Fast infimum-supremum interval operations for double-double arithmetic in rounding-to-nearest

Naoya Yamanaka
(Waseda University)

Shin'ichi Oishi
(Waseda University)

SCAN'2012

Sep. 24th, 2012@Novosibirsk

Introduction

In numerical calculations, the rounding errors occur.

Interval Arithmetic

treating the rounding errors easily ⇒ High reliability

Particularly, in high accuracy computation (e.g. long-arithmetic),

we'd like to get to know how large the rounding errors are.

e.g. MPFR, exflib, ARPREC, and so on...

Introduction

Trade-off between accuracy and executing time:

Accuracy $\Longleftrightarrow$ Executing Time

- need higher accuracy than double arithmetic partially or
- don't need over *doubled* double arithmetic

$\Rightarrow$ Need high speeding

- Accuracy : *doubled* double arithmetic
- Speeding : (hope to be) *fast*

Introduction

How to speed up

We'd like to use

double floating point arithmetic (it's fast)

- Accuracy : doubled double arithmetic
- Method : calculated by double arithmetic

⇒ Bailey's DD Algorithm
(which is based on Error Free Transformations)

Purpose

We propose an interval arithmetic based on Bailey's DD Algorithm

Additionally···

- Bailey's DD Algorithm ：Work on rounding-to-nearst mode
- In some numerical environment, we can not change the rounding mode.

We'd like to propose following interval arithmetic :

(High Accuracy) doubled double arithmetic.
(High Portability) don't need changing rounding mode.
(High Speed) use double floating point arithmetic

Detail

What is Bailey's DD

(High Accuracy) Software of doubled double arithmetic.
(High Portability) It is based on double floating point arithmetic.
(High Speed) Computational time is fast.

Let $x_1, x_2 \in \mathbb{F}$. A doubled-double number x is written by

$$x \simeq x_1 + x_2. \tag{1}$$

Then the following hold:

$$x_1 = \mathrm{fl}\,(x_1 + x_2)\,, \tag{2}$$

$$|x_2| \leqq 2^{-53}\ |x_1|\,. \tag{3}$$

DD is based on Error Free Transformation

- Knuth (1969)
  $[x,\ y] = \textbf{TwoSum}\,(a,\ b)$
- Dekker (1971)
  $[x,\ y] = \textbf{TwoProduct}\,(a,\ b)$

Theorem

$\mathbb{F}$: set of floating point numbers, $\circ \in +, -, \times$,
$\forall a,\ \forall b \in \mathbb{F} \Rightarrow \exists x,\ \exists y \in \mathbb{F}$

$$a \circ b = x + y, \qquad x = fl\,(a + b)\,, \qquad |y| \leqq 2^{-53}|x|$$

Policy of proposed algorithms

- Calculate the upper / lower bounds for each operations.
- Calculate them by floating point operations.

$a_h, a_l, b_h, b_l, c_h, c_l \in \mathbb{F}, \quad \circ \in +, -, \times, \div$

$$c_h + c_l \leftarrow (a_h + a_l) \circ (b_h + b_l)$$

We'd like to get the constant C satisfying:

$$c_h + \mathit{fl}(c_l - \mathit{fl}(C \cdot A)) \leqq (a_h + a_l) \circ (b_h + b_l) \leqq c_h + \mathit{fl}(c_l + \mathit{fl}(C \cdot A))$$

$$A = \mathit{fl}(|a_h| \circ |b_h|)$$

Summary

1. Error bounds

	algoirhtm	error bounds
Addition	Bailey *et. al.*	Yamanaka *et. al.*
Multiplication	Bailey *et. al.*	Yamanaka *et. al.*
Division	Dekker	Yamanaka *et. al.*

2. Calculation Form

$$c_h + c_l \leftarrow (a_h + a_l) \circ (b_h + b_l)$$

We'd like to get the constant C satisfying:

$$c_h + fl(c_l - fl(C \cdot A)) \leqq (a_h + a_l) \circ (b_h + b_l) \leqq c_h + fl(c_l + fl(C \cdot A))$$

$$A = fl(|a_h| \circ |b_h|)$$

Numerical result

Result 1

- Time comparison of calculation inf / sup of DD.
- Show by ratio.
- 100 million calls and time average.

	approx. by Hida & Bailey	error part
addition	(1)	0.83
multiplication	(1)	0.41
division	(1)	0.26

Result 2

- Comparison with MPFR 106 bits.
- Show by ratio.
- 100 million calls and time average.

	Time Ratio		Error Bounds	
	Proposed	MPFR (106bit)	Proposed	MPFR (106bit)
additon	(1)	44.4	9.8e-32	1.2e-32
multiplication	(1)	25.8	3.8e-31	1.2e-32
division	(1)	20.0	7.8e-31	1.2e-32

Application

exponential function of DD

Doubled double format can be

$$x = (-1)^s \sum_{i=0}^{105} m_i \cdot 2^{e-i}. \tag{4}$$

s : sign bit, e : exponent bit, m_i : $m_0 = 1$, $m_i = 0$ or 1.

$$\exp(x) = \exp\left((-1)^s \sum_{i=0}^{105} m_i \cdot 2^{e-i} \right) = \left(\prod_{i=0}^{105} \left(\exp(2^{e-i}) \right)^{m_i} \right)^{(-1)^s}. \tag{5}$$

If we have $\exp(2^i)$ as following form:

$$\exp(2^i) \simeq T_i^{(1)} + T_i^{(2)} \tag{6}$$

Then,

$$\exp(x) \simeq \prod_{i=0}^{105} \left(T_{e-i}^{(1)} + T_{e-i}^{(2)} \right)^{m_i}. \tag{7}$$

Summary

- We proposed fast double-double interval arithmetic algorithm based on a floating point arithmetic.
- Proposed algorithm are based on Bailey's DD.
- From numerical result, proposed algorithm is faster than MPFR (106 bit).
- Proposed algorithm are working on rounding to nearest mode so they work on any computers.