forma que não foi possível executar mais testes comparativos. Essas funcionalidades são exemplificadas a seguir.

Criação de Intervalos

```
>>> # intervalo vazio
>>> IReal()
empty interval
>>> # intervalo indefinido
>>> IReal("undefined")
undefined interval
>>> # entrada por string
>>> IReal("1/3")
[0.33333333333333331, 0.33333333333333337]
>>> IReal("1/10/3", "0.3e2")
[0.29999999999999999, 30.0]
>>> # entrada comum
>>> IReal(0.1)
[0.10000000000000001, 0.10000000000000001]
```

Operações de Conjuntos

```
>>> # interseção
>>> IReal(2, 3) & IReal(2.5)
[2.5, 2.5]
>>> IReal(-1, 0) & IReal(0.25, 10)
empty interval
>>> # união
>>> IReal(2, 3) | IReal(2.5)
[2.0, 3.0]
>>> IReal(-1, 0) | IReal(0.25, 10)
undefined interval
>>> # união convexa
>>> IReal(-1, 0).hull(IReal(0.25, 10))
[-1.0, 10.0]
```

Relações de Ordem

```
>>> # menor que
>>> IReal(2, 3) < IReal(3.1)
True
>>> # menor que ou igual
>>> IReal(3) <= IReal(3)
True
>>> # pertinência
>>> 0.0 in IReal(-1, -0.1)
False
>>> # continência
>>> IReal() in IReal(-1)
True
```

Funções Auxiliares

```
>>> # valor absoluto
>>> abs(IReal(-1, 1))
1.0
>>> # diâmetro
>>> IReal("0.1").diameter()
1.3877787807814457e-17
>>> # ponto médio
>>> IReal("0.1", "0.4").middle()
0.25000000000000006
>>> # distância de Hausdorff
>>> IReal(-10, 5).distance(IReal(10))
20.0
```

Conclusão

Este trabalho apresentou o framework intervalar IntPy, desenvolvido sob a licença livre GPL na linguagem também livre Python. O IntPy fornece as funcionalidades básicas necessárias para o trabalho

com matemática intervalar e, por ser de livre distribuição, é uma alternativa viável de baixo custo às soluções intervalares proprietárias. Além disso, por basear-se em uma arquitetura pouco acoplada, é fácil expandi-lo com novos recursos.

Foram realizadas comparações com o módulo intervalar PyInterval, também desenvolvido em Python, concluindo-se que o uso de arredondamentos direcionados nativos ao invés de inflacionamento diminui o diâmetro dos resultados, melhorando a precisão.

Vale notar que o IntPy é um projeto em seu estágio inicial e, portanto, com grande potencial de crescimento. O próximo passo é a implementação completa de todas as extensões intervalares de funções básicas.

Referências

[1] K. Beck, "Test-Driven Development: By Example", Addison-Wesley Professional, 2002.

[2] B. R. C. Bedregal, J. E. M. Dutra, JAVA-XSC: Estado da arte, *XXXII Conferencia Latinoamericana de Informática (CLEI 2006)*, v.1, 2006, 1-12.

[3] R. V. Ferreira, B. J. T. Fernandes, M. A. Campos, Interval Computations with JAVA-XSC, *XXVIII Congresso Nacional de Matemática Aplicada e Computacional*, 2005, São Paulo.

[4] P. S. Grigoletti, G. P. Dimuro, Módulo Python para Matemática Intervalar, em "TEMA Tend. Mat. Apl. Comput.", 8, No. 1 (2007), 73-82.

[5] U. W. Kulisch, W. L. Miranker, "Computer Arithmetic in Theory and Practice", Technical Report 33658, IBM Thomas L. Watson Research Center, 1979.

[6] U. W. Kulisch, W. L. Miranker, "Computer Arithmetic in Theory and Practice", New York, Academic Press, 1981.

[7] U. W. Kulisch, W. L. Miranker, "The Arithmetic of the Digital Computer: A New Approach", SIAM Review, 1986.

[8] R. E. Moore, "Interval Analysis", N. J., Prentice Hall Inc., Englewood Cliffs, 1966.

[9] R. E. Moore, "Methods and Applications of Interval Analysis", SIAM, Philadelphia, Pennsylnia, 1976.

[10] P. Stephen, "C Primer Plus", Sams Publishing, 2001.

[11] T. Sunaga, "Theory of An Interval Algebra and its Application to Numerical Analysis", 2:29-46, RAAG Memoirs, 1958.

[12] "Extending and Embedding the Python Interpreter", disponível em http://docs.python.org/extending/, acessado em 25 de setembro de 2008.

[13] "GNU General Public License", disponível em http://www.gnu.org/copyleft/gpl.html, acessado em 07 de setembro de 2008.

[14] "IEEE 754: Standard for Binary Floating-Point Arithmetic", disponível em http://grouper.ieee.org/groups/754/, acesssado em 02 de outubro de 2008.

[15] "GNU Compiler Collection", disponível em http://gcc.gnu.org/, acessado em 05 de setembro de 2008.

[16] "The Python Tutorial", disponível em http://docs.python.org/tutorial/, acessado em 25 de setembro de 2008.