Intervall Arithmetik in C++

Josef Höll
Julian Schulz
Thomas M. Pfister

Inhalt

Geschichte

- Intervallarithmetik bereits 300 v. Chr. bekannt durch Archimedes
- Anfang ihres Durchbruches durch Ramon Edgar Moore
  Buch “Interval Analysis”
- Wissenschaftler Gruppen an den Universitäten Karlsruhe und Wuppertal
  Spracherweiterung XSC - “Extensions for Scientific Computation”
- Erste Klassenbibliothek für C++ 1992 mit C-XSC
  - Erweiterung anfang 2000er mit C-XSC 2.0
  - Profil/BIAS (“Programmers Runtime Optimized Fast Interval Library, Basic Interval Arithmetic”)
- Genormt durch IEEE erst im Jahr 2015 (IEEE-1788-2015)

Anwendungsbereiche

- Rundungsfehleranalyse
- Toleranzanalyse
- Computergestütztes Beweisen
- “Fuzzy” Größendarstellung in der “Fuzzy logic”
- Branch and Bound Methoden - Globale Optimierung

Theorie
zur Intervall Arithmetik

Josef Höll

Definition Intervall

Als Menge:

$$[\underline{x}, \overline{x}] = \{x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}\}$$

oder Tupel von Zahlen:

$$\text{A pair of natural numbers } [a, b] \text{ satisfying } a \leq b$$

Intervall Operationen

$$x + y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$$

$$x - y = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]$$

$$x \times y = [\underline{x}\underline{y}, \overline{xy}]$$

Intervall Operationen

$\mathbb{R}$

$$x + y = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$$

$$x - y = [\underline{x} - \overline{y}, \overline{x} - \underline{y}]$$

$$x \times y = [min\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}, max\{\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\}]$$

$$\frac{1}{x} = [\frac{1}{\overline{x}}, \frac{1}{\underline{x}}] \text{ iff } \underline{x} > 0 \text{ or } \overline{x} < 0$$

$$x \div y = [x \times \frac{1}{y}]$$

Div-0

let $\underline{x}, \overline{x} \neq 0$

$$\frac{1}{[\underline{x},0]} = [-\infty, \frac{1}{\underline{x}}]$$

$$\frac{1}{[0,\overline{x}]} = [\frac{1}{\overline{x}}, \infty]$$

$$\frac{1}{[\underline{x},\overline{x}]} = [-\infty, \infty] \text{ if } 0 \in [\underline{x}, \overline{x}]$$

Beispiel

akzeptable Vortragsdauer d = [18, 25]
Schätzung der Sprechdauer
- Kollege 1 T = [5,9]
- Kollege 2 J = [6,8]
- Intro/Outro = 1

Sichere Sprechdauer
- x = ?

$$d = 1 + T + J + x$$
$$[18, 25] = 1 + [5, 9] + [6, 8] + x$$
$$[18, 25] = [12, 18] + x$$
$$[18, 25] - [12, 18] = x$$
$$[0, 13] = x$$

$$[18, 25] = [12, 18] + [6, 7]$$

Algebraische Eigenschaften

- Kommutativ
- Assoziativ
- Nicht multiplikativ oder additiv invers:

$$[1,2] - [-1,2] = [-1,1]$$

$$[1,2]/[1,2] = [\tfrac{1}{2},2]$$

- Teil-distributiv:

$$a(b+c) \subseteq ab + ac$$

Abhängigkeitsproblem

$$\text{let } f(x) = x^2 + x \text{ and } x = [-1, 1]$$

gibt idealerweise

$$[-\tfrac{1}{4}, 2]$$

resultiert aber in

$$[-1, 1]^2 + [-1, 1] = [0, 1] + [-1, 1] = [-1, 2]$$

Wirklich ausgerechnet: Maximum, Infimum von

$$h(x, y) = x^2 + y, \text{ and } x, y \in [-1, 1]$$

Theorem -
Single Use Expression

In an algebraic expression evaluated in exact interval arithmetic, the result is the exact range if each variable occurs only once in the expression.

(moderne) Alternative Systeme

- Midpoint-radius Arithmetic
- Circular Arithmetic
- Rectangular Arithmetic
- Kaucher Arithmetic
- Modal Interval Arithmetic

Anwendung in C++

Julian Schulz

C++ Klassenbibliotheken

C++ Klassenbibliotheken

- C-XSC
- Profil/BIAS

Abb.1

- Boost

Abb.2

Bibliothek-Inhalt

- **Arithmetische Operatoren**
- **Algebraische Funktionen**
- **Trigonometrische Funktionen**
- **Template class** *interval*
- u.v.m

Beispiel aus der Boost-Library

Beispiel aus der Boost-Library

Funktion

```
int sign_polynomial(double x, double P[], int sz)
```

Beispiel aus der Boost-Library

Funktion

```
int sign_polynomial(double x, double P[], int sz)
```

+0.0000000000000000...

oder

-0.0000000000000000...

Beispiel aus der Boost-Library

Funktion

```
#include <boost/numeric/interval.hpp>

int sign_polynomial(double x, double P[], int sz)
```

Beispiel aus der Boost-Library

Funktion

```
#include <boost/numeric/interval.hpp>

int sign_polynomial(double x, double P[], int sz) {

    using namespace boost::numeric;
    using namespace interval_lib;
```

Beispiel aus der Boost-Library

Funktion

```
#include <boost/numeric/interval.hpp>

int sign_polynomial(double x, double P[], int sz)

    using namespace boost::numeric;
    using namespace interval_lib;

    typedef interval<double> I;
```

Beispiel aus der Boost-Library

Funktion

```
#include <boost/numeric/interval.hpp>

int sign_polynomial(double x, double P[], int sz)

    using namespace boost::numeric;
    using namespace interval_lib;

    typedef interval<double> I;

    I y = P[sz - 1];

    for(int i = sz - 2; i >= 0; i--)
        y = y * x + P[i];
```

Beispiel aus der Boost-Library

Funktion

```
#include <boost/numeric/interval.hpp>

int sign_polynomial(double x, double P[], int sz)

    using namespace boost::numeric;
    using namespace interval_lib;

    typedef interval<double> I;

    I y = P[sz - 1];

    for(int i = sz - 2; i >= 0; i--)
        y = y * x + P[i];
```

Beispiel aus der Boost-Library

Funktion

```
#include <boost/numeric/interval.hpp>

int sign_polynomial(double x, double P[], int sz) {

    using namespace boost::numeric;
    using namespace interval_lib;

    typedef interval<double> I;

    I y = P[sz - 1];

    for(int i = sz - 2; i >= 0; i--)
        y = y * x + P[i];

    using namespace compare::certain;

    if (y > 0.) return 1;
    if (y < 0.) return -1;
    return 0;

  }
```

Beispiel aus der Boost-Library

Funktion

```
#include <boost/numeric/interval.hpp>

int sign_polynomial(double x, double P[], int sz) {

    using namespace boost::numeric;
    using namespace interval_lib;

    typedef interval<double> I;

    I y = P[sz - 1];

    for(int i = sz - 2; i >= 0; i--)
        y = y * x + P[i];

    using namespace compare::certain;

    if (y > 0.) return 1;
    if (y < 0.) return -1;
    return 0;

}
```

0.000…
-0.000…
? + ? - 0.000…

Schlusswort

Schlusswort

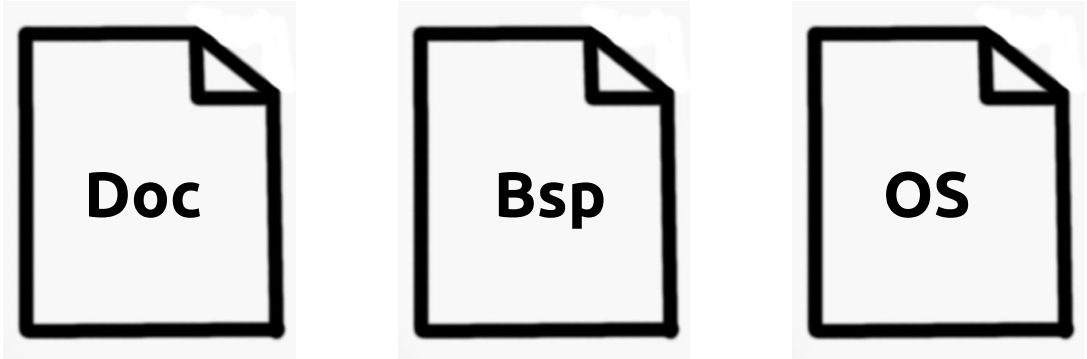

Abb.3

Danke für die Aufmerksamkeit!

Offen für Fragen.

Josef Höll
Julian Schulz
Thomas Pfister

Quellen & Links

- https://docplayer.org/35259758-Effiziente-intervallarithmetik-mit-c.html
- https://hal.inria.fr/hal-01559955/document
- https://interval.louisiana.edu/preprints/survey.pdf
- https://en.wikipedia.org/wiki/Interval_arithmetic
- https://www.boost.org/doc/libs/1_66_0/libs/numeric/interval/doc/examples.htm (C++ Bsp.)
- https://www.boost.org/doc/libs/1_66_0/libs/numeric/interval/doc/interval.htm (Boost Website+Abb.2: Logo)
- http://www.xsc.de/ (C-XSC Website)
- http://www.ti3.tuhh.de/keil/profil/index_e.html (Profil/BIAS Website + Abb.1: Logo)

Abb.3 Eigene Darstellung