Announcement

RNC7 conference

July 10-12, Nancy (France)

R. Brent, S. Oberman, V. Shapiro

Friendly Competition

Submission deadline: **February 15**

`http://rnc7.loria.fr`

Positions available in 2006 in Nancy

University Nancy 1:

- 5 assistant professors, 3 professors

University Nancy 2:

- 3 assistant professors

INPL:

- 1 assistant professor

Positions available in 2006 in Nancy

University Nancy 1:

- 5 assistant professors, 3 professors

University Nancy 2:

- 3 assistant professors

INPL:

- 1 assistant professor

INRIA (`www.inria.fr`):

- 9 experienced research positions (CR1) [January 19]
- 3 junior research positions in Lorraine [March]
- senior research positions [March]

Fast evaluation of sine and cosine for real argument

Paul Zimmermann

Dagstuhl Seminar 06021, 8-13 January, 2006

Motivation

The MPFR library

Motivation

The MPFR library

- binary **arbitrary precision** floating-point

Motivation

The MPFR library

- binary **arbitrary precision** floating-point
- written in C, LGPL, `www.mpfr.org`

Motivation

The MPFR library

- binary **arbitrary precision** floating-point
- written in C, LGPL, `www.mpfr.org`
- guarantees correct rounding (no disclaimer...)

What is correct rounding?

What is correct rounding?

Given a function f, an input x

What is correct rounding?

Given a function f, an input x

a rounding mode

What is correct rounding?

Given a function f, an input x

a rounding mode

a target precision p

What is correct rounding?

Given a function f, an input x

a rounding mode

a target precision p

compute the p-bit floating-point number closest to $f(x)$

What is correct rounding?

Given a function f, an input x

a rounding mode

a target precision p

compute the p-bit floating-point number closest to $f(x)$

to nearest: error at most $\frac{1}{2}$ ulp

What is correct rounding?

Given a function f, an input x

a rounding mode

a target precision p

compute the p-bit floating-point number closest to $f(x)$

to nearest: error at most $\frac{1}{2}$ ulp

directed rounding: error less than 1 ulp

What is correct rounding?

Given a function f, an input x

a rounding mode

a target precision p

compute the p-bit floating-point number closest to $f(x)$

to nearest: error at most $\frac{1}{2}$ ulp

directed rounding: error less than 1 ulp

for **all operations** (basic, elementary/special, i/o)

The problem

Input: n-bit floating-point number x

The problem

Input: n-bit floating-point number x

Output: n-bit floating-point approximation of $\sin x$ and/or $\cos x$ with rigorous error bound

The problem

Input: n-bit floating-point number x

Output: n-bit floating-point approximation of $\sin x$ and/or $\cos x$ with rigorous error bound

$$x = 0.11000111011101111111111110100001100101110010110010110 \text{ (53 bits)}$$

The problem

Input: n-bit floating-point number x

Output: n-bit floating-point approximation of $\sin x$ and/or $\cos x$ with rigorous error bound

$$x = 0.11000111011101111111111110100001100101110010110010110 \text{ (53 bits)}$$

$$\sin x \approx 0.10110011111000111010100000110111101011001000001000000$$

The problem

Input: n-bit floating-point number x

Output: n-bit floating-point approximation of $\sin x$ and/or $\cos x$ with rigorous error bound

$$x = 0.11000111011101111111111110100001100101110010110010110 \text{ (53 bits)}$$

$$\sin x \approx 0.10110011111000111010100000110111101011001000001000000$$

$$\cos x \approx 0.10110110001001000111001010111000001101011010010000110$$

Plan

- the $\exp$ case: 3 algorithms
- extension to $\sin$ and $\cos$
- implementation within MPFR

Computing $\exp x$: the naive approach

Reference: Brent, rpb032, Theorem 6.1, 1976.

Computing $\exp x$: the naive approach

Reference: Brent, rpb032, Theorem 6.1, 1976.

1. Argument reduction: $x' = x/2^k$

Computing $\exp x$: the naive approach

Reference: Brent, rpb032, Theorem 6.1, 1976.

1. Argument reduction: $x' = x/2^k$
2. Compute $y' = 1 + x' + \frac{x'^2}{2!} + \cdots + \frac{x'^l}{l!}$

Computing $\exp x$: the naive approach

Reference: Brent, rpb032, Theorem 6.1, 1976.

1. Argument reduction: $x' = x/2^k$
2. Compute $y' = 1 + x' + \frac{x'^2}{2!} + \cdots + \frac{x'^l}{l!}$
3. Reconstruction: $y \leftarrow y'^{2^k}$

Computing $\exp x$: the naive approach

Reference: Brent, rpb032, Theorem 6.1, 1976.

1. Argument reduction: $x' = x/2^k$
2. Compute $y' = 1 + x' + \frac{x'^2}{2!} + \cdots + \frac{x'^l}{l!}$
3. Reconstruction: $y \leftarrow y'^{2^k}$

Total cost: $O((l+k)M(n))$

n-bit target precision: $x' = O(2^{-k})$ thus $l = O(n/k)$

Optimal: $l, k \approx n^{1/2}$, cost $O(n^{1/2}M(n))$

The rectangular method (1/2)

Improve the Taylor evaluation. Brent/Kung, Smith (concurrent series).

$$1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \frac{1}{720}x^6 + \frac{1}{5040}x^7$$
$$+\frac{1}{40320}x^8 + \frac{1}{362880}x^9 + \frac{1}{3628800}x^{10} + \frac{1}{39916800}x^{11} + \frac{1}{479001600}x^{12}$$
$$+\frac{1}{6227020800}x^{13} + \frac{1}{87178291200}x^{14} + \frac{1}{1307674368000}x^{15}$$

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6}$$
$$+\frac{1}{24}x^4((1 + \frac{x}{5} + \frac{x^2}{30} + \frac{x^3}{210})$$
$$+\frac{1}{1680}x^4((1 + \frac{x}{9} + \frac{x^2}{90} + \frac{x^3}{990})$$
$$+\frac{1}{11880}x^4(1 + \frac{x}{13} + \frac{x^2}{182} + \frac{x^3}{2730})))$$

14 multiplications by x vs 6 multiplications (x^2, x^3, x^4, 3 by x^4)

The rectangular method (2/2)

Let $X = x^m$, then:

1. $p(x) = P(X)$ with coefficients in $1, x, \ldots, x^{m-1}$

The rectangular method (2/2)

Let $X = x^m$, then:

1. $p(x) = P(X)$ with coefficients in $1, x, \ldots, x^{m-1}$

2a. Compute $1, x, \ldots, x^m = X$: cost $O(mM(n))$

The rectangular method (2/2)

Let $X = x^m$, then:

1. $p(x) = P(X)$ with coefficients in $1, x, \ldots, x^{m-1}$

2a. Compute $1, x, \ldots, x^m = X$: cost $O(mM(n))$

2b. Evaluate coefficients of $P(X)$: cost $O(ln)$

The rectangular method (2/2)

Let $X = x^m$, then:

1. $p(x) = P(X)$ with coefficients in $1, x, \ldots, x^{m-1}$

2a. Compute $1, x, \ldots, x^m = X$: cost $O(mM(n))$

2b. Evaluate coefficients of $P(X)$: cost $O(ln)$

2c. Evaluate $P(X)$: cost $O(l/mM(n))$

The rectangular method (2/2)

Let $X = x^m$, then:

1. $p(x) = P(X)$ with coefficients in $1, x, \ldots, x^{m-1}$

2a. Compute $1, x, \ldots, x^m = X$: cost $O(mM(n))$

2b. Evaluate coefficients of $P(X)$: cost $O(ln)$

2c. Evaluate $P(X)$: cost $O(l/mM(n))$

Optimal: $m \approx \sqrt{l}$, cost $O(l^{1/2}M(n) + ln)$

With reconstruction: $O((l^{1/2} + n/l)M(n) + ln)$

Case $M(n) \gg n^{4/3}$: optimal $l \approx n^{2/3}$, cost $O(n^{1/3}M(n))$ (up to Toom-Cook 6-way)

Case $M(n) \ll n^{4/3}$: optimal $l \approx \sqrt{M(n)}$, cost $O(n\sqrt{M(n)})$

Computing $\exp x$: Brent's algorithm

Brent rpb032, Theorem 6.2, 1976.

Complexity $O(M(n)\log^2 n)$

Write $x = x_1 + x_2 + \cdots + x_k$

Computing $\exp x$: Brent's algorithm

Brent rpb032, Theorem 6.2, 1976.

Complexity $O(M(n)\log^2 n)$

Write $x = x_1 + x_2 + \cdots + x_k$

Fundamental Lemma: $e^{a+b} = e^a \cdot e^b$

Then $e^x = e^{x_1} e^{x_2} \cdots e^{x_k}$

Computing $\exp x$: Brent's algorithm

Brent rpb032, Theorem 6.2, 1976.

Complexity $O(M(n)\log^2 n)$

Write $x = x_1 + x_2 + \cdots + x_k$

Fundamental Lemma: $e^{a+b} = e^a \cdot e^b$

Then $e^x = e^{x_1} e^{x_2} \cdots e^{x_k}$

$x_i = \frac{r_i}{2^{2^i}}$ with $0 \leqslant r_i < 2^{2^{i-1}}$

Example:

$$x = 0.0\underbrace{1}_{r_1}\overbrace{10}^{r_2}\underbrace{0011}_{r_3}\overbrace{10111011}^{r_4}\underbrace{1111111111010000}_{r_5}\ldots$$

$\exp 3$ by binary splitting

Boldo: $3 = 4 \log 2 + 0.22741$

Here: $3 = 2^4 \frac{3}{16}$ (keep sparsity!)

$\exp \frac{3}{16} \approx S := \sum_{n=0}^{N-1} \frac{1}{n!} \frac{3^n}{16^n}$

Basic idea: S is a (huge) rational, compute it exactly!

Write $S(a,b) = \sum_{n=a}^{b-1} \frac{a!}{n!} \frac{3^{n-a}}{16^{n-a}}$ with $S(a,b) = \frac{T(a,b)}{Q(a,b)}$

$Q(a,b) = (a+1) \cdots (b-1) b \cdot 16^{b-a}$

$S(a,b) = S(a,m) + \frac{a!}{m!} \frac{3^{m-a}}{16^{m-a}} S(m,b)$

For $a < m < b$:

$$\begin{aligned} T(a,b) &= Q(m,b)S(a,m) + 3^{m-a}T(m,b) \\ Q(a,b) &= Q(a,m)Q(m,b) \end{aligned}$$

Computation of $\exp 3$ (2/2)

$$3/16 \quad 3/32 \quad 3/48 \quad 3/64 \quad 3/80 \quad 3/96 \quad 3/112 \quad 3/128$$

$$608/512 \qquad 3264/3072 \qquad 7968/7680 \qquad 14720/14336$$

$$1897152/1572864 \qquad\qquad 114361728/110100480$$

$$208886609132928/173173081374720$$

$$\frac{208886609132928}{173173081374720} \approx 1.20623024938$$

$$\exp \frac{3}{16} \approx 1.20623024942$$

Extension to sin/cos

- "naive" method: trivial (rpb032, $\sin(2x) = \pm 2 \sin x \sqrt{1 - \sin^2 x}$)
- rectangular method: ditto
- Brent's method: unfortunately $\sin(a + b) \neq \sin a \sin b$!

Extension to sin/cos (2)

... but $\sin(a+b) = \sin a \cos b + \cos a \sin b$...

Idea: write $X_i = x_i + x_{i+1} + x_{i+2} + \cdots$, thus $X_i = x_i + X_{i+1}$.

$$\begin{aligned} \sin X_i &= \sin x_i \cos X_{i+1} + \cos x_i \sin X_{i+1} \\ \cos X_i &= \cos x_i \cos X_{i+1} - \sin x_i \sin X_{i+1} \end{aligned}$$

1. Compute $\sin x_i$ (binary splitting)
2. Deduce $\cos x_i$ from $\sin x_i$
3. Reconstruct $\sin X_i$ and $\cos X_i$

Complexity

1. Compute $\sin x_i$ (binary splitting)

$O(M(n) \log n)$ for each x_i. Total $O(M(n) \log^2 n)$.

Complexity

1. Compute $\sin x_i$ (binary splitting)

$O(M(n)\log n)$ for each x_i. Total $O(M(n)\log^2 n)$.

2. Deduce $\cos x_i$ from $\sin x_i$

$O(M(n))$ for each x_i. Total $O(M(n)\log n)$.

Complexity

1. Compute $\sin x_i$ (binary splitting)

$O(M(n)\log n)$ for each x_i. Total $O(M(n)\log^2 n)$.

2. Deduce $\cos x_i$ from $\sin x_i$

$O(M(n))$ for each x_i. Total $O(M(n)\log n)$.

3. Reconstruct $\sin X_i$ and $\cos X_i$

$O(M(n))$ for each x_i. Total $O(M(n)\log n)$.

Complexity

1. Compute $\sin x_i$ (binary splitting)

$O(M(n)\log n)$ for each x_i. Total $O(M(n)\log^2 n)$.

2. Deduce $\cos x_i$ from $\sin x_i$

$O(M(n))$ for each x_i. Total $O(M(n)\log n)$.

3. Reconstruct $\sin X_i$ and $\cos X_i$

$O(M(n))$ for each x_i. Total $O(M(n)\log n)$.

Grand Total $O(M(n)\log^2 n)$.

Timings

Athlon XP 1700+, Linux, mpfr-dev with gmp-4.1.4.

bits	`mpfr_sin_cos`	new (sin+cos+rec)	old/new
10000	9.7ms	12.1ms (2.8+2.7+6.0)	0.8
20000	38.6ms	34.0ms (8.6+9.3+14.5)	1.1
50000	221ms	136ms (37+36+57)	1.6
100000	878ms	494ms (135+139+204)	1.8
200000	3.5s	1.4s (0.4+0.4+0.6)	2.5
500000	19.0s	5.5s (1.6+1.7+2.0)	3.5
1000000	63.7	15.8s (5.2+4.5+5.6)	4.0
2000000	218s	41.6s (14.8+12.6+13.0)	5.2

Thank you

RNC7 conference

July 10-12, Nancy (France)

Deadline: **15 February**

This is the “take home” slide