An Algorithm for Computing Logarithms and Arctangents

By B. C. Carlson

Abstract. An iterative algorithm with fast convergence can be used to compute logarithms, inverse circular functions, or inverse hyperbolic functions according to the choice of initial conditions. Only rational operations and square roots are required. The method consists in adding an auxiliary recurrence relation to Borchardt's algorithm to speed the convergence.

1. Introduction. Logarithms, inverse circular functions, and inverse hyperbolic functions are computed in practice by a variety of methods, including infinite series, continued fractions, Chebyshev approximations, and rational approximations [4].

A method known as Borchardt's algorithm [2, Eq. (2.3)], [3] employs very simple recurrence relations containing a square root but is seldom used, because the error is reduced by a factor of only four per cycle. Thus, 10D accuracy typically requires fifteen to twenty square roots. A related algorithm, due to Thacher [6], has a factor of sixteen per cycle but loses some significant figures through cancellation.

In the present paper, we introduce an auxiliary recurrence relation to speed the convergence of Borchardt's algorithm so that only three or four square roots are required for 10D accuracy. (We assume here that the range of the independent variable has been reduced by familiar devices such as arcsin(1/x) = π/2 − arcsin x. More cycles are required outside the reduced range.) The method amounts to repeated application of a well-known technique [5, pp. 86-87] for improving convergence by extrapolation. The recurrence relations are the same for all the functions considered, there are no constants to be stored, and no serious cancellation occurs.

The precision is limited only by the number of cycles performed, and the rate of convergence gradually accelerates. Although still not as fast as some other methods because of the square roots, the algorithm might well be preferable when economy of storage space is important. It might be useful also in verifying the accuracy of faster algorithms. The method is being extended to computation of elliptic integrals, but only elementary functions are discussed in the present paper.

2. Statement of the Algorithm. The same recurrence relations will be used for computing a logarithm, an inverse circular function, or an inverse hyperbolic function, but the initial values and the final step will depend on which function is being computed. For definiteness, we state first the complete algorithms for computing natural logarithms and arctangents and later give the changes necessary for computing other functions.
Algorithm for logarithms. If $x > 0$, compute

(2.1) $a_0 = \frac{1}{2}(1 + x)$, \hspace{1cm} $g_0 = x^{1/2}$,

(2.2) $a_{n+1} = \frac{1}{2}(a_n + g_n)$, \hspace{1cm} $g_{n+1} = (a_{n+1}g_n)^{1/2}$, \hspace{1cm} $n = 0, 1, 2, \ldots$,

(2.3) $d(0, n) = a_n$, \hspace{1cm} $n = 0, 1, 2, \ldots$,

(2.4) $d(k, n) = \frac{d(k - 1, n) - 2^{-2k}d(k - 1, n - 1)}{1 - 2^{-2k}}$, \hspace{1cm} $k = 1, 2, \ldots, n$.

Then,

(2.5) $\log x = \frac{x - 1}{d(n, n)} (1 + \epsilon_n)$,

where the fractional error $\epsilon_n$ is given approximately by

(2.6) $\epsilon_n \approx 2^{-n^2-3n-1}(\log x)^{2n+2}$.

Algorithm for arctangents. If $x$ is real, let

(2.7) $a_0 = 1$, \hspace{1cm} $g_0 = (1 + x^2)^{1/2}$,

and compute $d(n, n)$ according to (2.2), (2.3), and (2.4). Then,

(2.8) $\arctan x = \frac{x}{d(n, n)} (1 + \epsilon_n)$,

where the fractional error is given approximately by

(2.9) $\epsilon_n \approx (-1)^{n+1}2^{-n^2-n+1}(\arctan x)^{2n+2}$.

To save computation time, the denominator of (2.4) may be deleted if the right-hand sides of (2.5) and (2.8) are multiplied by $(1-1/4)(1-1/16)(1-1/64) \cdots (1-2^{-2n})$. Then, (2.4) requires only a shift in binary arithmetic and a subtraction. To show that the subtraction never leads to loss of significant figures, we observe first that for $k = 1$ the numerator of (2.4) is $a_n - \frac{1}{2}a_{n-1}$. By (2.2), the first term is more than twice the second, and hence, serious cancellation cannot occur. As $k$ increases, the amount of cancellation decreases, for it is easy to show by induction that the ratio of the first to the second term in the numerator of (2.4) exceeds $2^k$.

In some circumstances, one might also save computation time by taking $g_n$ as a first approximation to $g_{n+1}$, when extracting the square root in (2.2) by the iterative method [4, pp. 90-91].

The functions $\arcsin x$ and $\arccos x$ can be computed directly by starting from the initial values $a_0$ and $g_0$ shown in Table I and ending the algorithm according to the last column of the table. The inverse hyperbolic functions, which are expressible in terms of logarithms, can be computed in the same way. We have simplified the initial values by using the fact that $d(n, n)$ is homogeneous of first degree in $a_0$ and $g_0$. In the case of the inverse circular functions, the approximate error is given by (2.9), with $\arctan x$ replaced by the function $f(x)$ in the first column of the table. For the inverse hyperbolic functions, it is given by (2.6), with $\log x$ replaced
by \(2f(x)\), the factor 2 coming from the left side of
\[
2 \text{arctanh } x = \log \left( \frac{1 + x}{1 - x} \right).
\]

Table I. Initial and final data for computing inverse circular and inverse hyperbolic functions. The first two rows correspond to (2.1), (2.5), (2.7), and (2.8). Although arccos \(x\) can be computed if \(-1 < x < 0\), loss of significant figures occurs near \(-1\).

<table>
<thead>
<tr>
<th>(f(x))</th>
<th>Range of (x)</th>
<th>(a_0)</th>
<th>(g_0)</th>
<th>(f(x)d(n,n))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\log x)</td>
<td>(x &gt; 0)</td>
<td>(\frac{1}{2}(1 + x))</td>
<td>(x^{1/2})</td>
<td>(x - 1)</td>
</tr>
<tr>
<td>(\arctan x)</td>
<td>(-\infty &lt; x &lt; \infty)</td>
<td>1</td>
<td>((1 + x^2)^{1/2})</td>
<td>(x)</td>
</tr>
<tr>
<td>(\arcsin x)</td>
<td>(-1 \leq x \leq 1)</td>
<td>((1 - x^2)^{1/2})</td>
<td>1</td>
<td>(x)</td>
</tr>
<tr>
<td>(\arccos x)</td>
<td>(0 \leq x \leq 1)</td>
<td>(1)</td>
<td>((1 - x^2)^{1/2})</td>
<td>(x)</td>
</tr>
<tr>
<td>(\arctanh x)</td>
<td>(-1 &lt; x &lt; 1)</td>
<td>(x)</td>
<td>((1 - x^2)^{1/2})</td>
<td>(x)</td>
</tr>
<tr>
<td>(\arcsinh x)</td>
<td>(-\infty &lt; x &lt; \infty)</td>
<td>((1 + x^2)^{1/2})</td>
<td>1</td>
<td>(x)</td>
</tr>
<tr>
<td>(\arccosh x)</td>
<td>(x \geq 1)</td>
<td>(1)</td>
<td>((x^2 - 1)^{1/2})</td>
<td>(x)</td>
</tr>
</tbody>
</table>

3. Numerical Examples. If \(y > 0\), we can find an integer \(m\) such that \(y = 2^m x\), where \(x\) lies in the reduced range \(2^{-1/2} < x \leq 2^{1/2}\). Since \(\log y = \log x + m \log 2\), it suffices to compute logarithms in the reduced range, and the worst case is \(x = 2^{1/2}\) because the rate of convergence improves as \(|\log x|\) decreases, according to (2.6). Therefore, we take as an example the computation of \(\log x\) for \(x = 1.41421 35623 731\) and list the values of \(g_n, a_n = d(0, n),\) and \(d(k, n)\), all rounded to 10D:

<table>
<thead>
<tr>
<th>(n)</th>
<th>(g_n)</th>
<th>(a_n)</th>
<th>(d(1, n))</th>
<th>(d(2, n))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.18920</td>
<td>71150</td>
<td>1.20710</td>
<td>67812</td>
</tr>
<tr>
<td>1</td>
<td>1.19367</td>
<td>36437</td>
<td>1.19815</td>
<td>69481</td>
</tr>
<tr>
<td>2</td>
<td>1.19479</td>
<td>39440</td>
<td>1.19591</td>
<td>52959</td>
</tr>
<tr>
<td>3</td>
<td>1.19535</td>
<td>46200</td>
<td>1.19516</td>
<td>77280</td>
</tr>
</tbody>
</table>

\[d(3, 3) = 1.19516 \times 77046\]

\[\log x = \frac{x - 1}{d(3, 3)} = \frac{0.41421 35624}{1.19516 77046} = 0.34657 35903\]

This result is the correct value of \(\frac{1}{2}\) log 2 to 10D. Because \(g_3\) is not needed to find \(d(3, 3)\), only three square roots were extracted. Although \(g_n, a_n,\) and \(d(n, n)\) approach the same limit, \(a_3\) and \(d(3, 3)\) agree to only 3D. Since Borchardt's algorithm would use \(a_3\) in place of \(d(3, 3)\), the auxiliary relation (2.4) produces a dramatic improvement in accuracy without additional square roots.

The fractional error \(e_n\) computed from (2.5) can be compared with the estimate (2.6). For \(n = 0\), the actual value is \(e_0 = 0.010\) and the estimate is 0.006. For \(n > 0\), the actual errors are \(e_1 = 5 \times 10^{-6}, e_2 = 9 \times 10^{-10}, e_3 = 4 \times 10^{-14}\), and the estimates agree to one significant figure. These errors show the gradual acceleration of convergence expected from the exponent \(-n^2\) in (2.6).
Outside the reduced range, the fractional errors are larger and the estimates are less accurate. For example, if \( x = 1024 \), some actual errors are \( \epsilon_0 = 2.5 \), \( \epsilon_2 = 0.02 \), \( \epsilon_4 = 2 \times 10^{-6} \), \( \epsilon_6 = 6 \times 10^{-13} \), and, for \( n > 0 \), the estimated errors are approximately three times the actual errors.

In computing arctangents, the relation \( \arctan(1/x) = \pi/2 - \arctan x \) makes it sufficient to consider the reduced range \(-1 \leq x \leq 1\). We take as an example the worst case, \( x = 1 \):

<table>
<thead>
<tr>
<th>( n )</th>
<th>( g_n )</th>
<th>( a_n )</th>
<th>( d(1, n) )</th>
<th>( d(2, n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.41421</td>
<td>35624</td>
<td>1.00000</td>
<td>00000</td>
</tr>
<tr>
<td>1</td>
<td>1.30656</td>
<td>29649</td>
<td>1.20710</td>
<td>67812</td>
</tr>
<tr>
<td>2</td>
<td>1.28145</td>
<td>77239</td>
<td>1.25683</td>
<td>48730</td>
</tr>
<tr>
<td>3</td>
<td>1.27528</td>
<td>71547</td>
<td>1.26914</td>
<td>62985</td>
</tr>
<tr>
<td>4</td>
<td>1.27221</td>
<td>67266</td>
<td>1.27324</td>
<td>02026</td>
</tr>
</tbody>
</table>

\[
\arctan x = \frac{x}{d(4, 4)} = \frac{1}{1.27323 \ 95447} = 0.78539 \ 81634
\]

This result is the correct value of \( \pi/4 \) to 10D. Only four square roots were extracted. Since \( a_4 \) agrees with \( d(4, 4) \) to only 2D, the auxiliary relation (2.4) has supplied eight additional decimal places.

The actual fractional errors calculated from (2.8) are \( \epsilon_0 = -0.2 \), \( \epsilon_1 = 0.002 \), \( \epsilon_2 = -8 \times 10^{-6} \), \( \epsilon_3 = 8 \times 10^{-8} \), \( \epsilon_4 = -2 \times 10^{-11} \), and the errors estimated by (2.9) agree to within one unit in the first significant figure. The gradual acceleration of convergence expected from (2.9) is again discernible.

4. Proof of the Algorithm. The solution of the recurrence relations (2.2) with initial conditions (2.1) is easily verified to be

\[
(4.1) \quad a_n = 2^{-n-1}(x - 1) \coth(2^{-n-1} \log x), \quad g_n = 2^{-n-1}(x - 1) \csch(2^{-n-1} \log x),
\]

from which it follows that

\[
(4.2) \quad \lim_{n \to \infty} a_n = \lim g_n = (x - 1)/(\log x)
\]
as \( n \to \infty \). The solution with initial conditions (2.7) is

\[
(4.3) \quad a_n = 2^{-n}x \cot(2^{-n} \arctan x), \quad g_n = 2^{-n}x \csc(2^{-n} \arctan x),
\]

which implies

\[
(4.4) \quad \lim a_n = \lim g_n = x/(\arctan x).
\]

Both solutions can be deduced directly from the usual form of Borchardt's algorithm [2].
Consider (4.1) and define for $x > 0$ and $t > 0$,

$$L = (x - 1)/(\log x), \quad \tau^{1/2} = \frac{1}{2} \log x, \quad f(t) = Lt^{1/2} \coth t^{1/2},$$

with $f(0) = L$. If $\theta = \frac{1}{4}$, then

$$a_n = f(\theta^n).$$

Assume for the moment that $|\tau| < \pi^2$, so that $\tau^{1/2} \coth \tau^{1/2}$ can be represented by a power series in $\tau$. Then,

$$a_0 = f(\tau) = c_0 + c_1 \tau + c_2 \tau^2 + c_3 \tau^3 + \cdots,$$

$$a_1 = f(\theta \tau) = c_0 + \theta c_1 \tau + \theta^2 c_2 \tau^2 + \theta^3 c_3 \tau^3 + \cdots,$$

$$a_2 = f(\theta^2 \tau) = c_0 + \theta^2 c_1 \tau + \theta^4 c_2 \tau^2 + \theta^6 c_3 \tau^3 + \cdots,$$

and so on, where $c_n$ depends on the Bernoulli number $B_{2n}$:

$$c_n = L2^{2n}B_{2n}/(2n)!. $$

The problem is to estimate $c_0 = f(0) = \lim a_n$, knowing the values of a finite number of the $a_n$. We can eliminate the terms linear in $\tau$ by forming

$$a_1 - \frac{\theta a_0}{1 - \theta} = c_0 - \theta c_2 \tau^2 - \theta(1 + \theta)c_3 \tau^3 - \cdots,$$

$$a_2 - \frac{\theta a_1}{1 - \theta} = c_0 - \theta^2 c_2 \tau^2 - \theta^4(1 + \theta)c_3 \tau^3 - \cdots,$$

and so on. The left side of the last equation gives an improved estimate of $c_0$. If the first equation of (4.9) is multiplied by $\theta^2$ and subtracted from the second, the terms in $\tau^2$ are removed, and the procedure can be continued to any order in $\tau$. The same result, an extrapolation from $a_0, a_1, \cdots, a_n$ to $\lim a_n$, is achieved by the following lemma, which does not require convergence of the power series.

**Lemma 1.** Let $f(t)$ and its first $n + 1$ derivatives be defined and continuous on the interval $0 \leq t \leq \tau$. Let $\theta$ be a fixed number, $0 < \theta < 1$, and suppose that $f(\theta^m \tau)$ is known for $m = 0, 1, \cdots, n$, and $f(0)$ is to be estimated. Define

$$d(0, m) = f(\theta^m \tau), \quad m = 0, 1, \cdots, n,$$

$$d(k, m) = \frac{d(k - 1, m) - \theta^k d(k - 1, m - 1)}{1 - \theta^k}, \quad k = 1, 2, \cdots, m.$$

Then,

$$f(0) = d(n, n) + \theta^n \frac{(n + 1)!}{(n + 1)} f^{(n+1)}(x_n),$$

where $0 < x_n < \tau$.

**Proof.** Let $t_m = \theta^n \tau$ and define divided differences

$$f[0, t_m] = \frac{f(0) - f(t_m)}{0 - t_m},$$

$$f[0, t_m, t_{m-1}, \cdots, t_{m-k}] = \frac{f[0, t_m, t_{m-1}, \cdots, t_{m-k+1}] - f[0, t_{m-1}, t_{m-2}, \cdots, t_{m-k}]}{t_m - t_{m-k}}.$$
The solution of the recurrence relations (4.11) with initial conditions (4.10) is easily verified to be
\[
(4.14) \quad d(k, m) = f(0) + (-1)^k t_m t_{m-1} \cdots t_{m-k} f[0, t_m, t_{m-1}, \ldots, t_{m-k}] .
\]
Solving for \( f(0) \), we find
\[
(4.15) \quad f(0) = d(k, m) + \theta^N (-\tau)^{k+1} f[0, t_m, t_{m-1}, \ldots, t_{m-k}]
\]
where \( 0 < x_{km} < \theta^{m-k} \tau \) and
\[
(4.16) \quad N = \sum_{r=m-k}^{m} r = \frac{1}{2}(2m - k)(k + 1).
\]
The case \( k = m = n \) is (4.12).

To apply Lemma 1 to the computation of logarithms, we choose \( f \) as in (4.5) and \( \theta = 1 \), so that \( d(0, m) = a_m = f(\theta^m \tau) \). Then, (4.12) becomes
\[
(4.17) \quad L = d(n, n) - L_{\epsilon_n},
\]
where
\[
(4.18) \quad L_{\epsilon_n} = (-1)^n 2^{-n(n+1)} \left( \frac{\log x}{2} \right)^{2n+2} \frac{f^{(n+1)}(x_n)}{(n + 1)!}.
\]
To estimate \( \epsilon_n \), we replace \( x_n \) by 0 and note that
\[
(4.19) \quad \frac{f^{(n+1)}(0)}{(n + 1)!} = c_{n+1} = \frac{L 2^{2n+2} B_{2n+2}}{(2n + 2)!} \approx \frac{(-1)^n 2^n L}{\pi^{2n+3}} ,
\]
where the last approximation follows from [1, Eq. (23.1.15)]. Thus,
\[
(4.20) \quad \epsilon_n \approx 2^{-n^2-n+1} \left( \frac{\log x}{2\pi} \right)^{2n+2} ,
\]
in agreement with (2.6), while (4.17) and (4.5) imply (2.5).

The analysis for arctangents is similar, except that (4.5) is replaced by
\[
(4.21) \quad L = x/(\arctan x), \quad \tau^{1/2} = \arctan x, \quad f(t) = Lt^{1/2} \cot t^{1/2} ,
\]
and the right side of (4.8) is multiplied by \((-1)^n\).

Acknowledgment. This work was performed in the Ames Laboratory of the U. S. Atomic Energy Commission. I am obliged to Elaine Clark for numerical calculations on the IBM 360/65 at the Iowa State University Computation Center. I thank also Professor R. J. Lambert for helpful conversations.

Departments of Physics and Mathematics
Iowa State University
Ames, Iowa 50010


