A New Method for Approximating Logarithms with k-th Order

William Y.X. Zou (wyxzou@gmail.ca)
Pierre Elliott Trudeau High School,
Markham, Ontario, L6C 2E6, Canada

April 30, 2014

Abstract. We provide an algorithm to approximate logarithms with k-th order convergence. In addition to fast convergence, an upper bound for the error is available within pre-defined levels.

Keywords: Function Approximation, Newton's method, k-th Order Convergence, Logarithm, Maclaurin series

MSC1991: 65D15

1. Introduction

In general, a smooth function can be approximated by its Maclaurin series. The Maclaurin series for the natural log, one of the building blocks for more sophisticated functions, is well-known

$$\ln(1+x) = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^n}{n}, \qquad -1 < x \leq 1$$

Compared with the series of other fundamental functions such as exponential and trigonometrical functions, the Maclaurin series of the natural logarithm converges slowly and is computationally costly, thus it should be avoided. One present solution is to apply Newton's method to its inverse function e^x since e^x converges more quickly. More specifically, solving $y = \ln x$ is equivalent to solving the equation $e^y - x = 0$, whose solution y can be approximated by

$$y_{n+1} = y_n - \frac{e^{y_n} - x}{e^{y_n}}. \tag{1}$$

To measure the convergence speed, recall that a sequence u_k converges to the number L with order k if

$$\lim_{n\to\infty} \frac{|u_{n+1} - L|}{|u_n - L|^k} = \mu,$$

for some $\mu > 0$. It has been shown that Newton's method can be modified to various cubic (order 3) convergence algorithms.[3] One of

them is as follows

$$y_{n+1} = y_n - 2\frac{e^{y_n} - x}{e^{y_n} + x}. \tag{2}$$

Another alternative for high precision is the formula

$$\ln x \approx \frac{\pi}{2M(1, \frac{4}{s})} - m \ln 2, \tag{3}$$

where M denotes the arithmetic-geometric mean of 1 and $4/s$, and

$$s = x2^m > 2^{p/2},$$

with m chosen so that p bits of precision is attained. See [4] for more details.

In this paper, we will provide a new algorithm with general order k. It converges quickly with desired accuracy and is easy to implement. The idea is to transform $\ln(1+x)$ to $t + \ln(1+w)$ such that w can be arbitrarily small, such that the error can be upbounded by a pre-defined level. We construct such t that satisfies the conditions, then calculate w. The process can then be repeated recursively. The benchmark test in Table I indicates that, with order 5 and with 5 recursions, the algorithm conveges to a solution that is accurate to 1175 decimal place in the worst scenario. See Section 2 for more details.

2. The Algorithm

In this section, we describe the algorithm in detail and present a benchmark testing result.

Without loss of generality, we shall focus on the $\log_2 x$ with $x > 0$. Logarithms with other bases can be calculated by the formula $log_a b = \frac{\log_2 b}{\log_2 a}$. We need the following lemma to limit the range of x.

LEMMA 2.1. *For any $x > 0$, one can find integer m and positive number w such that*

$$\log_2 x = m \pm \log_2(1+w), \qquad 0 \le w \le \frac{1}{2}. \tag{4}$$

Proof. One can write as $x = 2^n + b$ such that $0 \le b < 2^n$ with some integer n. Let $r = \frac{b}{2^n}$ and we obtain

$$\log_2 x = \log_2 2^n(1+r) = n + \log_2(1+r), \qquad 0 \le r < 1.$$

The proposition is proved if $r \le \frac{1}{2}$. Otherwise,$\frac{1}{2} < r < 1$, so

$$\begin{aligned} n + \log_2(1+r) &= (n+1) - \log_2 2 + \log_2(1+r) \\ &= n + 1 - \log_2(\frac{2}{1+r}) \\ &= n + 1 - \log_2(1 + \frac{1-r}{1+r}) \\ &= m - \log_2(1+r_1), \quad 0 < r_1 < \frac{1}{3}. \end{aligned}$$

which implies Lemma 2.1 with $m = n + 1$.

Based on the Lemma 2.1, we only need calculate $\log_2(1+a_0)$. For a k-th order convergence algorithm, we aim to find a_1 and t such that

$$\log_2(1+a_0) = t + \log_2(1+a_1) \tag{5}$$

and

$$0 \le a_1 \le a_0^k. \tag{6}$$

Using Equation (5), Inequality (6) is equivalent to

$$\frac{1+a_0}{1+a_0^k} \le 2^t \le 1 + a_0$$

or

$$\ln(1+a_0) - \ln(1+a_0^k) \le t \ln 2 \le \ln(1+a_0).$$

To find such t, observe that for $a_0 > 0$ since

$$a_0 - \cdots + \frac{a_0^{2n-1}}{2n-1} - \frac{a_0^{2n}}{2n} \le \ln(1+a_0) \le a_0 - \frac{1}{2}a_0^2 + \cdots + \frac{1}{2n-1}a_0^{2n-1}$$

and

$$\ln(1+a_0^k) \ge a_0^k - \frac{a_0^{2k}}{2},$$

then

$$\ln(1+a_0) - ln(1+a_0^k) \le f_n(a_0) \equiv a_0 - \cdots + \frac{1}{2n-1}a_0^{2n-1} - (a_0^k - \frac{a_0^{2k}}{2})$$

and

$$\ln(1+a_0) \ge g_n(a_0) \equiv a_0 - \cdots + \frac{a_0^{2n-1}}{2n-1} - \frac{a_0^{2n}}{2n}.$$

We shall choose n such that $f_n(a_0) \le g_n(a_0)$ and then define t such that

$$t \ln 2 = \frac{f_n(a_0) + g_n(a_0)}{2} = a_0 - \cdots + \frac{a_0^{2n-1}}{2n-1} + \frac{1}{2}(\frac{a_0^{2k}}{2} - \frac{a_0^{2n}}{2n} - a_0^k).$$

It is easy to see that inequality $f_n(a_0) \leq g_n(a_0)$ is equivalent to

$$-\frac{a_0^{2n}}{2n} \geq -a_0^k + \frac{a_0^{2k}}{2}$$

or

$$a_0^k \geq \frac{a_0^{2n}}{2n} + \frac{a_0^{2k}}{2}.$$

If we choose $n = k$, the above inequality holds since

$$1 \geq \frac{a_0^k}{2k} + \frac{a_0^k}{2}$$

is true for any positive integer k if $0 \leq a_0 \leq \frac{1}{2}$. Once t is solved, we can use the formula

$$a_1 = \frac{1 + a_0}{2^t} - 1 \tag{7}$$

to find a_1 and the process can be repeated on $\log_2(1 + a_1)$. If we repeat this process n times, we have

$$t = t_1, t_2, \ldots, t_n.$$

Define

$$T = \sum_{i=1}^{n} t_i \qquad W = \frac{1 + a_0}{2^T} - 1. \tag{8}$$

We have

$$\log_2(1 + a_0) = T + \log_2(1 + W).$$

One can estimate $\log_2(1+W) = ln(1+W)/\ln 2$ by using the Maclaurin series.

$$\ln(1+W) \approx W - \frac{1}{2}W^2 + \ldots + (-1)^q \frac{1}{q} W^q + E_q, \quad |E_q| < \frac{1}{q+1} W^{q+1} \tag{9}$$

We summarize our analysis in the following Theorem.

THEOREM 2.2. *The algorithm discussed above with n recursions converges with k-th order with error bound* $a_0^{(q+1)k^n}$ *which is always less than* $(\frac{1}{2})^{(q+1)k^n}$.

In the recursive algorithm, we assume that one can precisely calculate 2^t in equations (7) and (8). If 2^t cannot be pre-computed precisely, identity (8) can be used to avoid compounding errors in previous recursions to find t_i, making it possible to obtain an upper bound on the absolute error.

MATLAB was used to conduct a benchmark test for $\log_2 1.5$, which is the worst case scenario for convergence ($a_0 = \frac{1}{2}$). The test results indeed suggest a k-th order convergence speed.

For all tests, we use the Multiprecision Computing Toolbox for MATLAB, which enables our precision to be as high as 2000 decimal places. The error is the difference between the algorithm output and the system function output.

Table I. Approximation Error of $\log_2 1.5$ Using the New Method

recursive steps	k=2	$k = 3$	$k = 4$	$k = 5$
1	1.3E-01	6.3E-02	3.2E-02	1.6E-02
2	8.4E-03	1.3E-04	5.0E-07	4.8E-10
3	3.5E-05	1.1E-12	3.2E-26	1.3E-47
4	6.1E-10	6.9E-37	5.0E-103	1.8E-235
5	1.9E-19	1.7E-109	3.1E-410	9.5E-1175
6	1.8E-38	2.4E-327	4.7E-1639	0.0E+00
7	1.6 E-76	6.6E-981	0.0E+00	0.0E+00
8	1.2 E-152	0.0E+00	0.0E+00	0.0E+00
9	7.3E-305	0.0E+00	0.0E+00	0.0E+00
10	2.6E-609	0.0E+00	0.0E+00	0.0E+00

3. Performance comparisons

In this section, we compare the performance of the three previous methods we mentioned in Section 1 to the new method. They are

1. Standard Newton's method by Equation 1.
2. Modified Newton's method by Equation 2.
3. The method using arithmetic-geometric mean by Equation 3. The test results will be labelled as "The approximation errors of $\log_2 1.5$ by Arithmetic-Geometric Method".

The algorithm using the arithmetic-geometric mean is a method that demonstrates good accuracy in one step. We report the test results for $log_2(1.5)$ with different m values and different levels of convergence for arithmetic-geometric mean. The results are shown in Table II. On

M ATLAB, the best accuracy is achieved at $m = 100$ with 29 decimal place. In summary, the high-speed algorithm is useful for one-step calculation, but it limited to around 30 decimal places.

It is well known that standard Newton's method and modified Newton's method have quadratic and cubic convergence. Our tests confirm that the new method converges at a higher order than the standard Newton's and modified Newton's method.

4. Time Complexity

From the error bound estimation in Theorem 2.2, it is easy to see that we need at most $\log_k(\frac{m \log_2 10}{q+1})$ recursions if m-digit precision is required, where k is the order of convergence and q is the number of terms in the Taylor expansion of Equation (9).

Each recursion needs $c_1 k$ arithmetic operations if we ignore the cost for 2^t, and the Taylor expansion of Equation (9) needs $c_2 q + \log_k \frac{m \log_2 10}{q+1}$ arithmetic operations, where c_1 and c_2 are absolute constants. The number of steps required by the new algorithm is

$$c_1 k \log_k(\frac{m \log_2 10}{q+1}) + c_2 q.$$

Therefore the time complexity for the algorithm is $O(k \log_k m)$, which outperforms the time complexity for Newton's method when k is large. The time complexity of Newton's method is $O(\log(m)Q(m))$, where $Q(m)$ is the cost of calculating $\frac{f(x)}{f'(x)}$.

Note that the time complexity of Newton's method is only clear if a good initial approximation is known. The new algorithm is not bound by any constraints.

5. Conclusion

We provide an algorithm with a higher order convergence than Newton's method. The algorithm converges at order k and has a controlled error bound. We expect that the advantages of the new algorithm to be more significant when extreme accuracy is required.

6. Acknowledgements

This paper was presented at the Canada Wide Science Fair. The author received valuable suggestions from the judging committee. The author would also like to thank his family for their ongoing support, the time they took to read and edit this paper, and for their constructive feedback.

Table II. The approximation errors of $\log_2 1.5$ by Arithmetic-Geometric Method(refer to Equation 3)

			m-value		
$ErrorBoundofAGM$	8	10	15	25	30
$1.00E-10$	-3.8719	-4.9688	-7.7897	-13.578	-16.5066
$1.00E-30$	-3.8719	-4.9688	-7.7897	-13.578	-16.5066
$1.00E-50$	-3.8719	-4.9688	-7.7897	-13.578	-16.5066
$1.00E-100$	-3.8719	-4.9688	-7.7897	-13.578	-16.5066
			m-value		
$ErrorBoundofAGM$	50	100	200	300	500
$1.00E-10$	-28.3908	-29.0317	-28.8238	-18.1456	-22.1571
$1.00E-30$	-28.3908	-29.0317	-28.8238	-18.1456	-22.1571
$1.00E-50$	-28.3908	-29.0317	-28.8238	-18.1456	-22.1571
$1.00E-100$	-28.3908	-29.0317	-28.8238	-18.1456	-22.1571

References

1. Borwein JM, Borwein PB.: Arithmetic-Geometric Mean and Fast Computations of Elementary Functions. SIAM Review 26(3)(1984):351–366;
2. Homeier H.: A modified Newton method for rootfinding with cubic convergence. Journal of Computational and Applied Mathematics 157(1)(2003): 227–230;
3. Homeier H.: On Newton-type methods with cubic convergence. Journal of Computational and Applied Mathematics 176(2)(2005): 425–432;
4. Saskaki T, Kanada Y.: Practically Fast Multiple-Precision Evaluation of LOG(X). Journal of information processing 5(4)(1982): 247–250;
5. Schraudolph N.: A Fast, Compact Approximation of the Exponential Function. MIT Press Journals 1(4)(1999): 853-862;