Computing transcendental functions with error bounds (a progress report)

Fredrik Johansson

LFANT seminar, 2015-10-13

Overview

Recent work (last 12 months) on Arb – a C library for arbitrary-precision interval arithmetic

- Much faster elementary functions (published in ARITH22)
- Hypergeometric functions (in brief)
- Elliptic/modular functions (including an addendum to the joint work with Andreas Enge and William Hart, described in a previous talk)

Advertisement: http://nemocas.org/

Computer algebra package for the Julia programming language

Uses the C libraries FLINT, Antic, Pari, GMP/MPIR, MPFR, Arb.
Plus algorithms for generic rings, implemented in Julia.

William Hart, Tommy Hofmann, Claus Fieker, Oleksandr Motsak (Kaiserslautern) and FJ

Elementary functions

Functions: exp, log, sin, cos, atan

Elementary functions

Functions: exp, log, sin, cos, atan

Input: floating-point number $x = a \cdot 2^b$, precision $p \geq 2$

Output: m, r with $f(x) \in [m - r, m + r]$ and $r \approx 2^{-p}|f(x)|$

Precision ranges

Hardware precision ($n \approx 53$ bits)

- Extensively studied - elsewhere!

Precision ranges

Hardware precision ($n \approx 53$ bits)

- Extensively studied - elsewhere!

Medium precision ($n \approx 100$-$10\,000$ bits)

- Multiplication costs $M(n) = O(n^2)$ or $O(n^{1.6})$
- Argument reduction + rectangular splitting: $O(n^{1/3} M(n))$
- In the lower range, software overhead is significant

Precision ranges

Hardware precision ($n \approx 53$ bits)

- Extensively studied - elsewhere!

Medium precision ($n \approx$ 100-10 000 bits)

- Multiplication costs $M(n) = O(n^2)$ or $O(n^{1.6})$
- Argument reduction + rectangular splitting: $O(n^{1/3} M(n))$
- In the lower range, software overhead is significant

Very high precision ($n \gg 10\,000$ bits)

- Multiplication costs $M(n) = O(n \log n \log\log n)$
- Asymptotically fast algorithms: binary splitting, arithmetic-geometric mean (AGM) iteration: $O(M(n) \log(n))$

Recipe for elementary functions

$$\exp(x) \qquad \sin(x), \cos(x) \qquad \log(1+x) \qquad \operatorname{atan}(x)$$

$$\downarrow$$

Domain reduction using π and $\log(2)$

$$\downarrow$$

$$x \in [0, \log(2)) \qquad x \in [0, \pi/4) \qquad x \in [0, 1) \qquad x \in [0, 1)$$

Recipe for elementary functions

$\exp(x)$ $\quad$ $\sin(x), \cos(x)$ $\quad$ $\log(1+x)$ $\quad$ $\operatorname{atan}(x)$

$\downarrow$

Domain reduction using π and $\log(2)$

$\downarrow$

$x \in [0, \log(2))$ $\quad$ $x \in [0, \pi/4)$ $\quad$ $x \in [0, 1)$ $\quad$ $x \in [0, 1)$

$\downarrow$

Argument-halving $r \approx 8$ times

$\exp(x) = [\exp(x/2)]^2$

$\log(1+x) = 2\log(\sqrt{1+x})$

$\downarrow$

$x \in [0, 2^{-r})$

$\downarrow$

Taylor series

Better recipe at medium precision

$\exp(x)$ $\quad$ $\sin(x), \cos(x)$ $\quad$ $\log(1+x)$ $\quad$ $\operatorname{atan}(x)$

$\downarrow$

Domain reduction using π and $\log(2)$

$\downarrow$

$x \in [0, \log(2))$ $\quad$ $x \in [0, \pi/4)$ $\quad$ $x \in [0, 1)$ $\quad$ $x \in [0, 1)$

$\downarrow$

Lookup table with $2^r \approx 2^8$ entries

$$\exp(t+x) = \exp(t)\exp(x)$$

$$\log(1+t+x) = \log(1+t) + \log(1+x/(1+t))$$

$\downarrow$

$x \in [0, 2^{-r})$

$\downarrow$

Taylor series

Argument reduction formulas

What we want to compute: $f(x), \quad x \in [0, 1)$
Table size: $q = 2^r$
Precomputed value: $f(t), \quad t = i/q,\ i = \lfloor 2^r x \rfloor$
Remaining value to compute: $f(y), \quad y \in [0, 2^{-r})$

Argument reduction formulas

What we want to compute: $f(x), \quad x \in [0, 1)$
Table size: $q = 2^r$
Precomputed value: $f(t), \quad t = i/q,\ i = \lfloor 2^r x \rfloor$
Remaining value to compute: $f(y), \quad y \in [0, 2^{-r})$

$$\exp(x) = \exp(t)\exp(y), \quad y = x - i/q$$

$$\sin(x) = \sin(t)\cos(y) + \cos(t)\sin(y), \quad y = x - i/q$$

$$\cos(x) = \cos(t)\cos(y) - \sin(t)\sin(y), \quad y = x - i/q$$

Argument reduction formulas

What we want to compute: $f(x), \quad x \in [0, 1)$
Table size: $q = 2^r$
Precomputed value: $f(t), \quad t = i/q,\ i = \lfloor 2^r x \rfloor$
Remaining value to compute: $f(y), \quad y \in [0, 2^{-r})$

$$\exp(x) = \exp(t)\exp(y), \quad y = x - i/q$$

$$\sin(x) = \sin(t)\cos(y) + \cos(t)\sin(y), \quad y = x - i/q$$

$$\cos(x) = \cos(t)\cos(y) - \sin(t)\sin(y), \quad y = x - i/q$$

$$\log(1 + x) = \log(1 + t) + \log(1 + y), \quad y = (qx - i)/(i + q)$$

$$\operatorname{atan}(x) = \operatorname{atan}(t) + \operatorname{atan}(y), \quad y = (qx - i)/(ix + q)$$

Optimizing lookup tables

$m = 2$ tables with $2^5 + 2^5$ entries gives same reduction as $m = 1$ table with 2^{10} entries

Function	Precision	m	r	Entries	Size (KiB)
exp	≤ 512	1	8	178	11.125
exp	≤ 4608	2	5	23+32	30.9375
sin	≤ 512	1	8	203	12.6875
sin	≤ 4608	2	5	26+32	32.625
cos	≤ 512	1	8	203	12.6875
cos	≤ 4608	2	5	26+32	32.625
log	≤ 512	2	7	128+128	16
log	≤ 4608	2	5	32+32	36
atan	≤ 512	1	8	256	16
atan	≤ 4608	2	5	32+32	36
Total					236.6875

Taylor series

Logarithmic series:

$\operatorname{atan}(x) = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \ldots$

$\log(1+x) = 2\operatorname{atanh}(x/(x+2))$

With $x < 2^{-10}$, need 230 terms for 4600-bit precision

Taylor series

Logarithmic series:

$\operatorname{atan}(x) = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \ldots$

$\log(1+x) = 2\operatorname{atanh}(x/(x+2))$

With $x < 2^{-10}$, need 230 terms for 4600-bit precision

Exponential series:

$\exp(x) = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \ldots$

$\sin(x) = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \ldots, \quad \cos(x) = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \ldots$

With $x < 2^{-10}$, need 280 terms for 4600-bit precision

Taylor series

Logarithmic series:

$\operatorname{atan}(x) = x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \ldots$

$\log(1+x) = 2\operatorname{atanh}(x/(x+2))$

With $x < 2^{-10}$, need 230 terms for 4600-bit precision

Exponential series:

$\exp(x) = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \ldots$

$\sin(x) = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \ldots, \quad \cos(x) = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \ldots$

With $x < 2^{-10}$, need 280 terms for 4600-bit precision

Above 300 bits: $\cos(x) = \sqrt{1 - \sin^2(x)}$

Above 800 bits: $\exp(x) = \sinh(x) + \sqrt{1 + \sinh^2(x)}$

Evaluating Taylor series using rectangular splitting

Paterson and Stockmeyer, 1973:

$\sum_{i=0}^{n} \square x^i$ in $O(n)$ cheap steps $+$ $O(n^{1/2})$ expensive steps

Evaluating Taylor series using rectangular splitting

Paterson and Stockmeyer, 1973:

$\sum_{i=0}^{n} \square x^i$ in $O(n)$ cheap steps + $O(n^{1/2})$ expensive steps

$$
\begin{array}{lllllllllll}
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & & + \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^4 & + \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^8 & + \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^{12} &
\end{array}
$$

Evaluating Taylor series using rectangular splitting

Paterson and Stockmeyer, 1973:

$\sum_{i=0}^{n} \square x^i$ in $O(n)$ cheap steps + $O(n^{1/2})$ expensive steps

$$
\begin{array}{lllllllll}
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^4 \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^8 \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^{12}
\end{array}
$$

- Smith, 1989: elementary and hypergeometric functions

Evaluating Taylor series using rectangular splitting

Paterson and Stockmeyer, 1973:

$\sum_{i=0}^{n} \square x^i$ in $O(n)$ cheap steps + $O(n^{1/2})$ expensive steps

$$
\begin{array}{lllllllll}
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^4 \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^8 \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^{12}
\end{array}
$$

- Smith, 1989: elementary and hypergeometric functions
- Brent & Zimmermann, 2010: improvements to Smith

Evaluating Taylor series using rectangular splitting

Paterson and Stockmeyer, 1973:

$\sum_{i=0}^{n} \square x^i$ in $O(n)$ cheap steps + $O(n^{1/2})$ expensive steps

$$
\begin{array}{lllllllll}
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^4 \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^8 \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^{12}
\end{array}
$$

- Smith, 1989: elementary and hypergeometric functions
- Brent & Zimmermann, 2010: improvements to Smith
- FJ, 2014: generalization to D-finite functions

Evaluating Taylor series using rectangular splitting

Paterson and Stockmeyer, 1973:

$\sum_{i=0}^{n} \square x^i$ in $O(n)$ cheap steps + $O(n^{1/2})$ expensive steps

$$
\begin{array}{lllllllll}
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^4 \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^8 \\
(& \square & + & \square x & + & \square x^2 & + & \square x^3 &) & x^{12}
\end{array}
$$

- Smith, 1989: elementary and hypergeometric functions
- Brent & Zimmermann, 2010: improvements to Smith
- FJ, 2014: generalization to D-finite functions
- New: optimized algorithm for elementary functions

Logarithmic series

Rectangular splitting:

$$x + \frac{1}{2}x^2 + x^3\left\{\frac{1}{3} + \frac{1}{4}x + \frac{1}{5}x^2 + x^3\left\{\frac{1}{6} + \frac{1}{7}x + \frac{1}{8}x^2\right\}\right\}$$

Logarithmic series

Rectangular splitting:

$$x + \frac{1}{2}x^2 + x^3\left\{\frac{1}{3} + \frac{1}{4}x + \frac{1}{5}x^2 + x^3\left\{\frac{1}{6} + \frac{1}{7}x + \frac{1}{8}x^2\right\}\right\}$$

Improved algorithm with fewer divisions:

$$x+\frac{1}{60}\left[30x^2+x^3\left\{20+15x+12x^2+x^3\left\{10+\frac{1}{56}\left[60\left[8x+7x^2\right]\right]\right\}\right\}\right]$$

Exponential series

Rectangular splitting:

$$1+x+\frac{1}{2}\left[x^2+\frac{1}{3}x^3\left\{1+\frac{1}{4}\left[x+\frac{1}{5}\left[x^2+\frac{1}{6}x^3\left\{1+\frac{1}{7}\left[x+\frac{1}{8}x^2\right]\right\}\right]\right]\right\}\right]$$

Exponential series

Rectangular splitting:

$$1+x+\frac{1}{2}\left[x^2+\frac{1}{3}x^3\left\{1+\frac{1}{4}\left[x+\frac{1}{5}\left[x^2+\frac{1}{6}x^3\left\{1+\frac{1}{7}\left[x+\frac{1}{8}x^2\right]\right\}\right]\right]\right\}\right]$$

Improved algorithm with fewer divisions:

$$1+x+\frac{1}{24}\left[12x^2+x^3\left\{4+1\left[x+\frac{1}{30}\left[6x^2+x^3\left\{1+\frac{1}{56}\left[8x+x^2\right]\right\}\right]\right]\right\}\right]$$

Taylor series evaluation using `mpn` arithmetic

We use n-word fixed-point numbers (ulp $= 2^{-64n}$)
Negative numbers implicitly or using two's complement!

Taylor series evaluation using `mpn` arithmetic

We use n-word fixed-point numbers (ulp $= 2^{-64n}$)
Negative numbers implicitly or using two's complement!

Example:

```
// sum = sum + term * coeff
sum[n] += mpn_addmul_1(sum, term, n, coeff)
```

- `term` is n words: real number in $[0, 1)$
- `sum` is $n + 1$ words: real number in $[0, 2^{64})$
- `coeff` is 1 word: integer in $[0, 2^{64})$

Taylor series summation

$$c_0 + c_1 x + c_2 x^2 + c_3 x^3 + x^4 \left[c_4 + c_5 x + c_6 x^2 + c_7 x^3 \right]$$

Taylor series summation

$$c_0 + c_1 x + c_2 x^2 + c_3 x^3 + x^4 \left[c_4 + c_5 x + c_6 x^2 + c_7 x^3 \right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[3], n, c[7])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[6])
sum[n] += mpn_addmul_1(sum, xpowers[1], n, c[5])
sum[n] += c[4]
```

Taylor series summation

$$c_0 + c_1 x + c_2 x^2 + c_3 x^3 + x^4 \left[c_4 + c_5 x + c_6 x^2 + c_7 x^3 \right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[3], n, c[7])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[6])
sum[n] += mpn_addmul_1(sum, xpowers[1], n, c[5])
sum[n] += c[4]
```

```
mpn_mul(tmp, sum, n+1, xpowers[4], n)
mpn_copyi(sum, tmp+n, n+1)
```

Taylor series summation

$$c_0 + c_1 x + c_2 x^2 + c_3 x^3 + x^4 \left[c_4 + c_5 x + c_6 x^2 + c_7 x^3 \right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[3], n, c[7])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[6])
sum[n] += mpn_addmul_1(sum, xpowers[1], n, c[5])
sum[n] += c[4]
```

```
mpn_mul(tmp, sum, n+1, xpowers[4], n)
mpn_copyi(sum, tmp+n, n+1)
```

```
sum[n] += mpn_addmul_1(sum, xpowers[3], n, c[3])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[2])
sum[n] += mpn_addmul_1(sum, xpowers[1], n, c[1])
sum[n] += c[0]
```

Alternating signs

$$c_0 - c_1 x + c_2 x^2 - c_3 x^3 + x^4 \left[c_4 - c_5 x + c_6 x^2 - c_7 x^3 \right]$$

Alternating signs

$$c_0 - c_1 x + c_2 x^2 - c_3 x^3 + x^4 \left[c_4 - c_5 x + c_6 x^2 - c_7 x^3 \right]$$

```
sum[n] -= mpn_submul_1(sum, xpowers[3], n, c[7])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[6])
sum[n] -= mpn_submul_1(sum, xpowers[1], n, c[5])
sum[n] += c[4]
```

Alternating signs

$$c_0 - c_1 x + c_2 x^2 - c_3 x^3 + x^4 \left[c_4 - c_5 x + c_6 x^2 - c_7 x^3 \right]$$

```
sum[n] -= mpn_submul_1(sum, xpowers[3], n, c[7])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[6])
sum[n] -= mpn_submul_1(sum, xpowers[1], n, c[5])
sum[n] += c[4]
```

```
mpn_mul(tmp, sum, n+1, xpowers[4], n)
mpn_copyi(sum, tmp+n, n+1)
```

Alternating signs

$$c_0 - c_1 x + c_2 x^2 - c_3 x^3 + x^4 \left[c_4 - c_5 x + c_6 x^2 - c_7 x^3 \right]$$

```
sum[n] -= mpn_submul_1(sum, xpowers[3], n, c[7])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[6])
sum[n] -= mpn_submul_1(sum, xpowers[1], n, c[5])
sum[n] += c[4]
```

```
mpn_mul(tmp, sum, n+1, xpowers[4], n)
mpn_copyi(sum, tmp+n, n+1)
```

```
sum[n] -= mpn_submul_1(sum, xpowers[3], n, c[3])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[2])
sum[n] -= mpn_submul_1(sum, xpowers[1], n, c[1])
sum[n] += c[0]
```

Including divisions (exponential series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{1}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

Including divisions (exponential series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{1}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[5], n, c[5])
sum[n] += mpn_addmul_1(sum, xpowers[4], n, c[4])
```

Including divisions (exponential series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{1}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[5], n, c[5])
sum[n] += mpn_addmul_1(sum, xpowers[4], n, c[4])
```

```
mpn_divrem_1(sum, 0, sum, n+1, q[4])
```

Including divisions (exponential series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{1}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[5], n, c[5])
sum[n] += mpn_addmul_1(sum, xpowers[4], n, c[4])

mpn_divrem_1(sum, 0, sum, n+1, q[4])

sum[n] += mpn_addmul_1(sum, xpowers[3], n, c[3])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[2])
sum[n] += mpn_addmul_1(sum, xpowers[1], n, c[1])
sum[n] += c[0]
```

Including divisions (exponential series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{1}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[5], n, c[5])
sum[n] += mpn_addmul_1(sum, xpowers[4], n, c[4])

mpn_divrem_1(sum, 0, sum, n+1, q[4])

sum[n] += mpn_addmul_1(sum, xpowers[3], n, c[3])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[2])
sum[n] += mpn_addmul_1(sum, xpowers[1], n, c[1])
sum[n] += c[0]

mpn_divrem_1(sum, 0, sum, n+1, q[0])
```

Including divisions (logarithmic series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3\right] + \frac{1}{q_4}\left[c_4 x^4 + c_5 x^5\right]$$

Including divisions (logarithmic series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{q_0}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

Including divisions (logarithmic series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{q_0}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[5], n, c[5])
sum[n] += mpn_addmul_1(sum, xpowers[4], n, c[4])
```

Including divisions (logarithmic series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{q_0}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[5], n, c[5])
sum[n] += mpn_addmul_1(sum, xpowers[4], n, c[4])

sum[n+1] = mpn_mul_1(sum, sum, n+1, q[0])
mpn_divrem_1(sum, 0, sum, n+2, q[4])
```

Including divisions (logarithmic series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{q_0}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[5], n, c[5])
sum[n] += mpn_addmul_1(sum, xpowers[4], n, c[4])

sum[n+1] = mpn_mul_1(sum, sum, n+1, q[0])
mpn_divrem_1(sum, 0, sum, n+2, q[4])

sum[n] += mpn_addmul_1(sum, xpowers[3], n, c[3])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[2])
sum[n] += mpn_addmul_1(sum, xpowers[1], n, c[1])
sum[n] += c[0]
```

Including divisions (logarithmic series)

$$\frac{1}{q_0}\left[c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \frac{q_0}{q_4}\left[c_4 x^4 + c_5 x^5\right]\right]$$

```
sum[n] += mpn_addmul_1(sum, xpowers[5], n, c[5])
sum[n] += mpn_addmul_1(sum, xpowers[4], n, c[4])

sum[n+1] = mpn_mul_1(sum, sum, n+1, q[0])
mpn_divrem_1(sum, 0, sum, n+2, q[4])

sum[n] += mpn_addmul_1(sum, xpowers[3], n, c[3])
sum[n] += mpn_addmul_1(sum, xpowers[2], n, c[2])
sum[n] += mpn_addmul_1(sum, xpowers[1], n, c[1])
sum[n] += c[0]

mpn_divrem_1(sum, 0, sum, n+1, q[0])
```

Timings (microseconds / function evaluation)

Bits	exp	sin	cos	log	atan
32	0.26	0.35	0.35	0.21	0.20
53	0.27	0.39	0.38	0.26	0.30
64	0.33	0.47	0.47	0.30	0.34
128	0.48	0.59	0.59	0.42	0.47
256	0.83	1.05	1.08	0.66	0.73
512	2.06	2.88	2.76	1.69	2.20
1024	6.79	7.92	7.84	5.84	6.97
2048	22.70	25.50	25.60	22.80	25.90
4096	82.90	97.00	98.00	99.00	104.00

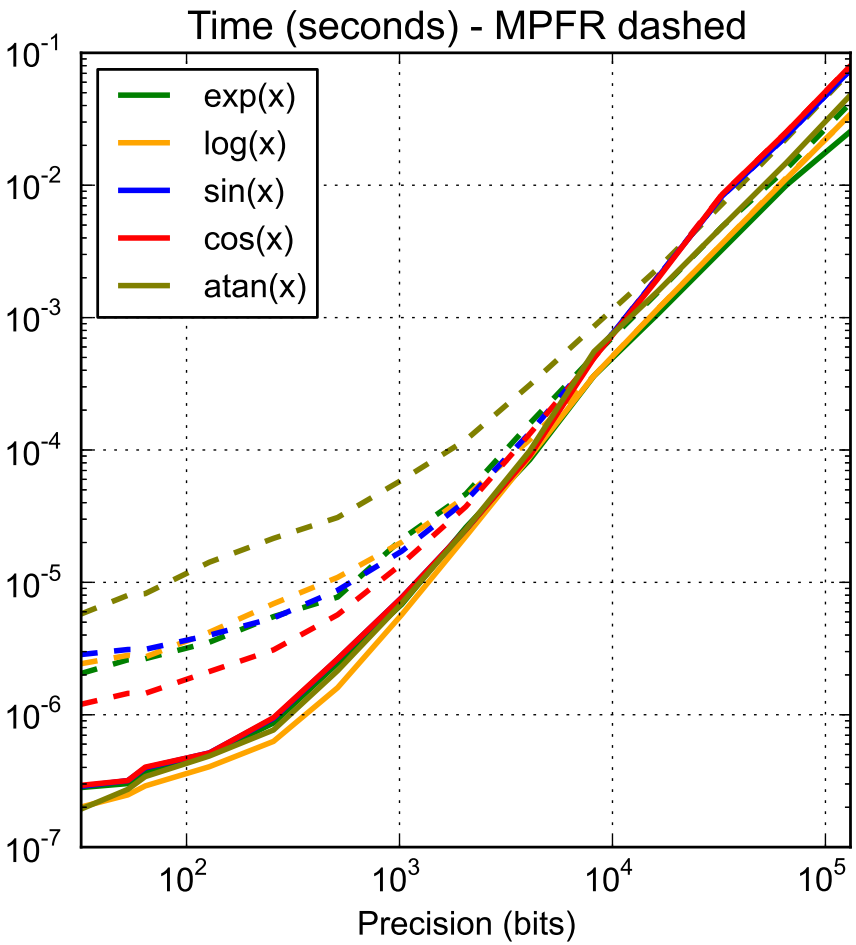
Measurements done on an Intel i7-2600S CPU.

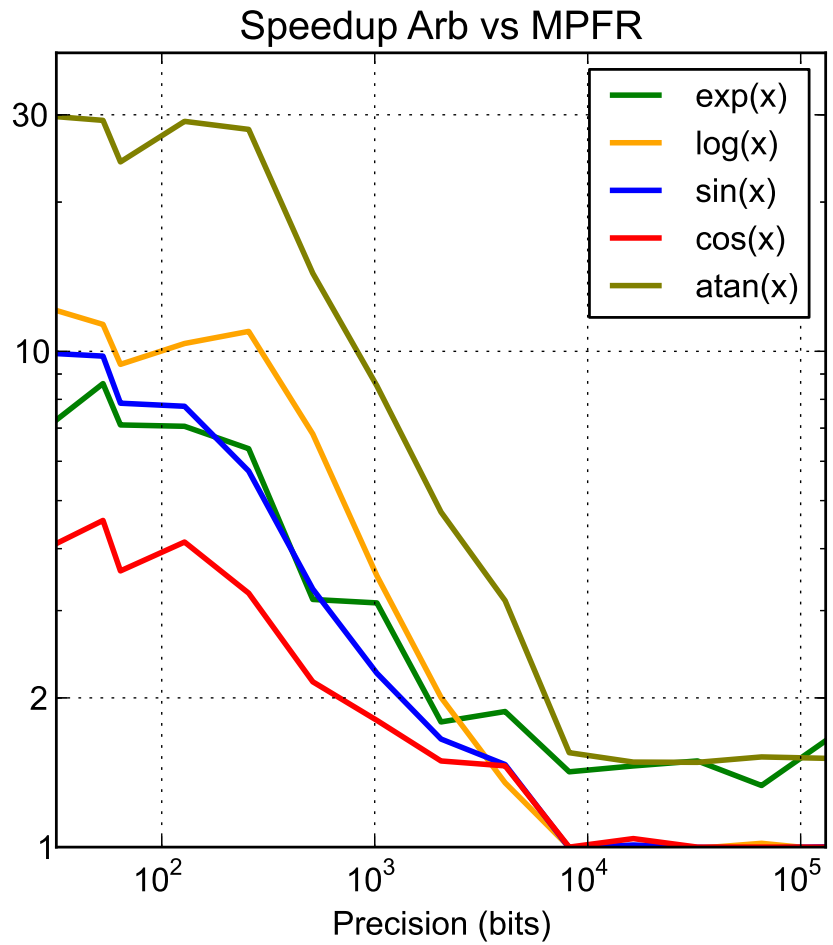
Speedup vs MPFR

Bits	exp	sin	cos	log	atan
32	7.9	8.2	3.6	11.8	29.7
53	9.1	8.2	3.9	10.9	25.9
64	7.6	6.9	3.2	9.3	23.7
128	6.9	6.9	3.6	10.4	30.6
256	5.6	5.4	2.9	10.7	31.3
512	3.7	3.2	2.1	6.9	14.5
1024	2.7	2.2	1.8	3.6	8.8
2048	1.9	1.6	1.4	2.0	4.9
4096	1.7	1.5	1.3	1.3	3.1

Measurements done on an Intel i7-2600S CPU.

Comparison to MPFR





Measurements done on an Intel i7-2600S CPU.

Summary, elementary functions

- Elementary functions with error bounds
- Variable precision up to 4600 bits

Summary, elementary functions

- Elementary functions with error bounds
- Variable precision up to 4600 bits
- `mpn` arithmetic + 256 KB of lookup tables + efficient algorithm to evaluate Taylor series (rectangular splitting, optimized denominator sequence)
- Similar algorithm for all functions (no Newton iteration, etc.)

Summary, elementary functions

- Elementary functions with error bounds
- Variable precision up to 4600 bits
- `mpn` arithmetic + 256 KB of lookup tables + efficient algorithm to evaluate Taylor series (rectangular splitting, optimized denominator sequence)
- Similar algorithm for all functions (no Newton iteration, etc.)
- Improvement over MPFR: up to 3-4x for cos, 8-10x for sin/exp/log, 30x for atan
- Gap to double precision LIBM (EGLIBC): 4-7x

Coverage of special functions in Arb

NIST Digital Library of Mathematical Functions

Foreword
Preface
Mathematical Introduction
1 Algebraic and Analytic Methods
2 Asymptotic Approximations
3 Numerical Methods
4 Elementary Functions
5 Gamma Function
6 Exponential, Logarithmic, Sine, and Cosine Integrals
7 Error Functions, Dawson's and Fresnel Integrals
8 Incomplete Gamma and Related Functions
9 Airy and Related Functions
10 Bessel Functions
11 Struve and Related Functions
12 Parabolic Cylinder Functions
13 Confluent Hypergeometric Functions
14 Legendre and Related Functions
15 Hypergeometric Function
16 Generalized Hypergeometric Functions and Meijer G-Function
17 q-Hypergeometric and Related Functions
18 Orthogonal Polynomials
19 Elliptic Integrals
20 Theta Functions
21 Multidimensional Theta Functions
22 Jacobian Elliptic Functions
23 Weierstrass Elliptic and Modular Functions
24 Bernoulli and Euler Polynomials
25 Zeta and Related Functions
26 Combinatorial Analysis
27 Functions of Number Theory
28 Mathieu Functions and Hill's Equation
29 Lamé Functions
30 Spheroidal Wave Functions
31 Heun Functions
32 Painlevé Transcendents
33 Coulomb Functions
34 $3j, 6j, 9j$ Symbols
35 Functions of Matrix Argument
36 Integrals with Coalescing Saddles
Bibliography
Index
Notations
Software
Errata

Most functions can be evaluated over $\mathbb{C}$

Many functions can be evaluated over $\mathbb{C}[[x]]/\langle x^n \rangle$

Generalized hypergeometric functions

$$ {}_pF_q(a_1 \dots a_p; b_1 \dots b_q; z) = \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_p)_k}{(b_1)_k \cdots (b_q)_k} \frac{z^k}{k!} $$

$$ (a)_k = a(a+1)(a+2)\cdots(a+k-1) $$

$$ S \pm R \qquad \underbrace{S = \sum_{k=0}^{N-1} T(k)}_{\text{Using interval arithmetic}} \qquad \underbrace{\left| \sum_{k=N}^{\infty} T(k) \right| \leq R}_{\text{Upper bound}} $$

Generalized hypergeometric functions

$$ {}_pF_q(a_1 \ldots a_p; b_1 \ldots b_q; z) = \sum_{k=0}^{\infty} \frac{(a_1)_k \cdots (a_p)_k}{(b_1)_k \cdots (b_q)_k} \frac{z^k}{k!} $$

$$ (a)_k = a(a+1)(a+2)\cdots(a+k-1) $$

$$ S \pm R \qquad \underbrace{S = \sum_{k=0}^{N-1} T(k)}_{\text{Using interval arithmetic}} \qquad \underbrace{\left| \sum_{k=N}^{\infty} T(k) \right| \leq R}_{\text{Upper bound}} $$

Evaluation supported for $a_i, b_i, z \in \mathbb{C}[[x]]/\langle x^n \rangle$ (when convergent)

Error bounds for the *divergent* asymptotic series ${}_2F_0(a, b, z)$ with $a, b, z \in \mathbb{C}$ based on Olver (DLMF 13.7).

Special cases

Error function, exponential, trigonometric, logarithmic integrals

$$\mathrm{erf}(z), \mathrm{erfc}(z), \mathrm{Ei}(z), \mathrm{Si}(z), \mathrm{Ci}(z), \mathrm{Shi}(z), \mathrm{Chi}(z), \mathrm{li}(z)$$

Incomplete gamma function

$$\Gamma(s, z) = \int_z^\infty t^{s-1} e^{-t} dt$$

Bessel functions

$$J_\nu(z), Y_\nu(z), I_\nu(z), K_\nu(z)$$

Others (to be done): Legendre functions, incomplete beta function, ...

Example: modified Bessel function of the second kind

Case 1: $|z| \approx \infty$: asymptotic series

$$K_a(z) = \left(\frac{\pi}{2z}\right)^{1/2} e^{-z} U^*(a + \tfrac{1}{2}, 2a + 1, 2z), \quad U^* \sim {}_2F_0(\ldots, -\tfrac{1}{2z})$$

Example: modified Bessel function of the second kind

Case 1: $|z| \approx \infty$: asymptotic series

$$K_a(z) = \left(\frac{\pi}{2z}\right)^{1/2} e^{-z} U^*(a + \tfrac{1}{2}, 2a + 1, 2z), \quad U^* \sim {}_2F_0(\ldots, -\tfrac{1}{2z})$$

Case 2: $|z| \approx 0$ and $a \notin \mathbb{Z}$: convergent series

$$K_a(z) = \frac{1}{2}\frac{\pi}{\sin(\pi a)}\left[\left(\frac{z}{2}\right)^{-a} {}_0\widetilde{F}_1\left(1 - a, \frac{z^2}{4}\right) - \left(\frac{z}{2}\right)^{a} {}_0\widetilde{F}_1\left(1 + a, \frac{z^2}{4}\right)\right]$$

Example: modified Bessel function of the second kind

Case 1: $|z| \approx \infty$: asymptotic series

$$K_a(z) = \left(\frac{\pi}{2z}\right)^{1/2} e^{-z} U^*(a+\tfrac{1}{2}, 2a+1, 2z), \quad U^* \sim {}_2F_0(\ldots, -\tfrac{1}{2z})$$

Case 2: $|z| \approx 0$ and $a \notin \mathbb{Z}$: convergent series

$$K_a(z) = \frac{1}{2}\frac{\pi}{\sin(\pi a)}\left[\left(\frac{z}{2}\right)^{-a} {}_0\widetilde{F}_1\left(1-a, \frac{z^2}{4}\right) - \left(\frac{z}{2}\right)^{a} {}_0\widetilde{F}_1\left(1+a, \frac{z^2}{4}\right)\right]$$

Case 3: $|z| \approx 0$ and $a \in \mathbb{Z}$: parameter limit

$$\lim_{\varepsilon \to 0} K_{a+\varepsilon}(z)$$

as in (Case 2), but with $\mathbb{C}[[\varepsilon]]/\langle\varepsilon^2\rangle$ arithmetic

Remaining difficulties

- Optimal algorithm selection
- Accurate error bounds when there is cancellation
- Asymptotic expansions (in general)
- Complete handling of ${}_2F_1$, ${}_3F_2$, ...

Elliptic and modular functions

Arithmetic-geometric mean: $\mathrm{agm}(z_1, z_2)$

Jacobi theta functions: $\theta_i(z, \tau)$

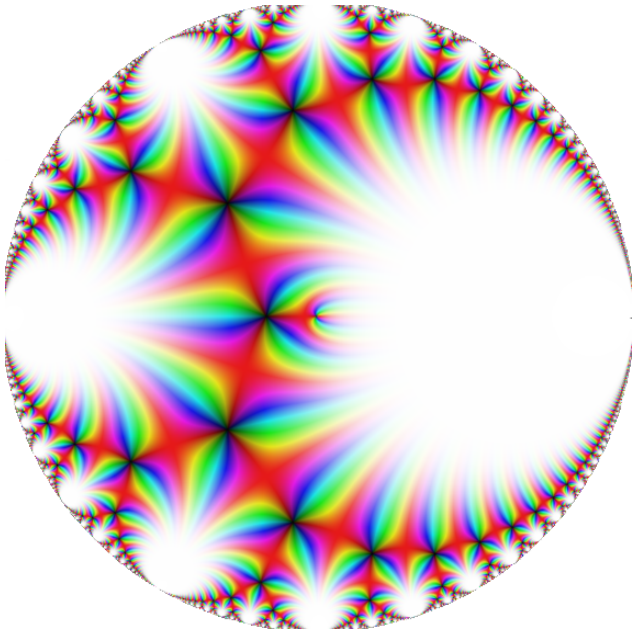
Modular forms and functions: $\eta(\tau), j(\tau), \lambda(\tau), \Delta(\tau), G_{2k}(\tau)$

Weierstrass elliptic function: $\wp(z, \tau)$

Complete elliptic integrals: $K(z), E(z)$

In all cases, for $z \in \mathbb{C}[[x]]/\langle x^n \rangle$ and $\tau \in \mathbb{H}$

Pictures of $j(\tau)$



As a function of $\tau \in [-2, 2] + [0, 1]i$ (top) and of q (bottom).

Pictures of $j(\tau)$

Deep zoom: $\tau \in [\sqrt{13}, \sqrt{13} + 10^{-101}] + [0, 2.5 \times 10^{-102}]i$

Example: Hilbert class polynomials

The quadratic forms with discriminant $D = -31$ are

$$x^2 + xy + 8y^2, \quad 2x^2 + xy + 4y^2, \quad 2x^2 - xy + 4y^2$$

Example: Hilbert class polynomials

The quadratic forms with discriminant $D = -31$ are

$$x^2 + xy + 8y^2, \quad 2x^2 + xy + 4y^2, \quad 2x^2 - xy + 4y^2$$

Therefore $H_{-31} = (x - j_1)(x - j_2)(x - j_3)$ where

$$j_1 = j\left(\frac{-1+\sqrt{-31}}{2}\right), \quad j_2 = j\left(\frac{-1+\sqrt{-31}}{4}\right), \quad j_3 = \bar{j_2} = j\left(\frac{+1+\sqrt{-31}}{4}\right)$$

Example: Hilbert class polynomials

The quadratic forms with discriminant $D = -31$ are

$$x^2 + xy + 8y^2, \quad 2x^2 + xy + 4y^2, \quad 2x^2 - xy + 4y^2$$

Therefore $H_{-31} = (x - j_1)(x - j_2)(x - j_3)$ where

$$j_1 = j\left(\frac{-1+\sqrt{-31}}{2}\right), \quad j_2 = j\left(\frac{-1+\sqrt{-31}}{4}\right), \quad j_3 = \bar{j_2} = j\left(\frac{+1+\sqrt{-31}}{4}\right)$$

Using interval arithmetic with 73 bits of precision, we compute

$$\begin{aligned} j_1 &= [-39492793.91155624414 \pm 6.10 \cdot 10^{-12}] \\ j_2 &= [743.455778122071940 \pm 3.22 \cdot 10^{-16}] \\ &\quad + [6253.062846903285089 \pm 8.87 \cdot 10^{-16}]i \end{aligned}$$

Example: Hilbert class polynomials

The quadratic forms with discriminant $D = -31$ are

$$x^2 + xy + 8y^2, \quad 2x^2 + xy + 4y^2, \quad 2x^2 - xy + 4y^2$$

Therefore $H_{-31} = (x - j_1)(x - j_2)(x - j_3)$ where

$$j_1 = j\left(\frac{-1+\sqrt{-31}}{2}\right), \quad j_2 = j\left(\frac{-1+\sqrt{-31}}{4}\right), \quad j_3 = \bar{j_2} = j\left(\frac{+1+\sqrt{-31}}{4}\right)$$

Using interval arithmetic with 73 bits of precision, we compute

$$\begin{aligned} j_1 &= [-39492793.91155624414 \pm 6.10 \cdot 10^{-12}] \\ j_2 &= [743.455778122071940 \pm 3.22 \cdot 10^{-16}] \\ &\quad + [6253.062846903285089 \pm 8.87 \cdot 10^{-16}]i \end{aligned}$$

Expanding gives $H_{-31} = x^3 + c_2x^2 + c_1x + c_0$ where

$$\begin{aligned} c_2 &= [39491307.00000000000 \pm 2.44 \cdot 10^{-12}] \\ c_1 &= [-58682638134.0000000 \pm 1.61 \cdot 10^{-8}] \\ c_0 &= [1566028350940383.000 \pm 3.22 \cdot 10^{-4}] \end{aligned}$$

Some benchmark results

Sage: complex analytic (floating-point)

classpoly: CRT method, by Andrew Sutherland

Pari: CRT method, by Hamish Ivey-Law

CM: complex analytic (floating-point), by Andreas Enge

Arb: complex analytic (interval arithmetic), by FJ

$-D$	deg	bits	Sage	classpoly	Pari	CM	Arb
1 000 003	105	8527	2.1 s	0.8 s	12 s	0.7 s	0.3 s
10 000 003	706	50889	601 s	8 s	194 s	101 s	45 s
100 000 003	1702	153095		82 s	1855 s	1822 s	680 s

Theta and eta series

$$\theta_2(\tau) = e^{\pi i \tau/4} \sum_{k=-\infty}^{\infty} q^{k(k+1)} = 2e^{\pi i \tau/4}(1 + q^2 + q^6 + q^{12} + q^{20} + \ldots)$$

$$\theta_3(\tau) = \sum_{k=-\infty}^{\infty} q^{k^2} = 1 + 2q + 2q^4 + 2q^9 + 2q^{16} + \ldots$$

$$\theta_4(\tau) = \sum_{k=-\infty}^{\infty} (-1)^k q^{k^2} = 1 - 2q + 2q^4 - 2q^9 + 2q^{16} - \ldots$$

$$\begin{aligned}\eta(\tau) &= e^{\pi i \tau/12} \sum_{k=-\infty}^{\infty} (-1)^k q^{(3k^2-k)/2} \\ &= e^{\pi i \tau/12} \left(1 - q - q^2 + q^5 + q^7 - q^{12} - q^{15} + \ldots\right)\end{aligned}$$

Computing theta and eta functions efficiently

Previously (in Andreas Enge's talk): computing n nonzero terms of a theta or eta series using $n + o(n)$ multiplications.

Computing theta and eta functions efficiently

Previously (in Andreas Enge's talk): computing n nonzero terms of a theta or eta series using $n + o(n)$ multiplications.

Improvement: for any integer-valued quadratic polynomial $F(X)$,

$$\sum_{i=0}^{n} q^{F(i)}$$

can be computed using

$$O(n/\log^r n)$$

multiplications, for any $r > 0$.

Rectangular splitting

This is a method for evaluating *dense* series:

$$\sum_{k=0}^{N} \square q^k =$$

$$\begin{aligned}
&(\square + \square q + \square q^2 + \ldots + \square q^{m-1}) \\
+q^m&(\square + \square q + \square q^2 + \ldots + \square q^{m-1}) \\
+q^{2m}&(\square + \square q + \square q^2 + \ldots + \square q^{m-1}) \\
+q^{3m}&(\square + \square q + \square q^2 + \ldots + \square q^{m-1}) \\
&\vdots
\end{aligned}$$

Cost is $m + N/m$ multiplications, or $O(N^{1/2})$ with $m \sim N^{1/2}$.

No improvement for our sparse series with $n = O(N^{1/2})$ terms.

Rectangular splitting

Idea: choose m such that $F(X)$ takes few distinct values mod m.

Consider $F(X) = X^2$ and

$$s(m) = \text{number of squares mod } m$$

Rectangular splitting

Idea: choose m such that $F(X)$ takes few distinct values mod m.

Consider $F(X) = X^2$ and

$$s(m) = \text{number of squares mod } m$$

We need $O(s(m) \log m + N/m)$ multiplications, where we want m large and $s(m)$ small.

Rectangular splitting

Idea: choose m such that $F(X)$ takes few distinct values mod m.

Consider $F(X) = X^2$ and

$$s(m) = \text{number of squares mod } m$$

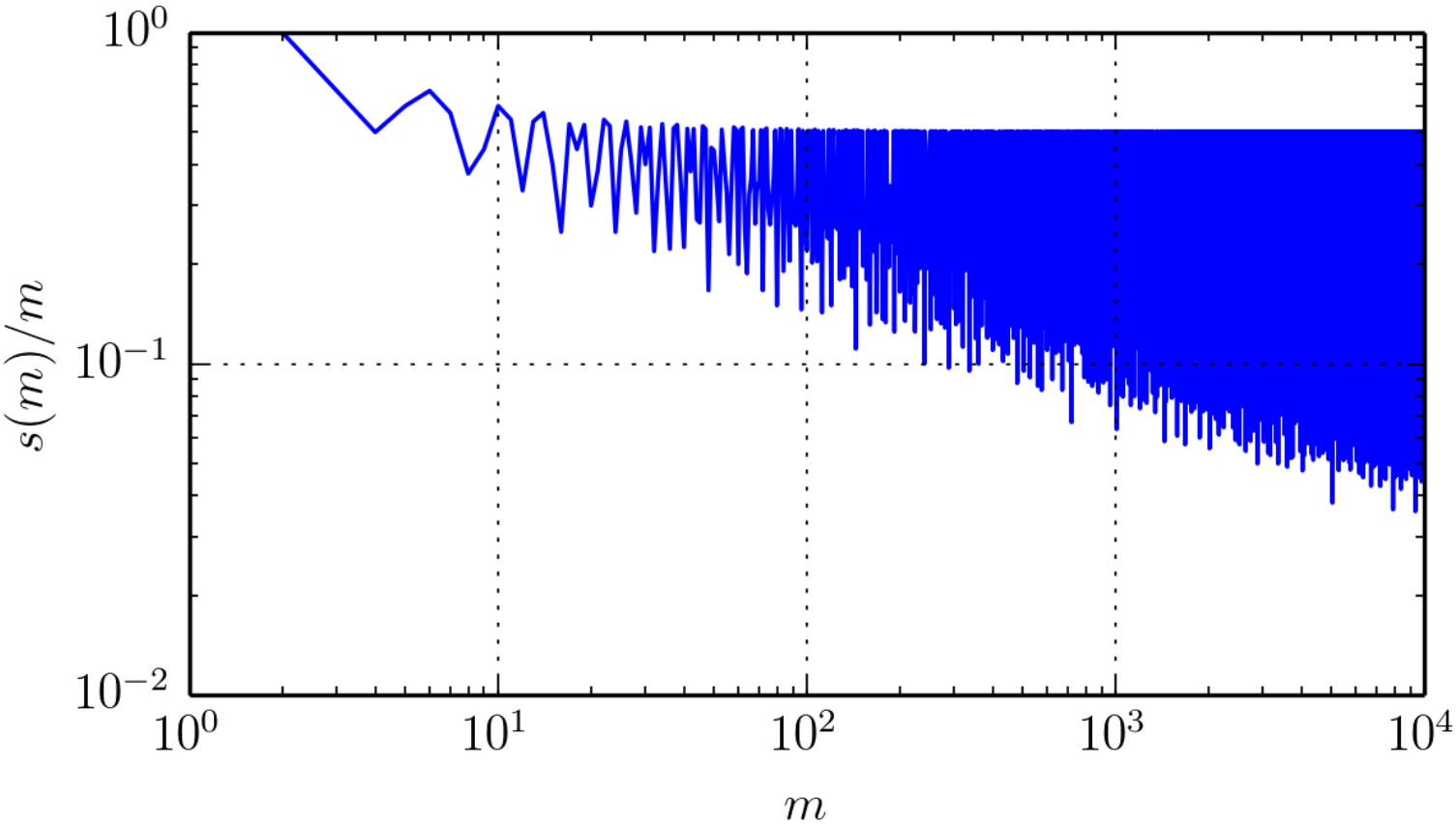
We need $O(s(m) \log m + N/m)$ multiplications, where we want m large and $s(m)$ small.

This suggests looking for m such that

$$\frac{s(m)}{m}$$

is small.

Successive minima



The m such that $s(m)/m < s(m')/m'$ for all $m' < m$ are a good choice.

k	$m = \text{A085635}(k)$	$s(m) = \text{A084848}(k)$	$s(m)/m$
1	$2 = 2$	2	1.0
2	$3 = 3$	2	0.67
3	$4 = 2^2$	2	0.50
4	$8 = 2^3$	3	0.38
5	$12 = 2^2 \cdot 3$	4	0.33
6	$16 = 2^4$	4	0.25
7	$32 = 2^5$	7	0.22
8	$48 = 2^4 \cdot 3$	8	0.17
9	$80 = 2^4 \cdot 5$	12	0.15
10	$96 = 2^5 \cdot 3$	14	0.15
11	$112 = 2^4 \cdot 7$	16	0.14
12	$144 = 2^4 \cdot 3^2$	16	0.11
13	$240 = 2^4 \cdot 3 \cdot 5$	24	0.10
14	$288 = 2^5 \cdot 3^2$	28	0.097
15	$336 = 2^4 \cdot 3 \cdot 7$	32	0.095
16	$480 = 2^5 \cdot 3 \cdot 5$	42	0.088

k	m	$s(m)$	$s(m)/m$
17	$560 = 2^4 \cdot 5 \cdot 7$	48	0.086
18	$576 = 2^6 \cdot 3^2$	48	0.083
19	$720 = 2^4 \cdot 3^2 \cdot 5$	48	0.067
20	$1008 = 2^4 \cdot 3^2 \cdot 7$	64	0.063
21	$1440 = 2^5 \cdot 3^2 \cdot 5$	84	0.058
22	$1680 = 2^4 \cdot 3 \cdot 5 \cdot 7$	96	0.057
23	$2016 = 2^5 \cdot 3^2 \cdot 7$	112	0.056
24	$2640 = 2^4 \cdot 3 \cdot 5 \cdot 11$	144	0.055
25	$2880 = 2^6 \cdot 3^2 \cdot 5$	144	0.050
26	$3600 = 2^4 \cdot 3^2 \cdot 5^2$	176	0.049
27	$4032 = 2^6 \cdot 3^2 \cdot 7$	192	0.048
28	$5040 = 2^4 \cdot 3^2 \cdot 5 \cdot 7$	192	0.038
	$\vdots$		
94	$41801760 = 2^5 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 29$	211680	0.0051
95	$42325920 = 2^5 \cdot 3^2 \cdot 5 \cdot 7 \cdot 13 \cdot 17 \cdot 19$	211680	0.0050
96	$48454560 = 2^5 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 19 \cdot 23$	241920	0.0050
97	$49008960 = 2^6 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	217728	0.0044
98	$54774720 = 2^6 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 19$	241920	0.0044
99	$61261200 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 17$	266112	0.0043
100	$68468400 = 2^4 \cdot 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 19$	295680	0.0043

The function $s(m)$ is multiplicative, and takes the values

$$s(m) = \begin{cases} \frac{1}{2}p^e - \frac{1}{2}p^{e-1} + \frac{p^{e-1} - p^{(e+1) \bmod 2}}{2(p+1)} + 1 & \text{for } p \text{ odd;} \\ 2 & \text{for } p = 2 \text{ and } e \leq 2; \\ 2^{e-3} + \frac{2^{e-3} - 2^{(e+1) \bmod 2}}{3} + 2 & \text{for } p = 2 \text{ and } e \geq 3, \end{cases}$$

at prime powers $m = p^e$.

The function $s(m)$ is multiplicative, and takes the values

$$s(m) = \begin{cases} \frac{1}{2}p^e - \frac{1}{2}p^{e-1} + \frac{p^{e-1} - p^{(e+1) \bmod 2}}{2(p+1)} + 1 & \text{for } p \text{ odd;} \\ 2 & \text{for } p = 2 \text{ and } e \leq 2; \\ 2^{e-3} + \frac{2^{e-3} - 2^{(e+1) \bmod 2}}{3} + 2 & \text{for } p = 2 \text{ and } e \geq 3, \end{cases}$$

at prime powers $m = p^e$.

Minimizing $s(m) \log m + N/m$ under the assumption that m is a product of distinct primes gives the bound in the theorem.

The function $s(m)$ is multiplicative, and takes the values

$$s(m) = \begin{cases} \frac{1}{2}p^e - \frac{1}{2}p^{e-1} + \frac{p^{e-1} - p^{(e+1) \bmod 2}}{2(p+1)} + 1 & \text{for } p \text{ odd;} \\ 2 & \text{for } p = 2 \text{ and } e \leq 2; \\ 2^{e-3} + \frac{2^{e-3} - 2^{(e+1) \bmod 2}}{3} + 2 & \text{for } p = 2 \text{ and } e \geq 3, \end{cases}$$

at prime powers $m = p^e$.

Minimizing $s(m) \log m + N/m$ under the assumption that m is a product of distinct primes gives the bound in the theorem.

The construction is analogous for other quadratic polynomials.

Successive minima for trigonal numbers ($k(k+1)$)

k	m	$t(m)$	$t(m)/m$
1	$2 = 2$	1	0.50
2	$6 = 2 \cdot 3$	2	0.33
3	$10 = 2 \cdot 5$	3	0.30
4	$14 = 2 \cdot 7$	4	0.29
5	$18 = 2 \cdot 3^2$	4	0.22
6	$30 = 2 \cdot 3 \cdot 5$	6	0.20
7	$42 = 2 \cdot 3 \cdot 7$	8	0.19
8	$66 = 2 \cdot 3 \cdot 11$	12	0.18
9	$70 = 2 \cdot 5 \cdot 7$	12	0.17
10	$90 = 2 \cdot 3^2 \cdot 5$	12	0.13
	$\vdots$		
100	$25160850 = 2 \cdot 3^2 \cdot 5^2 \cdot 11 \cdot 13 \cdot 17 \cdot 23$	199584	0.0079
101	$25675650 = 2 \cdot 3^3 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 19$	203280	0.0079
102	$28120950 = 2 \cdot 3^2 \cdot 5^2 \cdot 11 \cdot 13 \cdot 19 \cdot 23$	221760	0.0079
103	$29099070 = 2 \cdot 3^2 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19$	181440	0.0062

Successive minima for pentagonal numbers

k	m	$p(m)$	$p(m)/m$
1	$2 = 2$	2	1.0
2	$5 = 5$	3	0.60
3	$7 = 7$	4	0.57
4	$11 = 11$	6	0.55
5	$13 = 13$	7	0.54
6	$17 = 17$	9	0.53
7	$19 = 19$	10	0.53
8	$23 = 23$	12	0.52
9	$25 = 5^2$	11	0.44
10	$35 = 5 \cdot 7$	12	0.34
	$\vdots$		
100	$4555915 = 5 \cdot 7 \cdot 13 \cdot 17 \cdot 19 \cdot 31$	120960	0.027
101	$5159245 = 5 \cdot 7 \cdot 13 \cdot 17 \cdot 23 \cdot 29$	136080	0.026
102	$5311735 = 5 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23$	136080	0.026
103	$6697405 = 5 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 29$	170100	0.025

Example: computing θ_3

Suppose we want to compute

$$1 + 2\sum_{k=1}^{n} q^{k^2} \approx 1 + \sum_{k=1}^{\infty} 2q^{k^2}$$

for $q = \exp(-\pi)$, with n such that the error is less than 2^{-B}

Example: computing θ_3

Suppose we want to compute

$$1 + 2\sum_{k=1}^{n} q^{k^2} \approx 1 + \sum_{k=1}^{\infty} 2q^{k^2}$$

for $q = \exp(-\pi)$, with n such that the error is less than 2^{-B}

B	n	$\#(n^2)$	m	$s(m)$	$\#(\text{mod } m)$	$\#(\text{tot})$	Speedup
10^3	14	23	48	8	12	16	1.44
10^4	46	71	144	16	23	37	1.92
10^5	148	228	720	48	57	87	2.62
10^6	469	690	1680	96	109	239	2.89
10^7	1485	2098	10080	336	356	574	3.66

$\#(n^2)$: number of additions to generate $1, 4, 9, \ldots, n^2$
$\#(\text{mod } m)$: number of additions to generate $1, 4, 9, \ldots \text{ mod } m$
$\#(\text{tot})$: total multiplications in the rectangular splitting algorithm