COMPUTING LOGARITHM INTERVALS WITH THE ARITHMETIC-GEOMETRIC-MEAN ITERATION

DANIEL J. BERNSTEIN

Abstract. This paper presents a fast algorithm that, given a tight interval around a positive real number x, computes a tight interval around $\log x$. To obtain p bits of precision for typical values of x, the algorithm uses about $2\lg p$ square roots and about $5\lg p$ multiplications (or fewer for subsequent logarithms). Here log is the natural logarithm, and lg is the base-2 logarithm. This paper also presents short proofs of all necessary properties of complete elliptic integrals.

1. Introduction

This paper presents an algorithm that, given a positive real number x to high accuracy, computes $\log x$ to high accuracy. Here log is the natural logarithm. The algorithm has several useful features:

- It is extremely fast. For p bits of precision, and for x between (e.g.) 2 and 2^p, the algorithm time is dominated by about $7\lg p$ operations on p-bit numbers, specifically $2\lg p$ square roots and $5\lg p$ multiplications. Here lg is the base-2 logarithm.
- It computes π to high precision, practically for free, as a side effect of computing log.
- It computes subsequent logarithms at even higher speed: asymptotically, per logarithm, about $2\lg p$ square roots and about $2\lg p$ multiplications.
- Given a tight *interval* containing x, it computes tight intervals containing $\log x$ and π. See [4] for an application. For large p, the extra cost of handling intervals is negligible.

See Section 5 for details of the algorithm.

The algorithm relies on various properties of complete elliptic integrals and the arithmetic-geometric-mean (AGM) iteration. This paper presents streamlined proofs of those properties. Section 2 introduces the elliptic integrals I and I_1 used throughout the paper; Section 3 relates I and I_1 to logarithms; Section 4 introduces the AGM iteration.

Previous work. Salamin pointed out more than thirty years ago that the elliptic integral I could be used to quickly compute high-precision approximations to $\log x$ and π; see [2, Item 143]. Salamin's algorithm is dominated by $\Theta(\lg p)$ high-precision operations, with larger constant factors than the algorithm here.

Permanent document ID: `8f92b1e3ec7918d37b28b9efcee5e97f`.

Date: 20030717.

2000 *Mathematics Subject Classification.* Primary 65D20; Secondary 33E05, 65G30, 11Y60.

The author was supported by the National Science Foundation under grant DMS–0140542, and by the Alfred P. Sloan Foundation.

Subsequent refinements of Salamin's algorithm included almost all of the ideas necessary for minimizing the constant factors, but the ideas were never combined properly. See Section 6 for a survey of the literature.

Brent in [9, Section 6] introduced another method of computing high-precision exponentials and logarithms, without relying on elliptic integrals. See [3, Sections 12–16] for further discussion. Brent's method may be faster than the algorithm here for small p, but for large p it is slower by a factor roughly proportional to $\lg p$.

Logarithm algorithms are occasionally presented with explicit bounds suitable for interval computation. See [14] and [13].

The results of Sections 2, 3, and 4 (modulo language, and modulo the difference between explicit inequalities and order-of-magnitude estimates) have been known for two centuries, and are a small fraction of the wealth of material collected in [8]. However, I have not seen such short proofs of the I_1 properties, particularly Theorem 2.6 and Theorem 4.3.

2. Elliptic integrals: basic properties

For positive real numbers a, b define $I(a,b) = \int_0^\infty (x^2+a^2)^{-1/2}(x^2+b^2)^{-1/2}\,dx$. Also define $I_1(a,b) = \partial I(a,b)/\partial a = \int_0^\infty -a(x^2+a^2)^{-3/2}(x^2+b^2)^{-1/2}\,dx$.

Integrability, differentiation under the integral sign, etc. follow from the fact that all the integrands in this section are constant in sign, continuous in x, and differentiable in a, b.

Theorem 2.1. *$\pi/2a \le I(a,b) \le \pi/2b$ if $b \le a$.*

Proof. $x^2+b^2 \le x^2+a^2$ so $\int_0^\infty (x^2+a^2)^{-1}\,dx \le I(a,b) \le \int_0^\infty (x^2+b^2)^{-1}\,dx$. □

Theorem 2.2. *$0 \le -aI_1(a,b) \le I(a,b)$.*

Proof. $0 \le a^2 \le x^2+a^2$. □

Theorem 2.3. *$I(a,b) = I(1,b/a)/a$.*

Proof. Substitute $x = au$: $aI(a,b) = \int_0^\infty a^2(a^2u^2+a^2)^{-1/2}(a^2u^2+b^2)^{-1/2}\,du = \int_0^\infty (u^2+1)^{-1/2}(u^2+b^2/a^2)^{-1/2}\,du = I(1,b/a)$. □

Theorem 2.4. *$I(a,b) + aI_1(a,b) + b(\partial I(a,b))/\partial b = 0$.*

Proof. Apply $a(\partial/\partial a) + b(\partial/\partial b)$ to Theorem 2.3. □

Theorem 2.5. *$I(a,b) = I(\frac{a+b}{2}, \sqrt{ab})$.*

Proof. Substitute $u = (x - ab/x)/2$, using $(2u)^2 + (a+b)^2 = (x^2+a^2)(x^2+b^2)/x^2$ and $x\,du = (u^2+ab)^{1/2}\,dx$: $I(a,b) = \int_{-\infty}^\infty ((2u)^2+(a+b)^2)^{-1/2}(u^2+ab)^{-1/2}\,du = 2\int_0^\infty (2^2(u^2+(\frac{a+b}{2})^2))^{-1/2}(u^2+ab)^{-1/2}\,du = I(\frac{a+b}{2}, \sqrt{ab})$. □

Theorem 2.6. *$I(a,b) + 2aI_1(a,b) = I_1(\frac{a+b}{2}, \sqrt{ab})(a-b)/2$.*

Proof. Apply $a(\partial/\partial a) - b(\partial/\partial b)$ to Theorem 2.5: $aI_1(a,b) - b(\partial I(a,b)/\partial b) = I(\frac{a+b}{2}, \sqrt{ab})(a-b)/2$. By Theorem 2.4, $I(a,b) + aI_1(a,b) + b(\partial I(a,b)/\partial b) = 0$. □

Theorem 2.7. *$I(1,b) = 2\int_0^{\sqrt{b}} (x^2+1)^{-1/2}(x^2+b^2)^{-1/2}\,dx$.*

Proof. $\int_{\sqrt{b}}^\infty (x^2+1)^{-1/2}(x^2+b^2)^{-1/2}\,dx = \int_0^{\sqrt{b}} ((\frac{b}{u})^2+1)^{-1/2}((\frac{b}{u})^2+b^2)^{-1/2}\frac{b}{u^2}\,du = \int_0^{\sqrt{b}} (b^2+u^2)^{-1/2}(1+u^2)^{-1/2}\,du$. □

Theorem 2.8. $(1+b)^{-1}+I(1,b)+I_1(1,b)=2\int_0^{\sqrt{b}} b^2(x^2+1)^{-1/2}(x^2+b^2)^{-3/2}\,dx.$

Proof. By Theorem 2.4, $I(1,b)+I_1(1,b)+b(\partial I(1,b)/\partial b)=0$. By Theorem 2.7, $\partial I(1,b)/\partial b = b^{-1/2}(b+1)^{-1/2}(b+b^2)^{-1/2}+2\int_0^{\sqrt{b}} -b(x^2+1)^{-1/2}(x^2+b^2)^{-3/2}\,dx.$ □

Theorem 2.9. $a^2I(a,b)+(a^2-b^2)aI_1(a,b)=\int_0^{\infty} a^2(x^2+a^2)^{-3/2}(x^2+b^2)^{1/2}\,dx.$

Proof. $x^2+a^2-(a^2-b^2)=x^2+b^2.$ □

Theorem 2.9 explains why $I(a,b)$, $a^2I(a,b)+(a^2-b^2)aI_1(a,b)$, etc. are called **elliptic integrals**, and why related algebraic objects are called **elliptic curves**: substitute $x=a\tan\theta$ to see that $a^2I(a,b)+(a^2-b^2)aI_1(a,b)$ is the arc length of one quadrant of the ellipse $\theta\mapsto(a\cos\theta,b\sin\theta)$. Theorem 2.9 is not used elsewhere in this paper.

3. Elliptic integrals: logarithm bounds

This section presents precise bounds along the following lines: "If $b\approx 0$ then $I(1,b)\approx\log(4/b)$ and $I(1,b)+I_1(1,b)\approx 1$." Write $L(b)=\log(\sqrt{b^{-1}}+\sqrt{b^{-1}+1})$.

The simple proof technique used here is not new, and explicit bounds in this context are not new, but I am not aware of previous simple proofs of explicit bounds. For inexplicit bounds using the same proof technique, see [18, page 522] and [15]; for explicit bounds with longer proofs, see [7, pages 355–356] and [8, Theorem 7.2].

Theorem 3.1. *If* $0<b\le 1$ *then* $(2+\frac{1}{2}b^2)L(b)-(\frac{1}{2}b)(1+b)^{1/2}\le I(1,b)\le (2+\frac{1}{2}b^2+\frac{9}{32}b^4)L(b)-(\frac{1}{2}b-\frac{3}{16}b^2+\frac{9}{32}b^3)(1+b)^{1/2}$.

The difference between lower and upper bounds is on the scale of b^2. The bounds can easily be made tighter by the same technique.

Proof. $I(1,b)=2\int_0^{\sqrt{b}}(1+x^2)^{-1/2}(x^2+b^2)^{-1/2}\,dx$ by Theorem 2.7. Recall that $(1+x^2)^{-1/2}\ge 1-x^2/2$ for $0\le x\le 1$, since $1\ge 1-(3-x^2)x^4/4=(1-x^2/2)^2(1+x^2)$. Thus

$$\begin{aligned}
I(1,b) &\ge \int_0^{\sqrt{b}}(2-x^2)(x^2+b^2)^{-1/2}\,dx\\
&= (2+\tfrac{1}{2}b^2)\log(x+(x^2+b^2)^{1/2})-(\tfrac{1}{2}x)(x^2+b^2)^{1/2}\Big|_0^{\sqrt{b}}\\
&= (2+\tfrac{1}{2}b^2)\log\left(\frac{\sqrt{b}+\sqrt{b+b^2}}{\sqrt{0}+\sqrt{0+b^2}}\right)-(\tfrac{1}{2}\sqrt{b})(b+b^2)^{1/2}\\
&= (2+\tfrac{1}{2}b^2)L(b)-(\tfrac{1}{2}b)(1+b)^{1/2}.
\end{aligned}$$

Similarly, $(1+x^2)^{-1/2}\le 1-\frac{1}{2}x^2+\frac{3}{8}x^4$ for $0\le x$, so

$$\begin{aligned}
I(1,b) &\le \int_0^{\sqrt{b}}(2-x^2+\tfrac{3}{4}x^4)(x^2+b^2)^{-1/2}\,dx\\
&= (2+\tfrac{1}{2}b^2+\tfrac{9}{32}b^4)\log(x+(x^2+b^2)^{1/2})-(\tfrac{1}{2}-\tfrac{3}{16}x^2+\tfrac{9}{32}b^2)x(x^2+b^2)^{1/2}\Big|_0^{\sqrt{b}}\\
&= (2+\tfrac{1}{2}b^2+\tfrac{9}{32}b^4)L(b)-(\tfrac{1}{2}b-\tfrac{3}{16}b^2+\tfrac{9}{32}b^3)(1+b)^{1/2}.
\end{aligned}$$

□

Theorem 3.2. *If* $0 < b \le 1$ *then* $(2+b^2)(1+b)^{-1/2} - b^2 L(b) \le (1+b)^{-1} + I(1,b) + I_1(1,b) \le (2 + b^2 + \frac{3}{8}b^3 + \frac{9}{8}b^4)(1+b)^{-1/2} - (b^2 + \frac{9}{8}b^4)L(b)$.

Proof. $(1+b)^{-1}+I(1,b)+I_1(1,b) = 2\int_0^{\sqrt{b}} b^2(x^2+1)^{-1/2}(x^2+b^2)^{-3/2}\,dx$ by Theorem 2.8. Recall that $(1+x^2)^{-1/2} \ge 1 - x^2/2$ for $0 \le x \le 1$, as in Theorem 3.1, so

$$\begin{aligned}(1+b)^{-1} + I(1,b) + I_1(1,b) &\ge \int_0^{\sqrt{b}} b^2(2-x^2)(x^2+b^2)^{-3/2}\,dx \\ &= (2+b^2)x(x^2+b^2)^{-1/2} - b^2\log(x + (x^2+b^2)^{1/2})\Big|_0^{\sqrt{b}} \\ &= (2+b^2)(1+b)^{-1/2} - b^2 L(b).\end{aligned}$$

Similarly, $(1+x^2)^{-1/2} \le 1 - \frac{1}{2}x^2 + \frac{3}{8}x^4$ for $0 \le x$, so

$$\begin{aligned}(1+b)^{-1} &+ I(1,b) + I_1(1,b) \le \int_0^{\sqrt{b}} b^2(2 - x^2 + \tfrac{3}{4}x^4)(x^2+b^2)^{-3/2}\,dx \\ &= (2 + b^2 + \tfrac{3}{8}b^2x^2 + \tfrac{9}{8}b^4)x(x^2+b^2)^{-1/2} - (b^2 + \tfrac{9}{8}b^4)\log(x + (x^2+b^2)^{1/2})\Big|_0^{\sqrt{b}} \\ &= (2 + b^2 + \tfrac{3}{8}b^3 + \tfrac{9}{8}b^4)(1+b)^{-1/2} - (b^2 + \tfrac{9}{8}b^4)L(b).\end{aligned}$$

□

4. Elliptic integrals: AGM iteration

Throughout this section, a_0 and b_0 are positive real numbers; a_n and b_n are defined recursively by $a_{n+1} = (a_n + b_n)/2$ and $b_{n+1} = \sqrt{a_n b_n}$. As n increases, both a_n and b_n converge rapidly to $\pi/2I(a_0, b_0)$, the **arithmetic-geometric mean** of a_0 and b_0; see Theorems 4.2 and 4.5.

Theorem 4.1. *If* $n \ge 1$ *then* $a_n \ge b_n$.

Proof. The geometric mean cannot exceed the arithmetic mean: if $n \ge 0$ then $a_{n+1}^2 - b_{n+1}^2 = (a_n - b_n)^2/4 \ge 0$ so $a_{n+1} \ge b_{n+1}$. □

Theorem 4.2. $\pi/2a_n \le I(a_0, b_0) \le \pi/2b_n$ *for* $n \ge 1$.

Proof. By Theorem 2.5, $I(a_0,b_0) = I(a_1,b_1) = I(a_2,b_2) = \cdots = I(a_n,b_n)$. By Theorem 4.1, $a_n \ge b_n$. Apply Theorem 2.1. □

Theorem 4.3. *Define* $\epsilon_n = 2^n(a_n^2 - b_n^2)(-a_n(I_1/I)(a_n,b_n))$. *Then* $0 \le \epsilon_n \le 2^n(a_n^2 - b_n^2)$ *for* $n \ge 1$, *and* $(a_0^2 - b_0^2)(-a_0(I_1/I)(a_0,b_0)) = \epsilon_n + \sum_{0 \le i < n} 2^{i-1}(a_i^2 - b_i^2)$.

Proof. $0 < b_n \le a_n$ for $n \ge 1$, so $0 \le -a_n(I_1/I)(a_n,b_n) \le 1$ by Theorem 2.2; i.e., $0 \le \epsilon_n \le 2^n(a_n^2 - b_n^2)$.

Substitute $(a_{n+1}^2 - b_{n+1}^2)a_{n+1} = (a_n^2 - b_n^2)(a_n - b_n)/8$:

$$\epsilon_{n+1} = 2^n(a_n^2 - b_n^2)\left(\frac{-(a_n - b_n)}{4}\left(\frac{I_1}{I}\right)(a_{n+1}, b_{n+1})\right).$$

Then apply Theorems 2.6 and 2.5:

$$\epsilon_{n+1} = 2^n(a_n^2 - b_n^2)\left(\frac{-1}{2} - a_n\left(\frac{I_1}{I}\right)(a_n, b_n)\right) = -2^{n-1}(a_n^2 - b_n^2) + \epsilon_n.$$

Thus $\epsilon_0 = \epsilon_n + \sum_{0 \le i < n} 2^{i-1}(a_i^2 - b_i^2)$ by induction. □

Theorem 4.4. *If* $1 \le a_0/b_0 \le 1 + 2^{2^m}$ *then* $1 \le a_n/b_n \le 1 + 2^{2^{m-n}}$ *for* $n \ge 0$.

Proof. If $1 \le a_n/b_n \le 1 + 2^{2^{m-n}}$ then $a_{n+1}/b_{n+1} = (\sqrt{a_n/b_n} + \sqrt{b_n/a_n})/2 \le \sqrt{a_n/b_n} \le \sqrt{1 + 2^{2^{m-n}}} \le 1 + 2^{2^{m-n-1}}$. □

Theorem 4.5. *If* $m \ge 0$ *and* $1 \le a_0/b_0 \le 1 + 2^{2^m}$ *then* $1 \le a_n/b_n \le 1 + 2^{3-2^{n+1-m}}$ *for* $n \ge m$.

Proof. The base case $n = m$ follows from Theorem 4.4 since $2^{2^{m-m}} = 2^{3-2^{m+1-m}}$.

If $a_n/b_n = 1+\epsilon$ with $0 \le \epsilon \le 2^{3-2^{n+1-m}}$ then $4+4\epsilon+\epsilon^2 \le (4+\epsilon^2+\epsilon^4/16)(1+\epsilon)$, so $(1 + \epsilon) + 2 + 1/(1 + \epsilon) \le (2 + \epsilon^2/4)^2$, so $\sqrt{a_n/b_n} + \sqrt{b_n/a_n} \le 2 + \epsilon^2/4$, so $a_{n+1}/b_{n+1} \le 1 + \epsilon^2/8 \le 1 + 2^{3-2^{n+2-m}}$. □

5. Computing logarithm intervals

This section presents several algorithms that, given an interval containing x, compute an interval containing $\log x$. See [6] for fast subroutines to compute sums, differences, products, quotients, and square roots of intervals.

These algorithms use a parameter p to decide when to stop. For the "arbitrary x" algorithms, the output interval has approximately p bits of precision if the input interval does. For the "super-size x" algorithms, the output precision is limited to about $4 \lg x$ bits, even if p is much larger. The "super-size x" algorithms also slow down as $\lg\lg x$ grows past $\lg p$; if $\lg\lg x$ is much larger than $\lg p$, one should use the "arbitrary x" algorithms instead.

Beware that, as discussed in [5], the usual algorithms for arithmetic operations such as square root—and for sequences of arithmetic operations—contain many redundancies that can be eliminated. A sequence of AGM steps can be made almost three times faster than reported in [11, Theorem 9.1], for example. Note also that Borwein and Borwein in [8, page 222] observed speed improvements from a "quartic AGM" in which one computes $\sqrt{a_{n+2}}$ and $\sqrt{b_{n+2}}$ directly from $\sqrt{a_n}$ and $\sqrt{b_n}$; my impression is that these speedups become slowdowns when the square-root algorithms are properly optimized, but I will leave experiments to the reader.

Computing $\log x$ for a super-size x. Let $x > 4$ be a real number. Define $a_0 = 1$ and $b_0 = b = (2x/(x^2-1))^2$. Note that $\sqrt{b^{-1}}+\sqrt{b^{-1}+1} = x$, so $L(b) = \log x$, where L is defined in Section 3; note also that $0 < b < 1/2$. Define $a_{n+1} = (a_n + b_n)/2$ and $b_{n+1} = \sqrt{a_n b_n}$ for $n \ge 0$, as in Section 4. Define $c = 1 + (I_1/I)(1, b)$; note that $0 \le c \le 1$ by Theorem 2.2.

Compute an interval containing b. Successively compute intervals containing $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$, stopping when $a_n - b_n$ is no longer clearly larger than $1/2^p$. The number of steps n is at most about $\lg\lg x + \lg p$ by Theorem 4.5.

Compute an interval containing $[0, 2^n(a_n^2 - b_n^2)] + \sum_{0\le i<n} 2^{i-1}(a_i^2 - b_i^2)$. By Theorem 4.3, this interval contains $(1-b^2)(1-c)$. Divide by an interval containing $1 - b^2$, and subtract from 1, to obtain an interval containing c. Finally, compute an interval containing

$$\left[\frac{(\frac{1}{2}b - \frac{3}{16}b^2 + \frac{9}{32}b^3)c(1+b)^{1/2} - (1+b)^{-1} + (2+b^2)(1+b)^{-1/2}}{(2+\frac{1}{2}b^2+\frac{9}{32}b^4)c + b^2},\right.$$
$$\left.\frac{(\frac{1}{2}b)c(1+b)^{1/2} - (1+b)^{-1} + (2+b^2+\frac{3}{8}b^3+\frac{9}{8}b^4)(1+b)^{-1/2}}{(2+\frac{1}{2}b^2)c + (b^2+\frac{9}{8}b^4)}\right].$$

By Theorems 3.1 and 3.2, this interval contains $L(b) = \log x$. Note that terms such as $\frac{9}{8}b^4$ have very little effect on the output and can be replaced by crude bounds.

At this point one can also compute an interval containing π with a few additional operations: the interval

$$[(2+\tfrac{1}{2}b^2)L(b) - (\tfrac{1}{2}b)(1+b)^{1/2}, (2+\tfrac{1}{2}b^2+\tfrac{9}{32}b^4)L(b) - (\tfrac{1}{2}b - \tfrac{3}{16}b^2 + \tfrac{9}{32}b^3)(1+b)^{1/2}]$$

contains $I(1, b)$ by Theorem 3.1, and the interval $[2b_n I(1, b), 2a_n I(1, b)]$ contains π by Theorem 4.2. This very fast computation of π will be exploited later.

Numerical stability: The computation of c as $1-(1-c)$ loses about $\lg\log(4/b) \approx \lg\lg x$ bits of precision, since $c \approx 1/\log(4/b)$. The other arithmetic operations are stable, each losing only a bounded number of bits of precision. However, the intervals in Theorems 3.1 and 3.2 are inherently limited to about $\lg(1/b^2) \approx 4\lg x$ bits of precision.

Computing $\log x$ for several super-size x's. After computing a super-size log as explained above, one can compute another super-size log at somewhat higher speed, by taking advantage of the π interval obtained from the first computation.

Starting from x, define and compute $b, a_1, b_1, \ldots, a_n, b_n$ as above, but skip the computation of $\sum_i 2^{i-1}(a_i^2 - b_i^2)$. Compute an interval containing $[\pi/2a_n, \pi/2b_n]$; by Theorem 4.2, this interval contains $I(1, b)$. Compute an interval containing

$$\left[\frac{I(1,b) + (\frac{1}{2}b - \frac{3}{16}b^2 + \frac{9}{32}b^3)(1+b)^{1/2}}{2+\frac{1}{2}b^2+\frac{9}{32}b^4}, \frac{I(1,b) + (\frac{1}{2}b)(1+b)^{1/2}}{2+\frac{1}{2}b^2}\right];$$

by Theorem 3.1, this interval contains $L(b) = \log x$.

A further improvement is available in applications that compute logarithms only to divide them by each other. The above method is to use I_1/I to compute $\log x$ and π, then use I to compute $\pi/\log y$, then divide to obtain $(\log y)/\log x$; it is somewhat faster to use I twice to compute $\pi/\log x$ and $\pi/\log y$, then divide to obtain $(\log y)/\log x$.

Computing $\log x$ for an arbitrary x. Given an interval containing a positive real number x, find an integer k such that $2^k x$ is larger, but not much larger, than $2^{p/4}$. Then use the algorithm above to compute intervals containing the logs of the super-size numbers $2^k x$ and $2^{\lceil p/4\rceil}$. Extract intervals containing $\log 2 = (\log 2^{\lceil p/4\rceil})/\lceil p/4\rceil$ and $\log x = \log 2^k x - k\log 2$.

The time taken by this algorithm is practically independent of x. However, readers trying to formulate a theorem along these lines are cautioned to consider the possibility that k has many digits.

When x is between (for example) 2 and 2^p, the following algorithm is faster: select an integer $m > \lg p - 2 - \lg\lg x$, compute an interval containing the super-size number x^{2^m} by repeated squaring, compute an interval containing $\log x^{2^m}$, and divide by 2^m. Beware that this approach is slow when x is very close to 1.

Computing $\log x$ for several arbitrary x's. The general problem of computing logarithms of ℓ numbers can be reduced to the problem of computing logarithms of $\ell + 1$ super-size numbers, one of which is a power of 2, as in the case $\ell = 1$ above.

When all the numbers are in reasonable ranges, and when ℓ is small, it is faster to repeatedly square each number, obtaining ℓ super-size numbers. However, for large ℓ, this is slower than the power-of-2 method.

6. Previous logarithm algorithms using elliptic integrals

In the following survey, the word "optimal" means "as fast as possible among all the techniques that I know." It is not meant to exclude the possibility of future improvements.

Salamin, as reported in [2, Item 143], proposed using the AGM iteration to compute $\pi/\log x$ for super-size x, and thus to compute $\log x$ using π. This is the optimal strategy when x is super-size and π is already known.

Salamin proposed computing π as follows: compute $\exp 1$ by a different method, then compute $(\exp 1)^{2^m} = \exp 2^m$ by repeated squaring, then use the AGM iteration to compute $\pi/\log \exp 2^m = \pi/2^m$. Schroeppel in [2, Item 144] proposed using 2^{2^m} and $2^{2^m} \exp 1$ instead of $\exp 2^m$. Both methods are considerably slower than optimal.

For x in a reasonable range such as $[2, 4]$, Salamin proposed computing $\log x$ by computing $\log x^{2^m}$. This is the optimal strategy when only one log is to be computed, although it is not optimal when many logs are to be computed.

Brent in [10] proposed a different log algorithm using incomplete elliptic integrals. Brent's algorithm is somewhat slower than optimal: it saves half the AGM steps by focusing on moderate values of b, but it works with more than two values of b.

Brent, and independently Salamin in [16], also proposed using the Legendre-Gauss formula $I(1, 2^{-1/2})(I(1, 2^{-1/2}) + I_1(1, 2^{-1/2})) = \pi/2$ to compute π. This is faster than Salamin's previous method of computing π, but in the context of log computation it is not optimal. The same comment applies to several subsequent methods of computing π, not cited here.

Brent in [11, Section 9] proposed computing $\log x$ for arbitrary x by computing $\log 2^k x$ where k is chosen so that $2^k x$ is super-size. This is the optimal strategy when many logs are to be computed.

Borwein and Borwein in [7, Section 4] proposed another log algorithm in the spirit of [10] but relying solely on complete elliptic integrals. The strategy is somewhat slower than optimal.

Newman in [15] proposed computing π and $\log x$ for super-size x by using the AGM iteration to compute $(\log x)/\pi$, using it again to compute $(\log(x + 1))/\pi \approx (\log x + 1/x)/\pi$, and subtracting. Beware that the subtraction loses about $\lg x$ bits of precision. A better (but still suboptimal) strategy is to use $x \exp 1$ in place of $x + 1$, with $\exp 1$ computed by a different method.

Newman's goal was not maximum speed, but maximum simplicity. In particular, Newman avoided the Legendre-Gauss formula and any other use of I_1. However, I disagree with Newman's view of "Landen's transformation law" (Theorem 2.6 here) as "heavy use of elliptic function theory"; I hope I have convinced the reader that the basic properties of I_1 can be established just as easily as the basic properties of I. (The Legendre-Gauss formula is not much more difficult.)

Borwein and Borwein in [8, Algorithm 7.2] proposed the following method of computing $\log x$ for, e.g., x between 2 and 4: use I_1/I to compute two super-size logarithms, namely $\log 2^k$ and $\log 2^k x$ for an appropriate k. The use of I_1/I is optimal for computing one super-size logarithm, but it is suboptimal for computing two or more.

References

[1] Robert S. Anderssen, Richard P. Brent (editors), *The complexity of computational problem solving*, University of Queensland Press, Brisbane, 1976. ISBN 0–7022–1213–X. Available from `http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pub/pub031.html`.

[2] Michael Beeler, R. William Gosper, Richard Schroeppel, *HAKMEM*, Artificial Intelligence Memo No. 239, Massachusetts Institute of Technology, 1972. Available from `http://www.inwap.com/pdp10/hbaker/hakmem/hakmem.html`.

[3] Daniel J. Bernstein, *Fast multiplication and its applications*. Available from `http://cr.yp.to/papers.html#multapps`. ID `8758803e61822d485d54251b27b1a20d`.

[4] Daniel J. Bernstein, *Computing logarithm floors in essentially linear time*, draft. Available from `http://cr.yp.to/papers.html`.

[5] Daniel J. Bernstein, *Removing redundancy in high-precision Newton iteration*, draft.

[6] Daniel J. Bernstein, *Fast, harmless intervals around high-precision floating-point numbers*, draft.

[7] Jonathan M. Borwein, Peter B. Borwein, *The arithmetic-geometric mean and fast computation of elementary functions*, SIAM Review **26** (1984), 351–366. MR 86d:65029.

[8] Jonathan M. Borwein, Peter B. Borwein, *Pi and the AGM*, Wiley, New York, 1987. ISBN 0–471–83138–7. MR 89a:11134.

[9] Richard P. Brent, *The complexity of multiple-precision arithmetic*, in [1] (1976), 126–165. Available from `http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pub/pub032.html`.

[10] Richard P. Brent, *Fast multiple-precision evaluation of elementary functions*, Journal of the ACM **23** (1976), 242–251. ISSN 0004–5411. MR 52:16111. Available from `http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pub/pub034.html`.

[11] Richard P. Brent, *Multiple-precision zero-finding methods and the complexity of elementary function evaluation*, in [17] (1976), 151–176. MR 54:11843. Available from `http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pub/pub028.html`.

[12] Krzysztof Diks, Wojciech Ritter (editors), *Mathematical foundations of computer science 2002: 27th international symposium, MFCS 2002, Warsaw, Poland, 26–30.08.2002: proceedings*, Lecture Notes in Computer Science, 2420, Springer, Berlin, 2002.

[13] Mika Hirvensalo, Juhani Karhumäki, *Computing partial information out of intractable one—the first digit of* 2^n *at base* 3 *as an example*, in [12] (2002), 319–327.

[14] Wolfram Luther, Werner Otten, *Computation of standard interval functions in multiple-precision interval arithmetic*, Interval Computations (1994), 78–99. ISSN 0135–4868. MR 96g:65047.

[15] Donald J. Newman, *A simplified version of the fast algorithms of Brent and Salamin*, Mathematics of Computation **44** (1985), 207–210. MR 86e:65030.

[16] Eugene Salamin, *Computation of* π *using arithmetic-geometric mean*, Mathematics of Computation **30** (1976), 565–570. ISSN 0025–5718. MR 53:7928.

[17] Joseph F. Traub, *Analytic computational complexity*, Academic Press, New York, 1976. MR 52:15938.

[18] E. T. Whittaker, G. N. Watson, *A course of modern analysis*, Cambridge University Press, Cambridge, 1927. ISBN 0–521–58807–3. MR 97k:01072.

Department of Mathematics, Statistics, and Computer Science (M/C 249), The University of Illinois at Chicago, Chicago, IL 60607–7045

Email address: `djb@cr.yp.to`