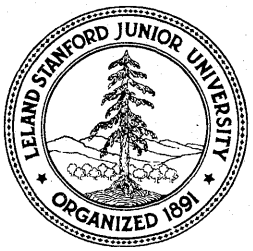
FAST MULTIPLE-PRECISION EVALUATION OF ELEMENTARY FUNCTIONS

by

R. P. Brent

STAN-CS-75-515
AUGUST 1975

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY



Fast Multiple-precision Evaluation of Elementary Functions

Richard P. Brent

Computer Science Department
Stanford University
Stanford, California 94305

and

Computer Centre
Australian National University
Box 4, Canberra
ACT 2600, Australia

Abstract

Let $f(x)$ be one of the usual elementary functions (exp, log, artan, sin, cosh, etc.), and let $M(n)$ be the number of single-precision operations required to multiply n-bit integers. We show that $f(x)$ can be evaluated, with relative error $O(2^{-n})$, in $O(M(n)\log(n))$ operations as $n \to \infty$, for any floating-point number x (with an n-bit fraction) in a suitable finite interval. From the Schönhage-Strassen bound on $M(n)$, it follows that an n-bit approximation to $f(x)$ may be evaluated in $O(n \log^2(n) \log\log(n))$ operations. Special cases include the evaluation of constants such as π, e, and e^{π}. The algorithms depend on the theory of elliptic integrals, using the arithmetic-geometric mean iteration and ascending Landen transformations.

Keywords and Phrases: analytic complexity, arithmetic-geometric mean, computational complexity, elementary function, elliptic integral, evaluation of π, exponential, Landen transformation, logarithm, multiple-precision arithmetic, trigonometric function.

CR Categories: 5.12, 5.15, 5.25

This research was supported in part by National Science Foundation grant DCR71-01996 A04 and by National Science Foundation grant DCR72-03712. Reproduction in whole or in part is permitted for any purpose of the United States Government.

1. Introduction.

We consider the number of operations required to evaluate the elementary functions $\exp(x)$, $\log(x)$ [*], $\arctan(x)$, $\sin(x)$, etc., with relative error $O(2^{-n})$, for x in some interval $[a,b]$, and large n . Here, $[a,b]$ is a fixed, nontrivial interval on which the relevant elementary function is defined. The results hold for computations performed on a multitape Turing machine, but to simplify the exposition we assume that a standard serial computer with a random-access memory is used.

Let $M(n)$ be the number of operations required to multiply two integers in the range $[0,2^n)$. We assume the number representation is such that addition can be performed in $O(M(n))$ operations, and that $M(n)$ satisfies the weak regularity condition

$$M(\alpha n) \leq \beta M(n) \quad , \tag{1.1}$$

for some α and β in $(0,1)$, and all sufficiently large n . Similar, but stronger, conditions are usually assumed either explicitly [6] or implicitly [9]. Our assumptions are certainly valid if the Schönhage-Strassen method [9,11] is used to multiply n-bit integers (in the usual binary representation) in $O(n \log(n) \log\log(n))$ operations.

The elementary function evaluations may be performed entirely in fixed point, using integer arithmetic and some implicit scaling scheme. However, it is more convenient to assume that floating-point computation is used. For example, a sign and magnitude representation could be used, with a fixed-length binary exponent and an n-bit binary fraction. Our results are independent of the particular floating-point number system are used, so long as the following conditions are satisfied.

1. Real numbers which are not too large or small can be approximated by floating-point numbers, with a relative error $O(2^{-n})$.
2. Floating-point addition and multiplication can be performed in $O(M(n))$ operations, with a relative error $O(2^{-n})$ in the result.

[*] $\log(x)$ denotes the natural logarithm.

3. The precision n is variable, and a floating-point number with precision n may be approximated, with relative error $O(2^{-m})$ and in $O(M(n))$ operations, by a floating-point number with precision m, for any positive $m < n$.

Throughout this paper, a floating-point number means a number in some representation satisfying conditions 1 to 3 above, not a single-precision number. We say that an operation is performed with precision n if the result is obtained with a relative error $O(2^{-n})$. It is assumed that the operands and result are approximated by floating-point numbers.

The main result of this paper, established in Sections 6 and 7, is that all the usual elementary functions may be evaluated, with precision n, in $O(M(n)\log(n))$ operations. Note that $O(M(n)n)$ operations are required if the Taylor series for $\log(1+x)$ is summed in the obvious way. Our result improves the bound $O(M(n)\log^2(n))$ given in [3], although the algorithms described there may be faster for small n.

Preliminary results are given in Section 2 to 5. In Section 2 we give, for completeness, the known result that division and extraction of square roots to precision n require $O(M(n))$ operations. Section 3 deals briefly with methods for approximating simple zeros of nonlinear equations to precision n, and some results from the theory of elliptic integrals are summarized in Section 4. Since our algorithms for elementary functions require a knowledge of π to precision n, we show, in Section 5, how this may be obtained in $O(M(n)\log(n))$ operations. An amusing consequence of the results of Section 6 is that e^{π} may also be evaluated, to precision n, in $O(M(n)\log(n))$ operations.

From Theorem 5.1 of [3], at least $O(M(n))$ operations are required to evaluate $\exp(x)$ or $\sin(x)$ to precision n. It is plausible to conjecture that $O(M(n)\log(n))$ operations are necessary.

Most of this paper is concerned with order of magnitude results, and multiplicative constants are ignored. In Section 8, though, we give upper bounds on the constants. From these bounds it is possible to estimate how large n needs to be before our algorithms are faster than the conventional ones.

After this report was written, Bill Gosper drew my attention to the paper of Salamin [10], where an algorithm very similar to our algorithm for evaluating π is described. A fast algorithm for evaluating $\log(x)$ was also found (independently) by Salamin (see [2]).

2. Reciprocals and Square Roots.

In this section we show that reciprocals and square roots of floating-point numbers may be evaluated, to precision n, in $O(M(n))$ operations. To simplify the statement of the following lemma, we assume that $M(x) = 0$ for all $x < 1$.

Lemma 2.1. If $\gamma \in (0,1)$, then $\sum_{j=0}^{\infty} M(\gamma^j n) = O(M(n))$ as $n \to \infty$.

Proof. If α and β are as in (1.1), there exists k such that $\gamma^k \le \alpha$. Thus, $\sum_{j=0}^{\infty} M(\gamma^j n) \le k \sum_{j=0}^{\infty} M(\alpha^j n) \le kM(n)/(1-\beta) + O(1)$, by repeated application of (1.1). Since $M(n) \to \infty$ as $n \to \infty$, the result follows. □

In the following lemma, we assume that $1/c$ is in the allowable range for floating-point numbers. Similar assumptions are implicit below.

Lemma 2.2. If c is a nonzero floating-point number, then $1/c$ can be evaluated, to precision n, in $O(M(n))$ operations.

Proof. The Newton iteration

$$x_{i+1} = x_i(2 - cx_i) \tag{2.1}$$

converges to $1/c$ with order 2. In fact, if $x_i = (1 - \varepsilon_i)/c$, substitution in (2.1) gives $\varepsilon_{i+1} = \varepsilon_i^2$. Thus, assuming $|\varepsilon_0| < 1/2$,

we have $|\varepsilon_i| < 2^{-2^i}$ for all $i \geq 0$, and x_k is a sufficiently good approximation to $1/c$ if $k \geq \log_2 n$. This assumes that (2.1) is satisfied exactly, but it is easy to show that it is sufficient to use precision n at the last iteration ($i = k-1$), precision slightly greater than $n/2$ for $i = k-2$, etc. (Details, and more efficient methods, are given in [3,4].) Thus, the result follows from Lemma 2.1. Since $x/y = x(1/y)$, it is clear that floating-point division may also be done in $O(M(n))$ operations. □

Lemma 2.3. If $c \geq 0$ is a floating-point number, then $c^{1/2}$ can be evaluated, to precision n, in $O(M(n))$ operations.

Proof. If $c = 0$ then $c^{1/2} = 0$. If $c \neq 0$, the proof is similar to that of Lemma 2.2, using the Newton iteration $x_{i+1} = (x_i + c/x_i)/2$. □

Lemma 2.4. For any fixed $k > 0$, $M(kn) = O(M(n))$ as $n \to \infty$.

Proof. Since we can add integers less than 2^n in $O(M(n))$ operations, we can add integers less than 2^{kn} in $O(kM(n)) = O(M(n))$ operations. The multiplication of integers less than 2^{kn} can be split into $O(k^2)$ multiplications of integers less than 2^n, and $O(k^2)$ additions, so can be done in $O(k^2M(n)) = O(M(n))$ operations. □

3. Solution of Nonlinear Equations.

In Section 6 we need to solve nonlinear equations to precision n. The following lemma is sufficient for this application. Stronger results are given in [3,4].

Lemma 3.1. If the equation $f(x) = c$ has a simple root $\zeta \neq 0$, f' is Lipschitz continuous near ζ, and we can evaluate $f(x)$ to precision n in $O(M(n)\phi(n))$ operations, where $\phi(n)$ is a positive, monotonic increasing function, for x near ζ, then ζ can be evaluated to precision n in $O(M(n)\phi(n))$ operations.

Proof. Consider the discrete Newton iteration

$$x_{i+1} = x_i - h_i(f(x_i) - c) / (f(x_i + h_i) - f(x_i)) \quad . \tag{3.1}$$

If $h_i = 2^{-n/2}$, $x_i - \zeta = O(2^{-n/2})$, and the right side of (3.1) is evaluated with precision n , then a standard analysis shows that $x_{i+1} - \zeta = O(2^{-n})$. Since a sufficiently good starting approximation x_0 may be found in $O(1)$ operations, the result follows in the same way as in the proof of Lemma 2.2, using the fact that Lemma 2.1 holds with $M(n)$ replaced by $M(n)\phi(n)$. The assumption $\zeta \neq 0$ is only necessary because we want to obtain ζ with a relative (not absolute) error $O(2^{-n})$.

Other methods, e.g. the secant method, may also be used if the precision is increased appropriately at each iteration. □

4. Results on Elliptic Integrals.

In this section we summarize some classical results from elliptic integral theory. Most of the results may be found in [1], so proofs are omitted. Elliptic integrals of the first and second kinds are defined by

$$F(\varphi,\alpha) = \int_0^{\varphi} (1 - \sin^2 \alpha \sin^2 \theta)^{-1/2} \, d\theta \tag{4.1}$$

and

$$E(\varphi,\alpha) = \int_0^{\varphi} (1 - \sin^2 \alpha \sin^2 \theta)^{1/2} \, d\theta \quad , \tag{4.2}$$

respectively. For our purposes we may assume that α and φ are in $[0 , \pi/2]$. The complete elliptic integrals, $F(\pi/2 , \alpha)$ and $E(\pi/2 , \alpha)$, are simply written as $F(\alpha)$ and $E(\alpha)$, respectively.

Legendre's Relation.

We need the identity

$$E(\alpha)F(\pi/2-\alpha) + E(\pi/2-\alpha)F(\alpha) - F(\alpha)F(\pi/2-\alpha) = \pi/2 \quad , \tag{4.3}$$

and, in particular, the special case

$$2E(\pi/4)F(\pi/4) - (F(\pi/4))^2 = \pi/2 \quad . \tag{4.4}$$

Small Angle Approximation.

From (4.1) it is clear that

$$F(\varphi,\alpha) = \varphi + O(\alpha^2) \tag{4.5}$$

as $\alpha \to 0$.

Large Angle Approximation.

From (4.1),

$$F(\varphi,\alpha) = F(\varphi\ ,\ \pi/2) + O(\pi/2 - \alpha)^2 \quad , \tag{4.6}$$

uniformly for $0 \le \varphi \le \varphi_0 < \pi/2$, as $\alpha \to \pi/2$. Also, we note that

$$F(\varphi\ ,\ \pi/2) = \log \tan(\pi/4 + \varphi/2) \quad . \tag{4.7}$$

Ascending Landen Transformation.

If $0 < \alpha_i < \alpha_{i+1} < \pi/2$, $0 < \varphi_{i+1} < \varphi_i \le \pi/2$,

$$\sin \alpha_i = \tan^2(\alpha_{i+1}/2) \quad , \tag{4.8}$$

and

$$\sin(2\varphi_{i+1} - \varphi_i) = \sin \alpha_i \sin \varphi_i \quad , \tag{4.9}$$

then

$$F(\varphi_{i+1}\ ,\ \alpha_{i+1}) = [(1 + \sin \alpha_i)/2]F(\varphi_i\ ,\ \alpha_i) \quad . \tag{4.10}$$

If $s_i = \sin \alpha_i$ and $v_i = \tan(\varphi_i/2)$, then (4.8) gives

$$s_{i+1} = 2s_i^{1/2} / (1 + s_i) \quad , \tag{4.11}$$

and (4.9) gives

$$v_{i+1} = w_3 / (1 + (1 + w_3^2)^{1/2}) \quad , \tag{4.12}$$

where

$$w_3 = \tan \varphi_{i+1} = (v_i + w_2)/(1 - v_i w_2) \quad , \tag{4.13}$$

$$w_2 = \tan(\varphi_{i+1} - \varphi_i/2) = w_1 / (1 + (1 - w_1^2)^{1/2}) \quad , \tag{4.14}$$

and

$$w_1 = \sin(2\varphi_{i+1} - \varphi_i) = 2s_i v_i / (1 + v_i^2) \quad . \tag{4.15}$$

Arithmetic-geometric Mean Iteration.

From the ascending Landen transformation it is possible to derive the arithmetic-geometric mean iteration of Gauss: if $a_0 = 1$, $b_0 = \cos \alpha > 0$,

$$a_{i+1} = (a_i + b_i)/2 \quad , \tag{4.16}$$

and

$$b_{i+1} = (a_i b_i)^{1/2} \quad , \tag{4.17}$$

then

$$\lim_{i \to \infty} a_i = \pi / [2F(\alpha)] \quad . \tag{4.18}$$

Also, if $c_0 = \sin \alpha$ and

$$c_{i+1} = a_i - a_{i+1} \quad , \tag{4.19}$$

then

$$E(\alpha) / F(\alpha) = 1 - \sum_{i=0}^{\infty} 2^{i-1} c_i^2 \quad . \tag{4.20}$$

An Infinite Product.

Let s_i, a_i and b_i be as above, with $\alpha = \pi/2 - \alpha_0$, so $s_0 = b_0/a_0$. From (4.11), (4.16) and (4.17), it follows that $s_i = b_i/a_i$ for all $i \geq 0$. Thus, $(1+s_i)/2 = a_{i+1}/a_i$, and

$$\prod_{i=0}^{\infty} [(1+s_i)/2] = \lim_{i\to\infty} a_i = \pi/[2F(\pi/2-\alpha_0)] \qquad (4.21)$$

follows from (4.18). (Another connection between (4.11) and the arithmetic-geometric mean iteration is evident if $s_0 = (1-b_0^2/a_0^2)^{1/2}$. Assuming (4.11) holds for $i < 0$, it follows that $s_{-i} = (1-b_i^2/a_i^2)^{1/2}$ for all $i \geq 0$. This may be used to deduce (4.18) from (4.10).)

5. Evaluation of π.

Let $a_0 = 1$, $b_0 = c_0 = 2^{-1/2}$, $A = \lim_{i\to\infty} a_i$, and $T = \lim_{i\to\infty} t_i$, where a_i, b_i and c_i are defined by (4.16), (4.17) and (4.19) for $i \geq 1$, and $t_i = 1/2 - \sum_{j=0}^{i} 2^{j-1} c_j^2$. From (4.4), (4.18) and (4.20), we have

$$\pi = A^2/T \quad . \qquad (5.1)$$

Since $a_i > b_0 > 0$ for all $i \geq 0$, and $c_{i+1} = a_i - a_{i+1} = a_{i+1} - b_i$, (4.17) gives $b_{i+1} = [(a_{i+1}+c_{i+1})(a_{i+1}-c_{i+1})]^{1/2} = a_{i+1} - O(c_{i+1}^2)$, so $c_{i+2} = O(c_{i+1}^2)$. Thus, the process converges with order at least 2, and $\log_2 n + O(1)$ iterations suffice to give an error $O(2^{-n})$ in the estimate of (5.1). A more detailed analysis shows

that $a_{i+1}^2/t_i < \pi < a_i^2/t_i$ for all $i \geq 0$, and also $a_i^2/t_i - \pi \sim 8\pi \exp(-2^i \pi)$ and $\pi - a_i^2/t_{i-1} \sim 2^{i+3} \pi^2 \exp(-2^i \pi)$ as $i \to \infty$. The speed of convergence is illustrated in Table 1.

Table 1: Convergence of approximations to π.

i	$\pi - a_{i+1}^2/t_i$	$a_i^2/t_i - \pi$
0	2.3'-1	8.6'-1
1	1.0'-3	4.6'-2
2	7.4'-9	8.8'-5
3	1.8'-19	3.1'-10
4	5.5'-41	3.7'-21

From the discussion above, it is clear that the following algorithm, given in pseudo-Algol, evaluates π to precision n.

Algorithm for π.

```
A ← 1; B ← 2^(-1/2); T ← 1/4; X ← 1;
while A-B > 2^(-n) do
    begin Y ← A; A ← (A+B)/2; B ← (BY)^(1/2);
          T ← T - X(A-Y)^2; X ← 2X
    end;
return A^2/T  [or, better, (A+B)^2/(4T)].
```

Since $\log_2 n + O(1)$ iterations are needed, it is necessary to work with precision $n + O(\log \log(n))$, even though the algorithm is numerically stable in the conventional sense. From Lemmas 2.2 to 2.4,

each iteration requires $O(M(n))$ operations, so π may be evaluated to precision n in $O(M(n)\log(n))$ operations. This is asymptotically faster than the usual $O(n^2)$ methods [8,13] if a fast multiplication algorithm is used. A high-precision computation of π by a similar algorithm is described in [5]. Note that, because the arithmetic-geometric mean iteration is not self-correcting, we can not obtain a bound $O(M(n))$ in the same way as for the evaluation of reciprocals and square roots by Newton's method.

6. Evaluation of exp(x) and log(x) .

Suppose $\delta > 0$ fixed, and $m \in [\delta, 1-\delta]$. If $\sin \alpha_0 = m^{1/2}$, we may evaluate $F(\alpha_0)$ to precision n in $O(M(n)\log(n))$ operations, using (4.18) and the arithmetic-geometric mean iteration, as for the special case $F(\pi/4)$ described in Section 5. (When using (4.18) we need π , which may be evaluated as described above.) Applying the ascending Landen transformation (4.8-10) with $i = 0,1,\ldots,k-1$ and $\varphi_0 = \pi/2$ gives

$$F(\varphi_k,\alpha_k) = \left\{ \prod_{i=0}^{k-1} [(1 + \sin \alpha_i)/2] \right\} F(\alpha_0) \quad . \tag{6.1}$$

Since $s_0 = \sin \alpha_0 = m^{1/2} \geq \delta^{1/2} > 0$, it follows from (4.11) that $s_i \to 1$ as $i \to \infty$. In fact, if $s_i = 1-\varepsilon_i$, then $\varepsilon_{i+1} = 1-s_{i+1} = 1 - 2(1-\varepsilon_i)^{1/2} / (2-\varepsilon_i) = \varepsilon_i^2/8 + \varrho(\varepsilon_i^3)$, so $s_i \to 1$ with order 2 . Thus, after $k \sim \log_2 n$ iterations we have $\varepsilon_k = O(2^{-n})$, so $\pi/2 - \alpha_k = O(2^{-n/2})$ and, from (4.6) and (4.7),

$$F(\varphi_k,\alpha_k) = \log \tan(\pi/4 + \varphi_k/2) + O(2^{-n}) \quad . \tag{6.2}$$

Assuming $k > 0$, the error is uniformly $O(2^{-n})$ for all $m \in [\delta , 1-\delta]$, since $\varphi_k \le \varphi_1 < \pi/2$.

Define the functions

$$U(m) = \left\{ \prod_{i=0}^{\infty} [(1 + \sin \alpha_i)/2] \right\} F(\alpha_0) \tag{6.3}$$

and

$$T(m) = \tan(\pi/4 + \varphi_\infty/2) \quad , \tag{6.4}$$

where $\varphi_\infty = \lim_{i \to \infty} \varphi_i$. Since $s_i \to 1$ with order 2 , the infinite product in (6.3) is convergent, and $U(m)$ is analytic for all $m \in (0,1)$. Taking the limit in (6.1) and (6.2) as n (and hence k) tends to ∞ , we have the fundamental identity

$$U(m) = \log T(m) \quad . \tag{6.5}$$

Using (4.11) to (4.15), we can evaluate

$U(m) = \left\{ \prod_{i=0}^{k-1} [(1+s_i) / 2] \right\} F(\alpha_0) + O(2^{-n})$ and

$T(m) = (1+v_k)/(1-v_k) + O(2^{-n})$, to precision n , in $O(M(n) \log(n))$ operations. The algorithms are given below in pseudo-Algol.

Algorithm for U(m) .

```
A ← 1; B ← (1-m)^1/2;
while A-B > 2^-n/2 do
    begin C ← (A+B)/2; B ← (AB)^1/2; A ← C end;
A ← π/(A+B); S ← m^1/2;
while 1-S > 2^-n/2 do
    begin A ← A(1+S)/2; S ← 2S^1/2/(1+S) end;
return A(1+S)/2.
```

Algorithm for T(m) .

```
V ← 1; S ← m^1/2;
while 1-S > 2^-n do
    begin W ← 2SV/(1+V^2);
        W ← W/(1+ (1-W^2)^1/2);
        W ← (V+W)/(1-VW);
        V ← V/(1+ (1+W^2)^1/2);
        S ← 2S^1/2/(1+S)
    end;
return (1+V)/(1-V).
```

Properties of U(m) and T(m) .

From (4.21) and (6.3),

$$U(m) = (\pi/2)F(\alpha_0)/F(\pi/2 - \alpha_0) \quad , \tag{6.6}$$

where $\sin \alpha_0 = m^{1/2}$ as before. Both $F(\alpha_0)$ and $F(\pi/2 - \alpha_0)$ may be evaluated by the arithmetic-geometric mean iteration, which leads

to a slightly more efficient algorithm for $U(m)$ than the one above, because the division by $(1+S)$ in the final "while" loop is avoided. From (6.5) and (6.6), we have the special cases $U(1/2) = \pi/2$ and $T(1/2) = e^{\pi/2}$. Also, (6.6) gives

$$U(m)U(1-m) = \pi^2/4 \quad , \tag{6.7}$$

for all $m \in (0,1)$.

Although we shall avoid using values of m near 0 or 1, it is interesting to obtain asymptotic expressions for $U(m)$ and $T(m)$ as $m \to 0$ or 1. From the algorithm for $T(m)$,

$$T(1-\varepsilon) = 4\varepsilon^{-1/2} - \varepsilon^{1/2} + O(\varepsilon^{3/2})$$

as $\varepsilon \to 0$. Thus, from (6.5),

$$U(1-\varepsilon) = L(\varepsilon) - \varepsilon/4 + O(\varepsilon^2) \quad ,$$

where $L(\varepsilon) = \log(4/\varepsilon^{1/2})$. Using (6.7), this gives

$$U(\varepsilon) = \pi^2/[4L(\varepsilon)] + O(\varepsilon/L^2) \quad ,$$

and hence

$$T(\varepsilon) = \exp(\pi^2/[4L(\varepsilon)]) + O(\varepsilon/L^2) \quad .$$

Some values of $U(m)$ and $T(m)$ are given in Table 2.

Table 2: The functions U(m) and T(m)

m	U(m)	T(m)
0.01	0.6693	1.9529
0.05	0.8593	2.3615
0.10	0.9824	2.6710
0.20	1.1549	3.1738
0.30	1.2972	3.6591
0.40	1.4322	4.1878
0.50	1.5708	4.8105
0.60	1.7228	5.6004
0.70	1.9021	6.6999
0.80	2.1364	8.4688
0.90	2.5115	12.3235
0.95	2.8714	17.6617
0.99	3.6864	39.8997

Evaluation of $\exp(x)$.

To evaluate $\exp(x)$ to precision n, we first use identities such as $\exp(2x) = (\exp(x))^2$ and $\exp(-x) = 1/\exp(x)$ to reduce the argument to a suitable domain, say $1 \leq x \leq 2$ (see below). We then solve the nonlinear equation

$$U(m) = x \ , \tag{6.8}$$

obtaining m to precision n, by a method such as the one described in Section 3. From Lemma 3.1, with $\phi(n) = \log(n)$, this may be done in $O(M(n)\log(n))$ operations. Finally, we evaluate $T(m)$ to precision n, again using $O(M(n)\log(n))$ operations. From (6.5) and (6.8), $T(m) = \exp(x)$, so we have computed $\exp(x)$ to precision n. Any preliminary transformations may now be undone.

Evaluation of $\log(x)$.

Since we can evaluate $\exp(x)$ to precision n in $O(M(n)\log(n))$ operations, Lemma 3.1 shows that we can also evalute $\log(x)$ in $O(M(n)\log(n))$ operations, by solving the equation $\exp(y) = x$ to the desired accuracy. A more direct method is to solve $T(m) = x$ (after suitable domain reduction), and then evaluate $U(m)$.

Further Details.

If $x \in [1,2]$ then the solution m of (6.8) lies in $(0.10, 0.75)$, and it may be verified that the secant method, applied to (6.8), converges if the starting approximations are $m_0 = 0.2$ and $m_1 = 0.7$. If desired, the discrete Newton method or some other locally convergent method may be used after a few iterations of the secant method have given a good approximation to m .

Similarly, if $x \in [3,9]$, the solution of $T(m) = x$ lies in $(0.16, 0.83)$, , and the secant method converges if $m_0 = 0.2$ and $m_1 = 0.8$.

If $x = 1+\varepsilon$ where ε is small, and for domain reduction the relation

$$\log(x) = \log(\lambda x) - \log(\lambda) \qquad (6.9)$$

is used, for some $\lambda \in (3,9)$, then $\log(\lambda x)$ and $\log(\lambda)$ may be evaluated as above, but cancellation in (6.9) will cause some loss of precision in the computed value of $\log(x)$. If $|\varepsilon| > 2^{-n}$, it is sufficient to evaluate $\log(\lambda x)$ and $\log(\lambda)$ to precision $2n$, for at most n bits are lost through cancellation in (6.9). On the other hand, there is no difficulty if $|\varepsilon| \leq 2^{-n}$, for then

$\log(1+\varepsilon) = \varepsilon(1+O(2^{-n}))$. When evaluating $\exp(x)$, a similar loss of precision never occurs, and it is sufficient to work with precision $n+O(\log\log(n))$, as in the evaluation of π (see Section 5). To summarize, we have proved:

Theorem 6.1. If $-\infty < a < b < \infty$, then $O(M(n)\log(n))$ operations suffice to evaluate $\exp(x)$ to precision n , uniformly for all floating-point numbers $x \in [a,b]$, as $n \to \infty$. Similarly for $\log(x)$ if $a > 0$.

7. Evaluation of Trigonometric Functions.

Suppose $\delta > 0$ fixed, and $x \in [\delta,1]$. Let $s_0 = \sin\alpha_0 = 2^{-n/2}$ and $v_0 = \tan(\varphi_0/2) = x/(1+(1+x^2)^{1/2})$, so $\tan\varphi_0 = x$. Applying the ascending Landen transformation, as for (6.1), gives

$$F(\varphi_k,\alpha_k) = \left\{\prod_{i=0}^{k-1}[(1+s_i)/2]\right\}F(\varphi_0,\alpha_0) \quad . \tag{7.1}$$

Also, from (4.5) and the choice of s_0 ,

$$F(\varphi_0,\alpha_0) = \operatorname{artan}(x) + O(2^{-n}) \quad . \tag{7.2}$$

From (4.11), $s_{i+1} \geq s_i^{1/2}$, so there is some $j \leq \log_2 n + O(1)$ such that $s_j \in [1/4, 4/5]$. Since $s_i \to 1$ with order 2 , there is some $k \leq 2\log_2 n + O(1)$ such that $1-s_k = O(2^{-n})$. From (4.6) and (4.7), $F(\varphi_k,\alpha_k) = \log\tan(\pi/4 + \varphi_k/2) + O(2^{-n})$. Thus, from (7.1) and (7.2),

$$\operatorname{artan}(x) = \left\{\prod_{i=0}^{k-1}[2/(1+s_i)]\right\}\log\tan(\pi/4+\varphi_k/2) + O(2^{-n}) \quad . \tag{7.3}$$

If we evaluate $\tan(\pi/4 + \varphi_k/2)$ as above, and use the algorithm of Section 6 to evaluate the logarithm in (7.3), we have $\operatorname{artan}(x)$ to precision n in $O(M(n)\log(n))$ operations. The algorithm may be written as follows.

Algorithm for $\operatorname{artan}(x)$, $x \in [8,1]$.

```
S ← 2^{-n/2}; V ← x/(1+(1+x^2)^{1/2}); Q ← 1;
while 1-S > 2^{-n} do
    begin Q ← 2Q/(1+S);
          W ← 2SV/(1+V^2);
          W ← W/(1+(1-W^2)^{1/2});
          W ← (V+W)/(1-VW);
          V ← W/(1+(1+W^2)^{1/2});
          S ← 2S^{1/2}/(1+S)
    end;
return Q log((1+V)/(1-V)).
```

After k iterations, $Q \le 2^k$, so at most $2\log_2 n + O(1)$ bits of precision are lost because V is small. Thus, it is sufficient to work with precision $n + O(\log(n))$, and Lemma 2.4 justifies our claim that $O(M(n)\log(n))$ operations are sufficient to obtain $\operatorname{artan}(x)$ to precision n.

If x is small, we may use the same idea as that described above for evaluating $\log(1+\varepsilon)$: work with precision $3n/2 + O(\log(n))$ if $x > 2^{-n/2}$, and use $\operatorname{artan}(x) = x(1+O(2^{-n}))$ if $0 \le x \le 2^{-n/2}$. (Actually, it is not necessary to increase the working precision if $\log((1+V)/(1-V))$ is evaluated carefully.)

Using the identity $\operatorname{artan}(x) = \pi/2 - \operatorname{artan}(1/x)$ $(x > 0)$, we can extend the domain to $[0,\infty)$. Also, since $\operatorname{artan}(-x) = -\operatorname{artan}(x)$, there is no difficulty with negative x . To summarize, we have proved the following theorem.

Theorem 7.1. $O(M(n)\log(n))$ operations suffice to evaluate $\operatorname{artan}(x)$ to precision n , uniformly for all floating-point numbers x , as $n \to \infty$.

Suppose $\theta \in [\delta , \pi/2 - \delta]$. From Lemma 3.1 and Theorem 7.1, we can solve the equation $\operatorname{artan}(x) = \theta/2$ to precision n in $O(M(n)\log(n))$ operations, and thus evaluate $x = \tan(\theta/2)$. Now $\sin\theta = 2x/(1+x^2)$, $\cos\theta = (1-x^2)/(1+x^2)$, etc., may easily be evaluated. For arguments outside $[\delta , \pi/2 - \delta]$, domain reduction techniques like those above may be used. Difficulties occur near certain integer multiples of $\pi/2$, but these may be overcome (at least for the usual floating-point number representations) by increasing the working precision. We state the following theorem for $\sin(x)$, but similar results hold for the other trigonometric functions (and also, of course, for the elliptic integrals and their inverse functions).

Theorem 7.2. If $[a,b] \subseteq (-\pi,\pi)$, then $O(M(n)\log(n))$ operations suffice to evaluate $\sin(x)$ to precision n , uniformly for all floating-point numbers $x \in [a,b]$, as $n \to \infty$.

8. Asymptotic Constants.

So far we have been concerned with order of magnitude results. In this section we give upper bounds on the constants K such that $w(n) \leq (K+o(1))M(n)\log_2 n$, where $w(n)$ is the number of operations required to evaluate π, $\exp(x)$, etc., to precision n. The following two assumptions will be made.

1. For all $\gamma > 0$ and $\varepsilon > 0$, the inequality $M(\gamma n) \leq (\gamma+\varepsilon)M(n)$ holds for sufficiently large n.
2. The number of operations required for floating-point addition, conversion between representations of different precision (at most n), and multiplication or division of floating-point numbers by small integers, is $o(M(n))$ as $n \to \infty$.

These assumptions certainly hold if a standard floating-point representation is used, and the multiplication algorithm has $M(n) \sim n(\log(n))^{\alpha}(\log\log(n))^{\beta}$ for some $\alpha \geq 0$, provided $\beta > 0$ if $\alpha = 0$.

The following result is proved in [3]. The algorithms used are similar to those of Section 2, but slightly more efficient.

Theorem 8.1. Precision n division of floating-point numbers may be performed in $(4+o(1))M(n)$ operations as $n \to \infty$, and square roots may be evaluated in $(11/2 + o(1))M(n)$ operations.

Using Theorem 8.1 and algorithms related to those of Sections 5 to 7, the following result is proved in [14].

Theorem 8.2. π may be evaluated to precision n in $(15/2 + o(1))M(n) \log_2 n$ operations as $n \to \infty$. If π and $\log 2$ are precomputed, the elementary function $f(x)$ can be evaluated to precision n in $(K + o(1))M(n) \log_2 n$ operations, where

$$K = \begin{cases} 13 & \text{if } f(x) = \log(x) \text{ or } \exp(x) , \\ 34 & \text{if } f(x) = \arctan(x) , \sin(x) , \text{ etc.}, \end{cases}$$

and x is a floating-point number in an interval on which $f(x)$ is defined and bounded away from 0 and ∞.

For purposes of comparison, note that evaluation of $\log(1+x)$ or $\log((1+x)/(1-x))$ by the usual series expansion requires $(c + o(1))M(n)n$ operations, where c is a constant of order unity (depending on the range of x and the precise method used). Since $13 \log_2 n < n$ for $n \geq 83$, the $O(M(n) \log(n))$ method for $\log(x)$ could be faster than the $O(M(n)n)$ method for n greater than a few hundred.

Acknowledgment.

The comments of Bill Gosper, Don Knuth and Daniel Shanks on an earlier draft of this paper were very useful. This work was performed while the author was visiting Stanford University.

References

[1] Abramowitz, M., and Stegun, I. A. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. National Bureau of Standards, Washington, D.C., 1964, Ch. 17.

[2] Beeler, M., Gosper, R. W., and Schroeppel, R. "Hakmem," Memo No. 239, M.I.T. Artificial Intelligence Lab., 1972, 70-71.

[3] Brent, R. P. "The complexity of multiple-precision arithmetic," Proc. Seminar on Complexity of Computational Problem Solving, (held at Australian National University, December 1974), Queensland University Press, Brisbane, 1975, 126-165.

[4] Brent, R. P. Computer Solution of Nonlinear Equations, Academic Press, New York, to appear, Ch. 6.

[5] Finkel, R., Guibas, L., and Simonyi, C. Manuscript in preparation.

[6] Fischer, M. J., and Stockmeyer, L. J. "Fast on-line integer multiplication," J. Comput. System Sci. 9 (December 1974), 317-331.

[7] Gosper, R. W. "Acceleration of series," Memo no. 304, M.I.T. Artificial Intelligence Lab., 1974.

[8] Guilloud, J. "Calculation of π to 1,000,000 places," Unpublished manuscript.

[9] Knuth, D. E. The Art of Computer Programming, Vol. 2. Addison-Wesley, Reading, Mass., 1969. Errata and addenda: Report CS 194, Computer Science Dept., Stanford University, 1970.

[10] Salamin, E. "A fast algorithm for the computation of π," submitted to Math. Comp.

[11] Schönhage, A., and Strassen, V. "Schnelle Multiplikation grosser Zahlen," Computing 7 (1971), 281-292.

[12] Schroeppel, R. Unpublished.

[13] Shanks, D., and Wrench, J. W. "Calculation of π to 100,000 decimals," Math. Comp. 16 (1962), 76-99.

[14] Brent, R. P. "Multiple-precision zero-finding methods and the complexity of elementary function evaluation," Proc. Symposium on Analytic Computational Complexity, (ed. J. F. Traub), Academic Press, New York, 1975.