THE COMPLEXITY OF MULTIPLE-PRECISION ARITHMETIC [1]

Richard P Brent

Computer Centre, Australian National University

In studying the complexity of iterative processes it is usually assumed that the arithmetic operations of addition, multiplication, and division can be performed in certain constant times. This assumption is invalid if the precision required increases as the computation proceeds. We give upper and lower bounds on the number of single-precision operations required to perform various multiple-precision operations, and deduce some interesting consequences concerning the relative efficiencies of methods for solving nonlinear equations using variable-length multiple-precision arithmetic.

1 Introduction

Traub [28] defines analytic computational complexity to be the optimality theory of analytic or continuous processes. Apart from some work by Schultz [24] on differential equations, most recent results have concerned iterative methods for the solution of nonlinear equations or systems of equations. See, for example, Brent [1,3,6], Brent, Winograd and Wolfe [7], Kung [14, 15], Kung and Traub [16–17], Paterson [21], Rissanen [22], Traub [28–32] and Wozniakowski [35, 36].

The authors just cited make the (usually implicit) assumption that arithmetic is performed with a fixed precision throughout a given computation. This is probably true for most computations programmed in Fortran or Algol 60. Suppose, though, that we are concerned with an iterative process for approximating an irrational number ζ (for example, $\sqrt{2}$, π or e) to arbitrary accuracy. The iterative process should (theoretically) generate a sequence (x_i) of real numbers, such that $\zeta = \lim_{i\to\infty} x_i$, provided no rounding errors occur. On a computing machine each x_i has to be approximated by a finite-precision machine-representable number $\widetilde{x}_i$, and $\zeta = \lim_{i\to\infty} \widetilde{x}_i$ can only hold if the precision increases indefinitely as $i \to \infty$. In practice, only a finite number of members of the sequence $(\widetilde{x}_i)$ will ever be generated, but if an accurate approximation to ζ is required it may be possible to save a large amount of computational work by using variable precision throughout the computation. This is likely to become easier to program as new languages (and possibly hardware), which allow the precision of floating-point numbers to be varied dynamically, are developed.

[1] First appeared in *The Complexity of Computational Problem Solving* (edited by R S Anderssen and R P Brent), Univ. of Queensland Press, 1976, 126–165. Retyped with minor corrections by Frances Page at Oxford University Computing Laboratory, 1999.

In Section 7 we discuss the effect of using variable precision when solving nonlinear equations. Before doing so, we consider the complexity of the basic multiple-precision arithmetic operations. We assume that a standard floating-point number representation is used, with a binary fraction of n bits. (Similar results apply for any fixed base, for example, 10.) We are interested in the case where n is much greater than the wordlength of the machine, so the fraction occupies several words. For simplicity, we assume that the exponent field has a fixed length and that numbers remain in the allowable range, so problems of exponent overflow and underflow may be neglected. Note that our assumptions rule out exotic number representations (for example, logarithmic [4] or modular [33, 34] representations) in which it is possible to perform some (but probably not all) of the basic operations faster than with the standard representation. To rule out "table-lookup" methods, we assume that a random-access memory of bounded size and a bounded number of sequential tape units are available. (Formally, our results apply to multitape Turing machines.)

In Sections 2 to 6 we ignore "constant" factors, that is factors which are bounded as $n \to \infty$. Although the constant factors are of practical importance, they depend on the computer and implementation as well as on details of the analysis. Certain machine-independent constants are studied in Sections 7 and 8.

If B is a multiple-precision operation, with operands and result represented as above (that is, "precision n" numbers), then $t_n(B)$ denotes the worst-case time required to perform B, obtaining the result with a relative error at most $2^{-n}c$, where c is independent of n. We assume that the computation is performed on a serial machine whose single-precision instructions have certain constant execution times. The following definition follows that in Hopcroft [11].

Definition 1.1 B is *linearly reducible* to C (written $B \leqq C$), if there is a positive constant K such that

$$t_n(B) \leq K t_n(C) \tag{1.1}$$

for all sufficiently large n. B is *linearly equivalent* to C (written $B \equiv C$) if $B \leqq C$ and $C \leqq B$.

In Section 2 we consider the complexity of multiple-precision addition and some linearly equivalent operations. Then, in Section 3, we show that multiple-precision division, computation of squares or square roots, and a few other operations are linearly equivalent to multiplication. Most of these results are well known [8, 9].

Sections 4 and 5 are concerned with the "operations" of evaluating exponentials, logarithms, and the standard trigonometric and hyperbolic functions (sin, artan, cosh, and so on). It turns out that most of (and probably all) these operations are linearly equivalent so long as certain restrictions are imposed.

Section 6 deals with the relationship between the four equivalence classes established in Sections 2 to 5, and several upper bounds on the complexity of operations in these classes are given. The best known constants relating operations which are linearly equivalent to multiplication are given in Section 7.

Finally, in Section 8, we compare the efficiencies of various methods for solving nonlinear equations using variable-length multiple-precision arithmetic. The relative efficiencies are different from those for the corresponding fixed-precision methods, and some of the conclusions may be rather surprising. The results of Sections 4 to 8 are mainly new.

In the analysis below, $c_1, c_2, \ldots$ denote certain positive constants which do not need to be specified further. The notation $f \sim g$ means that $\lim_{n\to\infty} f(n)/g(n) = 1$, and $f = O(g)$ means that

$|f(n)| \leq Kg(n)$ for some constant K and all sufficiently large n. Finally, the abbreviation "*mp*" stands for "variable-length multiple-precision".

2 Addition and linearly equivalent operations

Let A denote the operation of multiple-precision addition. Any reasonable implementation of floating-point addition, using at least one guard digit to avoid the possible occurrence of large relative errors, gives

$$t_n(A) \leq c_1 n \; . \tag{2.1}$$

Conversely, from the assumptions stated in Section 1, it is clear that

$$t_n(A) \geq c_2 n \; . \tag{2.2}$$

Hence, the complexity of multiple-precision addition is easily established. (For the operations discussed in Sections 3 and 5 the results are less trivial, in fact the conjectured lower bounds corresponding to (2.2) have not been proved rigorously.)

It is easy to see that bounds like (2.1) and (2.2) hold for multiple-precision subtraction, and multiplication or division of a multiple-precision number by a single-precision number (or even by any rational number with bounded numerator and denominator). Hence, all these operations are linearly equivalent to addition.

3 Multiplication and linearly equivalent operations

Let D, I, M, R and S denote the multiple-precision operations of division, taking reciprocals, multiplication, extraction of square roots and forming squares, respectively. In this section, we show that all these operations are linearly equivalent. The proofs are straightforward, but the result is surprising, as it seems intuitively obvious that taking a square root is inherently "more difficult" than forming a square, and similarly for division versus multiplication. (Some bounds on the relative difficulty of these operations are given in Section 7.)

Lemma 3.1

$$M \geqq S \geqq A \; . \tag{3.1}$$

Proof. Clearly

$$t_n(M) \geq t_n(S) \geq c_3 n \; , \tag{3.2}$$

so the result follows from (2.1).

Sharp upper bounds on $t_n(M)$ are not needed in this section, so we defer them until Section 6. Lemmas 3.2 and 3.3, although weak, are sufficient for our present purposes.

Lemma 3.2 *For all positive n,*

$$t_{2n}(M) \leq c_4 t_n(M) \; . \tag{3.3}$$

Proof. First assume that n is divisible by 3, and consider operations on the n-bit fractions only. If we can multiply n-bit numbers with relative error $2^{-n}c_0$ then we can multiply $n/3$-bit numbers exactly (assuming $2^{n/3} > 2c_0$). Thus, a $2n$-bit fraction x may be split up into

$$x = \lambda a + \lambda^2 b + \ldots + \lambda^6 f \; , \tag{3.4}$$

where $\lambda = 2^{-n/3}$ and $a, b, \ldots, f$ are integers in $\left[0, 2^{n/3}\right)$, and the product of two such $2n$-bit fractions may be formed exactly with 36 exact multiplications of $n/3$-bit numbers and some additions. Thus

$$t_{2n}(M) \le 36 t_n(M) + c_5 t_{2n}(A) \ , \tag{3.5}$$

and the result follows from Lemma 3.1. Trivial modifications to the above proof suffice, if n is not divisible by 3.

Lemma 3.3 *For some constant $c_6 < 1$,*

$$t_n(M) \le c_6 t_{8n}(M) \tag{3.6}$$

for all sufficiently large n.

Proof. If a, b, c and d are integers in $[0, 2^n)$, the identity

$$(a + \lambda b)(c + \lambda d) = ac + \lambda(bc + ad) + \lambda^2 bd \ , \tag{3.7}$$

with $\lambda = 2^{3n}$, may be used to obtain the products ac and bd from one $8n$-bit product. Thus

$$2 t_n(M) \le t_{8n}(M) + c_7 \ . \tag{3.8}$$

The result (with $c_6 = 3/4$) follows if n is sufficiently large that $t_{8n}(M) \ge 2c_7$. (We have assumed that the time required for one n-bit multiplication is half the time required for two independent n-bit multiplications, but much weaker assumptions would be sufficient.)

The following lemma will be used to estimate the work required for multiple-precision divisions and square roots.

Lemma 3.4 *Given $\alpha \in (0, 1)$, there is a constant c_8 such that, for any integers $n_0, \ldots, n_p$ satisfying*

$$1 \le n_j \le \alpha^j n \tag{3.9}$$

for $j = 0, 1, \ldots, p$, we have

$$\sum_{j=0}^{p} t_{n_j}(M) \le c_8 t_n(M) \ . \tag{3.10}$$

Proof. Let k be large enough that

$$\alpha^k \le 1/8 \ . \tag{3.11}$$

From (3.9) and (3.11),

$$t_{n_{jk}}(M) \le \alpha_6^j t_n(M) \tag{3.12}$$

for $j = 0, 1, \ldots, \lfloor p/k \rfloor$, provided n_{jk} is sufficiently large for Lemma 3.3 to be applicable. Thus,

$$\sum_{j=0}^{p} t_{n_j}(M) \le k t_n(M) \left(1 + c_6 + c_6^2 + \ldots\right) + c_7 \ , \tag{3.13}$$

where the term c_7 allows for those $t_{n_j}(M)$ for which Lemma 3.3 is not applicable. If

$$c_8 = k / \left(1 - c_6\right) + c_7 \ , \tag{3.14}$$

the result follows from (3.13).

The following lemma shows that multiple-precision multiplication is linearly equivalent to squaring. This result is essentially due to Floyd [9].

Lemma 3.5

$$M \equiv S \ . \tag{3.15}$$

Proof. Since squaring is a special case of multiplication,

$$M \geqq S \ . \tag{3.16}$$

Conversely, we may use the identity

$$4\lambda ab = (a + \lambda b)^2 - (a - \lambda b)^2 \ , \tag{3.17}$$

where λ is a power of 2 chosen so that

$$\tfrac{1}{2} \le |\lambda b/a| \le 2 \tag{3.18}$$

(unless $a = 0$ or $b = 0$). This scaling is necessary to avoid excessive cancellation in (3.17). (A detailed discussion of a similar situation is given in Brent [5].) From (3.17),

$$t_n(M) \le 2t_n(S) + 3t_n(A) + c_9, \tag{3.19}$$

so $M \leqq S$ follows from Lemma 3.1.

The next two lemmas show that multiple-precision multiplication is linearly equivalent to taking reciprocals and to division. The idea of the proof of Lemma 3.6 is to use a Newton iteration involving only multiplications and additions to approximate $1/a$. Computational work is saved by starting with low precision and approximately doubling the precision at each iteration. The basic idea is well-known and has even been implemented in hardware.

The possibility of saving work by increasing the precision at each iteration is examined more closely in Sections 7 and 8.

Lemma 3.6

$$I \leqq D \leqq M \ . \tag{3.20}$$

Proof. Consider the iteration

$$x_{j+1} = x_j \,(2 - a x_j) \tag{3.21}$$

obtained by applying Newton's method to the equation $x^{-1} - a = 0$. If

$$x_j = (1 - \varepsilon_j)\, a^{-1} \ , \tag{3.22}$$

then substitution in (3.21) shows that

$$\varepsilon_{j+1} = \varepsilon_j^2 \ , \tag{3.23}$$

so the order of convergence is two. A single-precision computation is sufficient to give an initial approximation such that $|\varepsilon_0| \le \frac{1}{2}$, and it follows from (3.23) that

$$|\varepsilon_j| \le 2^{-2^j} \tag{3.24}$$

for all $j \ge 0$.

In deriving (3.24) we have assumed that (3.21) is satisfied exactly, but a result like (3.24) holds so long as the right hand side of (3.21) is evaluated using a precision of at least 2^{j+1} bits. Thus, an n-bit approximation to a^{-1} can be obtained by performing $\lceil \log_2 n \rceil$ iterations of (3.22) with precision at least $2, 2^2, 2^3, \ldots, 2^{\lceil \log_2 n \rceil - 1}$, n at each iteration. From Lemma 3.4 (with $\alpha = \frac{1}{2}$), this gives

$$t_n(I) \le c_{10} t_n(M) \ . \tag{3.25}$$

Since $b/a = b(1/a)$, it follows that

$$t_n(D) \le c_{11} t_n(M) \ , \tag{3.26}$$

so $D \leqq M$. Since $I \leqq D$ is trivial, the proof is complete.

From $ab = a/(1/b)$ it is clear that $M \leqq D$. The proof that $M \leqq I$ is not quite so obvious, and uses the equivalences of multiplication and squaring (Lemma 3.5).

Lemma 3.7

$$M \leqq I \ . \tag{3.27}$$

Proof. We may apply the identity

$$a^2 (1 - \lambda a)^{-1} = \lambda^{-2} \left[(1 - \lambda a)^{-1} - (1 + \lambda a) \right] \tag{3.28}$$

to obtain an approximation to a^2, using only the operation of taking reciprocals, addition (or subtraction) and multiplication by powers of two. If $a \neq 0$, choose λ to be a power of two such that

$$2^{-n/3-1} < |\lambda a| < 2^{1-n/3} \ , \tag{3.29}$$

and evaluate the right hand side of (3.28), using precision n. This gives an approximation to a^2 with precision $\lceil n/3 \rceil$, so

$$S_{\lceil n/3 \rceil} \leqq I_n \ , \tag{3.30}$$

where the subscripts denote the precision. Thus, the result follows from Lemmas 3.2 and 3.5.

To conclude this section we consider the complexity of multiple-precision square roots. Results like Lemmas 3.8 and 3.9 actually hold if $x^{\frac{1}{2}}$ is replaced by x^p for any fixed rational $p \neq 0$ or 1 (we have already shown this for $p = -1$).

Lemma 3.8

$$M \leqq R \ . \tag{3.31}$$

Proof. The proof is similar to that of Lemma 3.7, using the approximation $2\lambda^{-2} \left[1 + \lambda a - (1 + 2\lambda a)^{\frac{1}{2}} \right]$ to a^2.

Lemma 3.9

$$R \leqq M \ . \tag{3.32}$$

Proof. The proof is similar to that of Lemma 3.6, using Newton's iteration

$$x_{j+1} = \tfrac{1}{2} \left(x_j + a/x_j \right) \ , \tag{3.33}$$

with precision increasing at each iteration, to approximate $\sqrt{a}$. Alternatively, it is possible to avoid multiple-precision division by using the iteration

$$x_{j+1} = x_j \left(3 - a x_j^2\right)/2 \tag{3.34}$$

to approximate $a^{-\frac{1}{2}}$, and then use $\sqrt{a} = a.a^{-\frac{1}{2}}$ to evaluate $\sqrt{a}$.

The results of Lemmas 3.5 to 3.9 may be summarized in the following:

Theorem 3.1

$$D \equiv I \equiv M \equiv R \equiv S\ . \tag{3.35}$$

4 Some regularity conditions

Before discussing the complexity of multiple-precision evaluation of exponentials, trigonometric functions, etc., we need some definitions. Throughout this section, let $\phi(x)$ be a real-valued function which is positive and monotonic increasing for all sufficiently large positive x.

Definition 4.1 $\phi \in \Phi_1$ iff, for all $\alpha \in (0,1)$, for some positive K, for all sufficiently large x and all $x_0, \ldots, x_J$ satisfying

$$1 \le x_j \le \alpha^j x \tag{4.1}$$

for $j = 0, \ldots, J$, we have

$$\sum_{j=0}^{J} \phi(x_j) \le K\phi(x)\ . \tag{4.2}$$

$\phi \in \Phi_2$ iff, for some $\alpha, \beta \in (0,1)$ and all sufficiently large x,

$$\phi(\alpha x) \le \beta\phi(x)\ . \tag{4.3}$$

$\phi \in \Phi_3$ iff, for some positive K_1, K_2 and p, there is a monotonic increasing function ψ such that

$$K_1 x^p \psi(x) \le \phi(x) \le K_2 x^p \psi(x) \tag{4.4}$$

for all sufficiently large x.

Note the similarity between the definition of Φ_1 and the statement of Lemma 3.4. In Section 5, we need to assume that the time $\phi(n)$ required to perform certain operations with precision n satisfies (4.2). The following lemmas make this assumption highly plausible. Lemma 4.1 shows that "for all α" in the definition of Φ_1 may be replaced by "for some α".

Lemma 4.1 *If, for some $\alpha \in (0,1)$ and some positive K, for all sufficiently large x and all $x_0, \ldots, x_J$ satisfying* (4.1)*, we have* (4.2)*, then $\phi \in \Phi_1$.*

Proof. Take any α_1 and α_2 in (0, 1), and suppose that (4.1) with α replaced by α_2 implies (4.2) with K replaced by K_2. Let m be a positive integer such that $\alpha_1^m \le \alpha_2$. If (4.1) holds with α replaced by α_1 for a sequence $(x_0, x_1, \ldots, x_J)$, then (4.1) also holds with α replaced by α_2 for each of the m subsequences

$$(x_0, x_m, \ldots), (x_1, x_{m+1}, \ldots), \ldots, (x_{m-1}, x_{2m-1}, \ldots)\ ,$$

so (4.2) holds with K replaced by $K_1 = mK_2$.

Lemmas 4.2 and 4.3 show that $\phi \in \Phi_2$ or $\phi \in \Phi_3$ is a sufficient condition for $\phi \in \Phi_1$. The proof of Lemma 4.2 is similar to that of Lemma 3.4 (using Lemma 4.1), so is omitted.

Lemma 4.2

$$\Phi_2 \subseteq \Phi_1 \ . \tag{4.5}$$

Lemma 4.3

$$\Phi_2 = \Phi_3 \ . \tag{4.6}$$

Proof. First suppose that $\phi \in \Phi_3$, so (4.4) holds for some function ψ and some positive K_1, K_2 and p. Choose $\alpha \in (0,1)$ such that $\beta = \alpha^p K_2/K_1 < 1$. For all sufficiently large x, we have

$$\phi(\alpha x) \le K_2 \alpha^p x^p \psi(\alpha x) \le K_2 \alpha^p x^p \psi(x) \le (K_2 \alpha^p / K_1)\, \phi(x) \le \beta \phi(x) \tag{4.7}$$

(using (4.4) and the monotonicity of ψ), so $\phi \in \Phi_2$.

Conversely, suppose that $\phi \in \Phi_2$, so (4.3) holds for all sufficiently large x (say $x \ge x_0 > 0$) and some $\alpha, \beta \in (0,1)$. Choose p small enough that $\beta \le \alpha^p$, so

$$\phi(\alpha x) \le \alpha^p \phi(x) \tag{4.8}$$

for $x \ge x_0$. Since $\phi(x)$ is positive for sufficiently large x, we may assume that $\phi(x_0) > 0$. Let $K_1 = \alpha^p, K_2 = 1$, and

$$\psi(x) = \sup_{x_0 \le y \le x} \phi(y)/y^p \tag{4.9}$$

for $x \ge x_0$. Thus, $\psi(x)$ is monotonic increasing and

$$\psi(x) \ge \phi(x)/x^p \tag{4.10}$$

so

$$\phi(x) \le K_2 x^p \psi(x) \tag{4.11}$$

for $x \ge x_0$.

By repeated application of (4.8) we have, for $k \ge 0$,

$$\phi(x)/x^p \ge \phi(\alpha^k x)/(\alpha^k x)^p \tag{4.12}$$

provided $\alpha^k x \ge x_0$. Thus, from (4.9),

$$\psi(x) = \sup_{\alpha x \le y \le x} \phi(y)/y^p \tag{4.13}$$

$$\le \phi(x)/(\alpha x)^p \tag{4.14}$$

for $x \ge x_0/\alpha$. Thus,

$$\phi(x) \ge K_1 x^p \psi(x) \tag{4.15}$$

and, in view of (4.11), $\phi \in \Phi_2$.

5 Linear equivalence of various elementary functions

In this section, we consider the multiple-precision "operations" of evaluating certain elementary functions (log, exp, sin, artan, etc). First we prove three theorems which apply under fairly general conditions. Theorem 5.1 is a generalization of Lemmas 3.7 and 3.8, and gives a simple condition under which the evaluation of $f(x)$ is at least as difficult as a multiplication (in the sense of Definition 1.1).

NOTATION. If f is a real-valued function defined on some finite interval $[a, b]$, the operation of evaluating $f(x)$ to (relative) precision n for $x \in [a, b]$ is denoted by $E^{(n)}_{[a,b]}(f)$. If there is no risk of confusion, we write simply $E_{[a,b]}(f)$ or $E(f)$. We sometimes write $t_n(f)$ for $t_n(E(f))$. $LC^{(m)}[a, b]$ is the class of functions with Lipschitz continuous m-th derivatives on $[a, b]$. We always assume that $b > a$.

Theorem 5.1 *If $f \in LC^{(2)}[a, b]$ and there is a point $x_0 \in (a, b)$ such that $f''(x_0) \neq 0$, then*

$$E(f) \geq M \ . \tag{5.1}$$

Proof. For all sufficiently small h, we have (from [2, Lemma 3.2])

$$f(x_0 + h) + f(x_0 - h) - 2f(x_0) = h^2 f''(x_0) + R(x) \ , \tag{5.2}$$

where

$$|R(x)| \leq c_{12}|h|^3 \ . \tag{5.3}$$

Let $c = f''(x_0) \neq 0$. Three evaluations of f and some additions may be used to approximate ch^2, using (5.2). If h is of order $2^{-n/3}$, the resulting approximation to ch^2 has relative error of order $2^{-n/3}$. Proceeding as in the proof of Lemma 3.5, we see that six evaluations of f and some additions may be used to approximate cxy to precision $n/3$, for any x and y. Applying this result, with x replaced by the stored constant c^{-2}, and y replaced by the computed cxy, shows that 12 evaluations of f give $c\left(c^{-2}\right)(cxy) = xy$ to precision $n/3$. The result now follows from Lemma 3.2.

REMARK. If $f''(x)$ is not constant on $[a, b]$, the point x_0 may be chosen so that $f''(x)$ is rational, so (5.2) may be used to approximate h^2, and the result follows more easily (as in the proof of Lemma 3.7).

Theorem 5.2 gives conditions under which the multiple-precision evaluation of the inverse function $g = f^{(-1)}$ of a function f is linearly reducible to the evaluation of f. (The inverse function satisfies $g(f(x)) = x$.) The condition $0 \notin [a, b]$ could be dropped, if we only required the computation of g with an absolute (rather than relative) error of order 2^{-n}.

Theorem 5.2 *If $0 \notin [a, b]$, $f \in LC^{(1)}[a, b]$, $f'(x) \neq 0$ on $[a, b]$, $E(f) \geqq M$, and*

$$t_n(f) \in \Phi_1 \ , \tag{5.4}$$

then

$$E(g) \leqq E(f) \ , \tag{5.5}$$

where $g = f^{(-1)}$ and Φ_1 is as in Definition 4.1.

Proof. Since $f'(x)$ is continuous and nonzero on $[a, b]$, there is no loss of generality in assuming that

$$f'(x) \geq c_{13} > 0 \tag{5.6}$$

on $[a, b]$. Thus, $g(y)$ exists on $[c, d] = [f(a), f(b)]$. Also, since $0 \neq [a, b]$, we have

$$|g(y)| \geq c_{14} > 0 \tag{5.7}$$

on $[c, d]$.

To estimate $g(y)$ we may solve $\psi(x) = 0$ by a discrete version of Newton's method, where

$$\psi(x) = f(x) - y \ . \tag{5.8}$$

Consider the iteration

$$x_{j+1} = x_j - \psi(x_j)/\mu_j \ , \tag{5.9}$$

where

$$\mu_j = \left(\psi(x_j + h_j) - \psi(x_j)\right)/h_j \ , \tag{5.10}$$

and the computation of μ_j and x_{j+1} is performed with precision $n_j \leq n$, giving computed values $\widehat{\mu}_j$ and $\widehat{x}_{j+1}$ respectively. If h_j is of order $2^{-n_j/2}$, then

$$|\widehat{\mu}_j - \psi'(\widehat{x}_j)| \leq \ 2^{-n_j/2} c_{15} \ , \tag{5.11}$$

and it is easy to show that

$$|\widehat{x}_{j+1} - g(y)| \ \leq \ |\widehat{x}_j - g(y)|^2 \, c_{16} \ + \ 2^{-n_j/2} \, |\widehat{x}_j - g(y)| \, c_{17} \ + \ 2^{-n_j} \, c_{18} \ . \tag{5.12}$$

Since a sufficiently good starting approximation x_0 may be found using single-precision (or at most bounded-precision) computation, (5.12) ensures that

$$|\widehat{x}_{j+1} - g(y)| \ \leq \ |\widehat{x}_j - g(y)|^2 \, c_{19} \ , \tag{5.13}$$

provided

$$|\widehat{x}_j - g(y)| \ \geq \ 2^{-n_j/2} \ . \tag{5.14}$$

Hence, we may approximately double the precision at each iteration, and (5.13) guarantees convergence of order two. A final iteration with $h_j = 2^{-n/2}$ will be sufficient to give

$$|\widehat{x}_{j+1} - g(y)| \ \leq \ 2^{-n} c_{20} \ . \tag{5.15}$$

Since $E(f) \geq M$, the result follows from (5.4), (5.7), (5.15), and Lemma 3.6.

Theorem 5.3 *If $0 \notin [a, b]$, $f \in LC^{(1)}[a, b]$, $f(x)f'(x) \neq 0$ on $[a, b]$, $g = f^{(-1)}$, $E(f) \geqq M$, $E(g) \geqq M$, $t_n(f) \in \Phi_1$, and $t_n(g) \in \Phi_1$, then*

$$E(f) \equiv E(g) \ . \tag{5.16}$$

Proof. Since $t_n(f) \in \Phi_1$, Theorem 5.2 applied to f gives $E(g) \leqq E(f)$. Similarly, applying Theorem 5.2 to $f^{(-1)}$ gives $E(f) \leqq E(g)$, so the result follows.

We are now ready to deduce the linear equivalence of *mp* evaluation of various elementary functions f_i, assuming that $t_n(f_i) \in \Phi_1$. In view of Lemmas 4.2 and 4.3, this assumption is very plausible.

Corollary 5.1 *If* $0 < a < b$, $c < d$, $1 \notin [a,b]$, $t_n\left(E_{[a,b]}(\log)\right) \in \Phi_1$, *and* $t_n\left(E_{[c,d]}(\exp)\right) \in \Phi_1$, *then*

$$E_{[a,b]}(\log) \equiv E_{[c,d]}(\exp) \ . \tag{5.17}$$

Proof. From Theorem 5.1, $E_{[a,b]}(\log) \geqq M$ and $E_{[c,d]}(\exp) \geqq M$. Also, the identities

$$\exp(-x) = 1/\exp(x) \tag{5.18}$$

and

$$\exp(\lambda x) = (\exp(x))^{\lambda} \tag{5.19}$$

(for suitable rational λ) may be used to show that $E_{[c,d]}(\exp) \equiv E_{[c',d']}(\exp)$ for any $c' < d'$. Hence, the result follows from Theorem 5.3.

REMARK. If $1 \in [a,b]$, then Theorem 5.2 shows that

$$E^{(n)}_{[c,d]}(\exp) \leqq E^{(n)}_{[a,b]}(\log) \ , \tag{5.20}$$

and a proof like that of Theorem 5.2 shows that

$$E^{(n)}_{[a,b]}(\log) \leqq E^{(2n)}_{[c,d]}(\exp) \ , \tag{5.21}$$

so the conclusion of Corollary 5.1 follows, if

$$E^{(2n)}_{[c,d]}(\exp) \equiv E^{(n)}_{[c,d]}(\exp) \ . \tag{5.22}$$

Although (5.22) is plausible, no proof of it is known. (The corresponding result for multiplication is given in Lemma 3.2.)

Corollary 5.2

$$\begin{aligned} E(\sinh) &\equiv E(\cosh) \equiv E(\tanh) \equiv E(\mathrm{arsinh}) \\ &\equiv E(\mathrm{arcosh}) \equiv \mathrm{E}(\mathrm{artanh}) \equiv \mathrm{E}(\exp) \equiv \mathrm{E}(\log) \end{aligned} \tag{5.23}$$

on any nontrivial closed intervals on which the respective functions are bounded and nonzero, assuming $t_n(\sinh) \in \Phi_1$ *etc.*

Corollary 5.3

$$E(\sin) \equiv E(\cos) \equiv E(\tan) \equiv E(\arcsin) \equiv \mathrm{E}(\arccos) \equiv \mathrm{E}(\arctan) \tag{5.24}$$

on any nontrivial closed intervals on which the respective functions are bounded and nonzero, assuming $t_n(\sin) \in \Phi_1$ *etc.*

REMARKS. The proofs of Corollaries 5.2 and 5.3 are similar to that of Corollary 5.1 (using well-known identities), so are omitted. Since $\exp(ix) = \cos(x) + i\sin(x)$, it is plausible that $E(\exp) \equiv E(\sin)$, but we have not proved this. (It is just conceivable that the evaluation of $\exp(x)$ for complex x is not linearly reducible to the evaluation of $\exp(x)$ for real x.)

6 Upper and lower bounds

In this section we give some upper and lower bounds on $t_n(A)$, $t_n(M)$, $t_n(\exp)$ and $t_n(\sin)$. Since the multiplicative constants are not specified, the bounds apply equally well to the operations which are linearly equivalent to addition, multiplication, etc. (see Sections 2 to 5). The lower bounds are trivial: $t_n\binom{\exp}{\sin} \ge c_{21}t_n(M) \ge c_{22}t_n(A) \ge c_{23}n$ (from (2.2), Lemma 3.1 and Theorem 5.1). The upper bounds are more interesting.

UPPER BOUNDS ON $t_n(M)$

The obvious algorithm for multiplication of multiple-precision numbers gives

$$t_n(M) \le c_{24}n^2 , \tag{6.1}$$

but this is not the best possible upper bound. Karatsuba and Ofman [12] showed that

$$t_n(M) \le c_{25}n^{1.58\ldots} , \tag{6.2}$$

where $1.58\ldots = \log_2 3$. The idea of the proof is that, to compute

$$(a + \lambda b)(c + \lambda d) = ac + \lambda(ad + bc) + \lambda^2 bd , \tag{6.3}$$

where λ is a suitable power of two, we compute the three products $m_1 = ac$, $m_2 = bd$, and $m_3 = (a + b)(c + d)$, and use the identity

$$ad + bc = m_3 - (m_1 + m_2) . \tag{6.4}$$

Thus, $2n$-bit integers can be multiplied with three multiplications of (at most) $(n + 1)$-bit integers, some multiplications by powers of two, and six additions of (at most) $4n$-bit integers. This observation leads to a recurrence relation from which (6.2) follows.

More complicated identities like (6.4) may be used to reduce the exponent in (6.2). Recently Schönhage and Strassen [23] showed that the exponent can be taken arbitrarily close to unity. Their method gives the best known upper bound

$$t_n(M) \le c_{26}n\ \log(n) \log\log(n) , \tag{6.5}$$

and uses an alogrithm related to the fast Fourier transform to compute certain convolutions. For a description of this and earlier methods see Knuth [13 (revised)]. Knuth conjectures that (6.5) is optimal, though the term $\log\log(n)$ is rather dubious. (It may be omitted if a machine with random-access memory of size $O(n^p)$ for some fixed positive p is assumed.) From results of Morgenstern [19] and Cook and Aanderaa [8], it is extremely probable that

$$\lim_{n\to\infty} t_n(M)/n = \infty , \tag{6.6}$$

which implies that $M \not\equiv A$, but more work remains to be done to establish this rigorously.

UPPER BOUNDS ON $t_n(\exp)$ AND $t_n(\sin)$

To evaluate $\exp(x)$ to precision n from the power series

$$\exp(\pm x) = \sum_{j=0}^{\infty} (\pm x)^j/j! , \tag{6.7}$$

it is sufficient to take $c_{27}n/\log(n)$ terms, so

$$t_n(\exp) \le c_{28}t_n(M)n/\log(n) . \tag{6.8}$$

Theorem 6.1 shows that the bound (6.8) may be reduced by a factor of order $\sqrt{n}/\log(n)$.

Theorem 6.1

$$t_n(\exp) \le c_{29}\sqrt{n}\, t_n(M) \tag{6.9}$$

and

$$t_n(\sin) \le c_{30}\sqrt{n}\, t_n(M)\ . \tag{6.10}$$

Proof. To establish (6.9), we use the identity

$$\exp(x) = (\exp(x/\lambda))^\lambda \tag{6.11}$$

with $\lambda = 2^q$, where $q = \lfloor n^{\frac{1}{2}} \rfloor$. If $[a, b]$ is the domain of x, and $c = \max(|a|, |b|)$, then

$$|(x/\lambda)^r/r!| \ \le\ 2^{-qr}\ , \tag{6.12}$$

if r is large enough that

$$c^r \le r!\ . \tag{6.13}$$

Hence, it is sufficient to take $r = \lceil n/q \rceil$ terms in the power series for $\exp(x/\lambda)$ to give an absolute error of order 2^{-n} in the approximation to $\exp(x/\lambda)$. Since $\exp(x/\lambda)$ is close to unity, the relative error will also be of the order 2^{-n} for large n. From (6.11), q squarings may be used to compute $\exp(x)$ once $\exp(x/\lambda)$ is known.

The method just described gives $\exp(x)$ to precision $n - n^{\frac{1}{2}}$, for the relative error in $\exp(x/\lambda)$ is amplified by the factor λ. This may be avoided by taking $r = \lceil n/q \rceil + 1$, and either working with precision $n + n^{\frac{1}{2}}$, or evaluating

$$\exp(|x/\lambda|) - 1 \simeq \sum_{j=1}^{r} |x/\lambda|^j/j! \tag{6.14}$$

and then using the identity

$$(1+\varepsilon)^2 - 1 = 2\varepsilon + \varepsilon^2 \tag{6.15}$$

to evaluate $\exp(|x|) - 1$ without appreciable loss of significant figures. Thus, (6.9) follows (using Lemma 3.2 if necessary).

The proof of (6.10) is similar, using the identity

$$\sin(x) = \pm 2\sin(x/2)\sqrt{1 - \sin^2(x/2)}\ , \tag{6.16}$$

q times to reduce the computation of $\sin(x)$ to that of $\sin(x/\lambda)$ (recall Lemma 3.9).

REMARKS. If x is a rational number with small numerator and denominator, the time required to sum r terms in the power series for $\exp(x/\lambda)$ is $O(rn)$, and the time required for q squarings is $O\left(q t_n(M)\right)$. Thus, choosing $r = \lfloor \sqrt{t_n(M)} \rfloor$ and $q = \lceil n/r \rceil$ gives total time $O\left(n\sqrt{t_n(M)}\right)$. It is also possible to evaluate $\exp(x)$ in this time for general x, by using a form of preconditioning to reduce the number of multiplications required to evaluate the power series for $\exp(x/\lambda)$.

A NUMERICAL EXAMPLE

The following example illustrates the ideas of Theorem 6.1. Suppose we wish to calculate e to 30 decimal places. The obvious method is to use the approximation

$$e \simeq \sum_{j=0}^{28} 1/j! \tag{6.17}$$

(since $29! \simeq 8.8 \times 10^{30}$). On the other hand

$$e \simeq \left(\sum_{j=0}^{10} \frac{1}{j!\ 256^j} \right)^{256} \tag{6.18}$$

also gives the desired accuracy (since $11!\ 256^{10} \simeq 4.8 \times 10^{31}$). Thus, the computation of 18 inverse factorials may be saved at the expense of 8 squarings.

Similarly, the computation of e to 10^6 decimal places by the obvious method requires the sum of about 205,030 inverse factorials, but the approximation

$$e \simeq \left(\sum_{j=0}^{1819} \frac{1}{j!\ 2^{1820j}} \right)^{2^{1820}}, \tag{6.19}$$

requiring only 1820 terms and 1820 squarings, is sufficiently accurate.

BASE CONVERSION

Schönhage has shown that conversion from binary to decimal or vice versa may be done in time $O\left(n\left(\log(n)\right)^2 \log\left(\log(n)\right)\right)$ (see Knuth [13, ex. 4.4.14 (revised)]). We describe his method here, as a similar idea is used below to improve Theorem 6.1.

Let $\beta > 1$ be a fixed base (e.g. $\beta = 10$), and suppose we know the base β representation of an integer x, i.e. we know the digits $d_0, \ldots, d_{t-1}$, where $0 \le d_i < \beta$ and $x = \sum_0^{t-1} d_i \beta^i$. Suppose that n-bit binary numbers can be multiplied exactly in time $M(n)$, where

$$2M(n) \le M(2n) \tag{6.20}$$

for all sufficiently large n. (This is certainly true if the Schönhage-Strassen method [13, 23] is used.) We describe how the binary representation of x may be found in time $O\left(M(n)\log(n)\right)$, where n is sufficiently large for x to be representable as an n-bit number (i.e. $2^n \ge \beta^t$).

Without changing the result, we may suppose $t = 2^k$ for some positive integer k. Let the time for conversion to binary and computation of β^{2^k} be $C(k)$. Thus, we can compute $\beta^{t/2}$ and convert the numbers $x_1 = \sum_0^{t/2-1} d_i\beta^i$ and $x_2 = \sum_{t/2}^{t-1} d_i\beta^{i-t/2}$ to binary in time $2C(k-1)$, and then $x = x_1 + \beta^{t/2}x_2$ and $\beta^t = \left(\beta^{t/2}\right)^2$ may be computed in time $2M(n/2) + O(n)$. Thus

$$C(k) \le 2C(k-1) + 2M(n/2) + O(n)\ , \tag{6.21}$$

so

$$\begin{aligned} C(k) &\le 2M(n/2) + 4M(n/4) + 8M(n/8) + \ldots + O(n\log(n)) \\ &\le O(M(n)\log(n)) \end{aligned} \tag{6.22}$$

(using (6.20)).

The proof that conversion from base 3 to base β may be done in time (6.22) is similar, and once we can convert integers it is easy to convert floating-point numbers.

COMPUTATION OF e AND π

We may regard $e - 2 = 1/2! + 1/3! + \ldots$ as given by a mixed-base fraction $0.111\ldots$, where the base is $2, 3, \ldots$ Hence, it is possible to evaluate e to precision n, using a slight modification of the above base-conversion method, in time $O(M(n)\log(n))$.

Similarly, artan(1/j) may be computed to precision n in time $O(M(n)\log^2(n))$, for any small integer $j \geq 2$, and then π may be computed from well-known identities such as

$$w = 16\,\mathrm{artan}(1/5) - 4\,\mathrm{artan}(1/239)\ . \tag{6.23}$$

The methods just described are asymptotically faster than the $O(n^2)$ methods customarily used in multiple-precision calculations of e and π (see, for example, Shanks and Wrench [25, 26]). It would be interesting to know how large n has to be before the asymptotically faster methods are actually faster. A proof that even faster methods are impossible would be of great interest, for it would imply the transcendence of e and π.

IMPROVED UPPER BOUNDS ON $t_n(\exp)$ AND $t_n(\sin)$

The following lemma uses an idea similar to that described above for base conversion and computation of e.

Lemma 6.1 *If p and q are positive integers such that $p^2 \leq q \leq 2^n$, then $\exp(p/q)$ may be computed to precision n in time $O(M(n)\log(n))$.*

Proof. The approximation

$$\exp(p/q) \simeq \sum_{j=0}^{k} \frac{(p/q)^j}{j!} \tag{6.24}$$

is sufficiently accurate if k is chosen so that

$$\frac{(p/q)^{k+1}}{(k+1)!} \leq 2^{-n} \leq \frac{(p/q)^k}{k!}\ . \tag{6.25}$$

Since $p^2 \leq q$, (6.25) gives $k!q^{k/2} \leq 2^n$, so certainly

$$k!q^k \leq 2^{2n}\ , \tag{6.26}$$

Hence, a method like that described above for the computation of e may be used, and (6.26) ensures that the integers in intermediate computations do not grow too fast.

From Lemma 6.1 it is easy to deduce Theorem 6.2, which is an improvement of Theorem 6.1 for large n. The methods used in the proof of Theorem 6.1 and the following remarks are, however, faster than that of Theorem 6.2 for small and moderate values of n.

Theorem 6.2 *If $M(n)$ satisfies* (6.20) *then*

$$t_n(\exp) \leq c_{32} M(n)\log^2(n) \tag{6.27}$$

and

$$t_n(\sin) \leq c_{33} M(n)\log^2(n)\ . \tag{6.28}$$

Proof. Without affecting the result (6.27) we may assume that $n = 2^k$ for some positive integer k. (This assumption simplifies the proof, but it is not essential.) Given an n-bit fraction $x \in [0, 1)$, we write

$$x = \sum_{i=0}^{k} p_i/q_i \ , \tag{6.29}$$

where $q_i = 2^{2^i}$ and $0 \le p_i < 2^{2^{i-1}}$ for $i = 0, 1, \ldots, k$. By Lemma 6.1, $\exp(p_i/q_i)$ can be computed, to sufficient precision, in time $O(M(n)\log(n))$, so

$$\exp(x) = \prod_{i=0}^{k} \exp(p_i/q_i) \tag{6.30}$$

can be computed in time $O(M(n)(\log(n))^2)$. This establishes (6.27), and the proof of (6.28) is similar.

Corollary 6.1

$$t_n(\exp) \le c_{34} n (\log(n))^3 \log\log(n) \tag{6.31}$$

and

$$t_n(\sin) \le c_{35} n (\log(n))^3 \log\log(n) \ . \tag{6.32}$$

Proof. This is immediate from the bound (6.5) and Theorem 6.2.

Corollary 6.2

$$t_n(E_{[a,b]}(f)) \le c_{36} n (\log(n))^3 \log\log(n) \ ,$$

where

$$\begin{aligned} f(x) \ = \ & \log(x), \exp(x), \sin(x), \cos(x), \tan(x), \sinh(x), \\ & \cosh(x), \tanh(x), \mathrm{arsin(x)}, \mathrm{artan(x)}, \mathrm{arsinh(x)}, \end{aligned}$$

etc, and $[a, b]$ is any finite interval on which $f(x)$ is bounded.

Proof. This follows from (6.5), Corollaries 5.1 (and the note following), 5.2, 6.1, and Lemma 3.2.

7 Best constants for operations equivalent to multiplication

In this section, we consider in more detail the relationship between the *mp* operations D, I, M, R, and S defined in Section 3. It is convenient to consider also the operation Q of forming inverse square roots (i.e., $y \leftarrow x^{-\frac{1}{2}}$). From Theorem 3.1, if we can perform any one of these operations (say Y) to precision n in time $t_n(Y)$, then the time required to perform any of the other operations to precision n is at most a constant multiple of $t_n(Y)$.

Definition 7.1 C_{XY} is the minimal constant such that, for all positive ε and all sufficiently large n, the operation X can be performed (to precision n) in time $(C_{XY} + \varepsilon)t_n(Y)$ if Y can be performed in time $t_n(Y)$, where $X, Y = D, I, M, Q, R$ or S.

The following inequalities are immediate consequences of Definition 7.1:

$$C_{XY} C_{YZ} \ge C_{XZ} \tag{7.1}$$

and

$$C_{XY}C_{YX} \geq C_{XX} = 1\ . \tag{7.2}$$

ASSUMPTIONS

To enable us to give moderate upper bounds on the constants C_{XY}, it is necessary to make the following plausible assumption (compare (4.3), (6.20)) throughout this section: for all positive α and ε, and all sufficiently large n,

$$t_{\alpha n}(Y) \leq (\alpha + \varepsilon)t_n(Y) \tag{7.3}$$

for $Y = D, I, M, Q, R$ and S. We also assume (6.6).

Table 7.1 gives the best known upper bounds on the constants C_{XY}. Space does not permit a detailed proof of all these upper bounds, but the main ideas of the proof are sketched below.

TABLE 7.1 Upper bounds on C_{XY}

	$X = D$	I	M	Q	R	S
$Y = D$	1.0	1.0	2.0	3.0	2.0	2.0
I	7.0	1.0	6.0	15.0	14.0	3.0
M	4.0	3.0	1.0	4.5	5.5	1.0
Q	10.0	4.0	6.0	1.0	5.0	3.0
R	7.5	6.0	6.0	3.0	1.0	3.0
S	7.5	5.5	2.0	7.0	9.0	1.0

$\underline{C_{IM} \leq 3}$

Use the Newton iteration

$$x_{i+1} = x_i - x_i(ax_i - 1) \tag{7.4}$$

to approximate $1/a$ using multiplications. At the last iteration it is necessary to compute ax_i to precision n, but $x_i(ax_i - 1)$ only to (relative) precision $n/2$. Since the order of convergence is 2, the assumptions (7.3) (with $\alpha = \frac{1}{2}$) and (6.6) give

$$C_{IM} \leq (1 + \tfrac{1}{2})(1 + \tfrac{1}{2} + \tfrac{1}{2} + \ldots) = 3\ . \tag{7.5}$$

$\underline{C_{QM} \leq 4.5}$

Use the third-order iteration

$$x_{i+1} = x_i - \tfrac{1}{2}x_i\left(\varepsilon_i - \tfrac{3}{4}\varepsilon_i^2\right) \tag{7.6}$$

where

$$\varepsilon_i = ax_i^2 - 1 \tag{7.7}$$

to approximate $a^{-\frac{1}{2}}$. At the last iteration it is necessary to compute ax_i^2 to precision n, ε_i^2 to precision $n/3$, and $x_i\left(\varepsilon_i - \frac{3}{4}\varepsilon_i^2\right)$ to precision $2n/3$. Thus

$$C_{QM} \leq (2 + \tfrac{1}{3} + \tfrac{2}{3})(1 + \tfrac{1}{3} + \tfrac{1}{9} + \ldots) = \tfrac{9}{2}\ . \tag{7.8}$$

Note that this bound is sharper than the bound $C_{QM} \leq 5$ which may be obtained from the second-order iteration

$$x_{i+1} = x_i - \tfrac{1}{2} x_i \varepsilon_i \; . \tag{7.9}$$

$\underline{C_{RD} \leq 2}$

Use Newton's iteration

$$x_{i+1} = \tfrac{1}{2}(x_i + a/x_i) \tag{7.10}$$

to approximate $\sqrt{a}$.

$\underline{C_{MS} \leq 2}$

This follows from (3.19) and our assumptions.

$\underline{C_{IS} \leq 5.5}$

Use the third-order iteration

$$x_{i+1} = x_i - x_i \left(\varepsilon_i - \varepsilon_i^2\right) \tag{7.11}$$

where

$$\varepsilon_i = a x_i - 1 \tag{7.12}$$

to approximate $1/a$.

$\underline{C_{QS} \leq 7}$

Use the third-order iteration (7.6).

$\underline{C_{SI} \leq 3}$

From the proof of Lemma 3.7,

$$t_{n/3}(S) \leq t_n(I) + O(n) \; . \tag{7.13}$$

The result follows from the assumption (7.3) with $\alpha = 3$. (This is the first time we have used (7.3) with $\alpha > 1$. The assumption is plausible in view of the Schönhage-Strassen bound (6.5).) Upper bounds on C_{SQ} and C_{SR} follow similarly.

$\underline{C_{MI} \leq 6}$

This follows from (7.1) and our bounds on C_{MS} and C_{SI}. Similarly for the bounds on C_{MQ}, C_{MR} and C_{RI}.

$\underline{C_{QR} \leq 3}$

Use the identity

$$a^{-\frac{1}{2}} = \frac{1}{\lambda}\left(\sqrt{a+\lambda} - \sqrt{a-\lambda}\right) + O\left(\lambda^2/a^{5/2}\right) \; , \tag{7.14}$$

where λ is a power of 2 such that

$$2^{-n/3-1} \leq \lambda/a \leq 2^{1-n/3} \; . \tag{7.15}$$

Thus

$$t_{2n/3}(Q) \le 2t_n(R) + O(n) \ , \tag{7.16}$$

and the result follows from (7.3).

$\underline{C_{DR} \le 7.5}$

Use the identity

$$b/a = \frac{1}{\lambda}\left(\sqrt{a^2+\lambda b} - \sqrt{a^2-\lambda b}\right) + O\left(\lambda^2 b^3/a^5\right) \ , \tag{7.17}$$

where λ is a power of 2 such that (for $b \ne 0$)

$$2^{-n/3-1} \le \lambda b/a^2 \le 2^{1-n/3} \ . \tag{7.18}$$

Thus

$$t_{2n/3}(D) \le t_n(S) + 2t_n(R) + O(n) \ , \tag{7.19}$$

and the result follows.

$\underline{C_{IR} \le 6}$

$$a^{-1} = (a^2)^{-\frac{1}{2}} \ , \tag{7.20}$$

so

$$C_{IR} \le C_{SR} + C_{QR} \le 6 \ . \tag{7.21}$$

The bound on C_{IQ} also follows from (7.20), and then the bound on C_{RQ} follows from $a^{\frac{1}{2}} = \left(a^{-1}\right)^{-\frac{1}{2}}$.

8 Comparison of some *mp* methods for nonlinear equations

In this section, we briefly consider methods for finding multiple-precision solutions of non-linear equations of the form

$$f(x) = 0, \tag{8.1}$$

where $f(x)$ can be evaluated for any x in some domain. Additional results are given in [38].

There are many well-known results on the efficiency of various methods for solving (8.1), e.g., Hindmarsh [10], Ostrowski [20], Traub [27] and the references given in Section 1, but the results are only valid if arithmetic operations (in particular the evaluation of $f(x), f'(x)$ etc.) require certain constant times. The examples given below demonstrate that different considerations are relevant when multiple-precision arithmetic of varying precision is used.

For simplicity, we restrict attention to methods for finding a simple zero ζ of f by evaluating f at various points. We assume that f has sufficiently many continuous derivatives in a neighbourhood of ζ, but the methods considered do not require the evaluation of these derivatives.

Since $f(x)$ is necessarily small near ζ, it is not reasonable to assume that $f(x)$ can be evaluated to within a small *relative* error near ζ. In this section, an evaluation of f "with precision n" means with an *absolute* error of order 2^{-n}. We suppose that such an evaluation requires time $w(n) = t_n(E(f))$, where

$$w(cn) \sim c^{\alpha} w(n) \tag{8.2}$$

for some constant $\alpha > 1$ and all positive c. Since $\alpha > 1$, the bound (6.5) and condition (8.2) give

$$\lim_{n\to\infty} t_n(M)/w(n) = 0 \ , \tag{8.3}$$

so we may ignore the time required for a fixed number of multiplications and divisions per iteration, and merely consider the time required for function evaluations. Our results also apply if $\alpha = 1$, so long as (8.3) holds. (For example, the evaluation of $\exp(x)$ by the method of Corollary 6.1 requires time $w(n) \sim c_{37} n (\log(n))^3 \log\log(n)$, which satisfies (8.2) with $\alpha = 1$, and also satisfies (8.3).)

Definition 8.1 If an *mp* zero-finding method requires time $t(n) \sim C(\alpha)w(n)$ to approximate $\zeta \neq 0$ with precision n, where $w(n)$ and ζ are as above, then $C(\alpha)$ is the *asymptotic constant* of the method. (Not to be confused with the asymptotic error constant as usually defined for fixed-precision methods [2].)

Given several *mp* methods with various asymptotic constants, it is clear that the method with minimal asymptotic constant is the fastest (for sufficiently large n). The method which is fastest may depend on α, as the following examples show.

DISCRETE NEWTON *mp* METHODS

Consider iterative methods of the form

$$x_{i+1} = x_i - f(x_i)/g_i \ , \tag{8.4}$$

where g_i is a finite-difference approximation to $f'(x_i)$. If $\varepsilon_i = |x_i - \zeta|$ is sufficiently small, $f(x_i)$ is evaluated with absolute error $O\left(\varepsilon_i^2\right)$, and

$$g_i = f'(x_i) + O(\varepsilon_i) \ , \tag{8.5}$$

then

$$|x_{i+1} - \zeta| = O\left(\varepsilon_i^2\right) \ , \tag{8.6}$$

so the method has order (at least) 2.

The simplest method of estimating $f'(x_i)$ to sufficient accuracy is to use the one-sided difference

$$g_i = \frac{f(x_i + h_i) - f(x_i)}{h_i} \ , \tag{8.7}$$

where h_i is of order ε_i, and the evaluation of $f(x_i + h_i)$ and $f(x_i)$ are performed with an absolute error $O\left(\varepsilon_i^2\right)$. Thus, to obtain ζ to precision n by this method (N_1), we need two evaluations of f to precision n (at the last iteration), preceded by two evaluations to precision $n/2$, etc. (The same idea is used above, in the proof of Theorem 5.2.) The time required is

$$t(n) \sim 2w(n) + 2w(n/2) + 2w(n/4) + \ldots \ . \tag{8.8}$$

Thus, from (8.2) and Definition 8.1, the asymptotic constant is

$$C_{N_1}(\alpha) = 2(1 + 2^{-\alpha} + 2^{-2\alpha} + \ldots) = 2/(1 - 2^{-\alpha}) \ . \tag{8.9}$$

Since

$$2 < C_{N_1}(\alpha) \leq 4 \ , \tag{8.10}$$

the time required to solve (8.1) to precision n is only a small multiple of the time required to evaluate f to the same precision. The same applies for the methods described below.

Using (8.7) is not necessarily the best way to estimate $f'(x_i)$. Let p be a fixed positive integer, and consider estimating $f'(x_i)$ by evaluating f at the points

$$x_i - \lfloor p/2 \rfloor h_i, x_i - (\lfloor p/2 \rfloor - 1)h_i, \ldots, x_i + \lceil p/2 \rceil h_i \ .$$

(The points need not be equally spaced so long as their minimum and maximum separations are of order h_i.) Let g_i be the derivative (at x_i) of the Lagrange interpolating polynomial agreeing with f at these points. Since estimates $f'(x_i)$ with truncation error $O\left(h_i^p\right)$, we need h_i of order $\varepsilon_i^{1/p}$. Then, to ensure that (8.5) holds, the function evaluations at the above points must be made with absolute error $O\left(\varepsilon_i^{1+1/p}\right)$. Thus to obtain ζ to precision n by this method (N_p) we need one evaluation f to precision n and p evaluations to precision $n(1+1/p)/2$, preceded by one evaluation precision $n/2$ and p to precision $n(1+1/p)/4$, etc. The asymptotic constant is

$$C_N(p, \alpha) = \left(1 + p\left(\frac{p+1}{2p}\right)^{\alpha}\right) \Big/ (1 - 2^{-\alpha}) \ . \tag{8.11}$$

Let

$$C_N(\alpha) = \min_{p=1,2,\ldots} C_N(p, \alpha) \ , \tag{8.12}$$

so the "optimal mp discrete Newton method" has asymptotic constant $C_N(\alpha)$. From (8.11), the p which minimizes $C_N(p, \alpha)$ also minimizes $p^{1/\alpha}(1 + 1/p)$, so the minimum for $\alpha > 1$ occurs at $p = \lfloor \alpha - 1 \rfloor$ or $\lceil \alpha - 1 \rceil$. In fact, $p = 1$ is optimal if

$$1 \le \alpha < \log(2)/\log(4/3) = 2.4094\ldots \ , \tag{8.13}$$

and $p \ge 2$ is optimal if

$$\frac{\log(1 - p^{-1})}{\log(1 - p^{-2})} < \alpha < \frac{\log(1 + p^{-1})}{\log(1 + 1/(p(p+2)))} \ . \tag{8.14}$$

The result that method N_2 is more efficient than method N_1 if $\alpha > 2.4094\ldots$ is interesting, for N_2 requires one more function evaluation per iteration than N_1, and has the same order of convergence. The reason is that not all the function evaluations need to be as accurate for method N_2 as for method N_1. We give below several more examples where methods with lower order and/or more function evaluations per iteration are more efficient than methods with higher order and/or less function evaluations per iteration.

For future reference, we note that

$$1 < C_N(\alpha) \le 4 \ , \tag{8.15}$$

$$C_N(1) = 4 \ , \tag{8.16}$$

and

$$C_N(\alpha) - 1 \sim e\alpha 2^{-\alpha} \tag{8.17}$$

as $\alpha \to \infty$.

A CLASS OF mp SECANT METHODS

It is well-known that the secant method is more efficient that the discrete Newton method for solving nonlinear equations with fixed-precision arithmetic [2, 20]. For mp methods the comparison depends on the exponent α in (8.2).

Let k be a fixed positive integer and p_k the positive real root of

$$x^{k+1} = 1 + x^k \ . \tag{8.18}$$

The iterative method S_k is defined by

$$x_{i+1} = x_i - f(x_i)\left(\frac{x_i - x_{i-k}}{f(x_i) - f(x_{i-k})}\right) \ , \tag{8.19}$$

where the function evaluations are performed to sufficient accuracy to ensure that the order of convergence is at least p_k. Thus, S_1 is the usual secant method with order $p_1 = \frac{1+\sqrt{5}}{2} = 1.618\ldots; S_2, S_3$ etc. are methods with lower orders $p_2 = 1.4655\ldots, p_3 = 1.3802\ldots$, etc. With fixed-precision S_1 is always preferable to S_2, S_3 etc., but this is not always true if mp arithmetic is used.

Suppose i and k fixed, $\delta > 0$ small, and write $\varepsilon = |x_{i-k} - \zeta|$ and $p = p_k - \delta$. Since the order of convergence is at least p, we have

$$|x_i - \zeta| = O\left(\varepsilon^{p^k}\right) \ , \tag{8.20}$$

$$|x_{i+1} - \zeta| = O\left(\varepsilon^{p^{k+1}}\right) \ , \tag{8.21}$$

$$|x_i - x_{i-k}| = O(\varepsilon) \ , \tag{8.22}$$

and

$$|f(x_i)| = O\left(\varepsilon^{p^k}\right) \ . \tag{8.23}$$

For the approximate evaluation of the right side of (8.19) to give order p, the absolute error in the evaluation of $f(x_i)$ must be $O\left(\varepsilon^{p^{k+1}}\right)$, and the relative error in the evaluation of $(f(x_i) - f(x_{i-k}))/(x_i - x_{i-k})$ must be $O\left(\varepsilon^{p^{k+1}-p^k}\right)$, so the absolute error in the evaluation of $f(x_{i-k})$ must be $O\left(\varepsilon^{p^{k+1}-p^k+1}\right)$. From (7.18), for δ sufficiently small,

$$p^{k+1} - p^k + 1 > p \ , \tag{8.24}$$

so the evaluation of ζ to precision n by method S_k requires evaluations of f to precision $n, n/p, n/p^2, \ldots, n/p^{k-1}, 2n/p^{k+1}, 2n/p^{k+2}$, etc. Thus, the asymptotic constant is

$$\begin{aligned} C_S(k,\alpha) &= 1 + p^{-\alpha} + \ldots + p^{(1-k)\alpha} + (2p^{-(k+1)})^\alpha(1 + p^{-\alpha} + \ldots) \\ &= \frac{1 - p^{-k\alpha} + (2p^{-(k+1)})^\alpha}{1 - p^{-\alpha}} \ , \end{aligned} \tag{8.25}$$

where (after letting $\delta \to 0$) $p = p_k$ satisfies (8.18).

We naturally choose k to minimize $C_S(k,\alpha)$, giving the "optimal mp secant method" with asymptotic constant

$$C_S(\alpha) = \min_{k=1,2,\ldots} C_S(k,\alpha) \ . \tag{8.26}$$

The following lemmas show that the optimal secant method is S_1 if $\alpha < 4.5243\ldots$, and S_2 if $\alpha > 4.5243\ldots$

Lemma 8.1

$$C_S(k,1) = 3 + p_k^k - p_k \ . \tag{8.27}$$

Proof. Easy from (8.18) and (8.25).

Lemma 8.2

$$C_S(k,\alpha) - 1 \sim \begin{cases} (3-\sqrt{5})^\alpha & \text{if } k = 1 \ , \\ p_k^{-\alpha} & \text{if } k \geq 2 \ , \end{cases} \tag{8.28}$$

as $\alpha \to \infty$.

Proof. From (8.25),

$$C_S(k,\alpha) - 1 \sim p_k^{-\alpha} - p_k^{-k\alpha} + \left(2p_k^{-(k+1)}\right)^\alpha \tag{8.29}$$

as $\alpha \to \infty$. If $k \geq 2$ then, from (8.18),

$$p_k^k = p_k^{-1} + p_k^{k-1} \geq p_k^{-1} + p_k > 2 \ , \tag{8.30}$$

so

$$p_k^{-1} > 2p_k^{-(k+1)} \ . \tag{8.31}$$

Thus, the result for $k \geq 2$ follows from (8.29). The result for $k = 1$ also follows from (8.29), for $2p_1^{-2} = 3 - \sqrt{5}$.

Lemma 8.3

$$C_S(\alpha) = \begin{cases} C_S(1,\alpha) & \text{if } 1 \leq \alpha \leq \alpha_0 \ , \\ C_S(2,\alpha) & \text{if } \alpha \geq \alpha_0 \ , \end{cases} \tag{8.32}$$

where $\alpha_0 = 4.5243\ldots$ *is the root of*

$$C_S(1,\alpha_0) = C_S(2,\alpha_0) \ . \tag{8.33}$$

Proof. The details of the proof are omitted, but we note that the result follows from Lemmas 8.1 and 8.2 for (respectively) small and large values of α.

From (8.25), $C_S(k,\alpha)$ is a monotonic decreasing function of α, so the same is true of $C_S(\alpha)$. Thus, from Lemmas 8.1, 8.2 and 8.3,

$$1 < C_S(\alpha) \leq 3 \ , \tag{8.34}$$

$$C_S(1) = 3 \ , \tag{8.35}$$

and

$$C_S(\alpha) - 1 \sim p_2^{-\alpha} = (0.6823\ldots)^\alpha \tag{8.36}$$

as $\alpha \to \infty$. Comparing these results with (8.15) to (8.17), we see that the optional *mp* secant method is more efficient than the optimal *mp* discrete Newton method for small α, but less efficient for large α. (The changeover occurs at $\alpha = 8.7143\ldots$)

AN *mp* METHOD USING INVERSE QUADRATIC INTERPOLATION

For fixed-precision arithmetic the method of inverse quadratic interpolation [2] is slightly more efficient than the secant method, for it has order $P_Q = 1.8392\ldots > 1.6180\ldots$, and requires the same number (one) of function evaluations per iteration. For *mp* arithmetic, it turns out that inverse quadratic interpolation (Q) is always more efficient than the secant method S_1, but it is less efficient than the secant method S_2 if $\alpha > 5.0571\ldots$

Since the analysis is similar to that for method S_1 above, the details are omitted. The order p_Q is the positive real root of

$$x^3 = 1 + x + x^2 \ . \tag{8.37}$$

For brevity, we write $\sigma = 1/p_Q = 0.5436\ldots$

To evaluate ζ to precision n by method Q requires evaluations of f to precision n, $(1-\sigma+\sigma^2)n$, and $\sigma^j(1-\sigma-\sigma^2+2\sigma^3)n$ for $j = 0, 1, 2, \ldots$ Hence, the asymptotic constant is

$$\begin{aligned} C_Q(\alpha) &= 1 + (1-\sigma+\sigma^2)^\alpha + (1-\sigma-\sigma^2+2\sigma^3)^\alpha/(1-\sigma^\alpha) \\ &= 1 + (1-\sigma+\sigma^2)^\alpha + (3\sigma^3)^\alpha/(1-\sigma^\alpha) \end{aligned} \tag{8.38}$$

from (8.31). Corresponding to the results (8.15) to (8.17) and (8.34) to (8.36), we have that $C_Q(\alpha)$ is monotonic decreasing,

$$1 < C_Q(\alpha) \le C_Q(1) = \tfrac{1}{2}(7 - 2\sigma - \sigma^2) = 2.8085\ldots \ , \tag{8.39}$$

and

$$C_Q(\alpha) - 1 \sim (1-\sigma+\sigma^2)^\alpha = (0.7519\ldots)^\alpha \tag{8.40}$$

as $\alpha \to \infty$. Method Q is more efficient than the optimal *mp* secant method if $\alpha < 5.0571\ldots$, and more efficient than the optimal *mp* discrete Newton method if $\alpha < 7.1349\ldots$ We do not know any *mp* method which is more efficient than method Q for α close to 1.

OTHER *mp* METHODS USING INVERSE INTERPOLATION

Since inverse quadratic interpolation is more efficient than linear interpolation (at least for α close to 1), it is natural to ask if inverse cubic or higher degree interpolation is even more efficient. Suppose $\frac{1}{2} \le \mu < 1$, and consider an inverse interpolation method I_μ with order $1/\mu$. In particular, consider the method I_μ which uses inverse interpolation at $x_i, x_{i-1}, \ldots, x_{i-k}$ to generate x_{i+1}, where k is sufficiently large, and the function evaluations at $x_i, \ldots, x_{i-k}$ are sufficiently accurate to ensure that the order is at least $1/\mu$ and, in general, no more than $1/\mu$. (The limiting case $I_{1/2}$ is the method which uses inverse interpolation through all previous points $x_0, x_1, \ldots, x_i$ to generate x_{i+1}.)

By an analysis similar to those above, it may be shown that the asymptotic constant of method I_μ is

$$C_I(\mu, \alpha) = \sum_{j=0}^{\infty} (s_j(\mu))^\alpha \ , \tag{8.41}$$

where $s_0(\mu) = 1$ and

$$s_j(\mu) = \max\left[\mu s_{j-1}(\mu), 1 + j\mu^{j+1} - \mu(1-\mu^j)/(1-\mu)\right] \tag{8.42}$$

for $j = 1, 2, \ldots$ Space does not allow a proof of (8.41), but related results are given in [20, Appendix H]. We note the easily verified special cases

$$C_I\left(\frac{\sqrt{5}-1}{2},\ \alpha\right) = C_S(1, \alpha) \tag{8.43}$$

and

$$C_I(\sigma, \alpha) = C_Q(\alpha) \ . \tag{8.44}$$

The method with maximal order (see [7]) is $I_{1/2}$, with asymptotic constant

$$C_I(\tfrac{1}{2},\alpha) = \sum_{j=2}^{\infty} (j2^{1-j})^{\alpha} \ . \tag{8.45}$$

The "optimal *mp* inverse interpolatory method" is the method I_μ with $\mu(\alpha)$ chosen to minimize $C_I(\mu,\alpha)$, so its symptotic constant is

$$C_I(\alpha) = \min_{\frac{1}{2}\le\mu\le 1} C_I(\mu,\alpha) \ . \tag{8.46}$$

The following lemma shows that the optimal choice is $\mu = \sigma$, corresponding to the inverse quadratic method Q discussed above, if $\alpha \le 4.6056\ldots$

Lemma 8.4 *If $C_I(\alpha) = C_I(\mu(\alpha),\alpha)$ then*

$$\mu(\alpha) = \sigma = 0.5436\ldots \ \textit{if}\ 1 \le \alpha \le 4.6056\ldots \ , \tag{8.47}$$

$\mu(\alpha)$ is a monotonic decreasing function of α, and

$$\lim\nolimits_{\alpha\to\infty} \mu(\alpha) = \tfrac{1}{2} \ . \tag{8.48}$$

From (8.39),

$$C_I(\tfrac{1}{2},\alpha) - 1 \sim \left(\tfrac{3}{4}\right)^{\alpha} \tag{8.49}$$

as $\alpha \to \infty$, so Lemma 8.4 shows that the optimal inverse interpolatory is more efficient than methods S_1 and Q (as expected), but less efficient than method S_2 or the optimal discrete Newton method, for large α. In fact $C_I(\alpha) < C_S(\alpha)$ for $1 \le \alpha < 5.0608\ldots$

A LOWER BOUND FOR $C(\alpha)$

The following theorem shows that $C(\alpha) \ge 1$ for all useful *mp* methods. The results above (e.g. (7.17)) show that the constant "1" here is best possible, as methods with $C(\alpha) \to 1$ as $\alpha \to \infty$ are possible. The minimal value of $C(\alpha)$ for any finite α is an open question.

Theorem 8.1 *If an mp method is well-defined and converges to a zero of the functions $f_1(x) = F(x) - y$ and $f_2(y) = F^{(-1)}(y) - x$, where x and y are restricted to nonempty domains D_x and D_y, and F is some invertible mapping of D_x onto D_y such that $t_n(E(F))$ satisfies* (8.2), *then the asymptotic constant of the method satisfies $C(\alpha) \ge 1$.*

Proof. If $C(\alpha) < 1$ then, by solving $f_1(x) = 0$, we can evaluate $F^{(-1)}(y)$ (for y in D_y) in time less than $t_n(E(F))$, for all sufficiently large n. Applying the same argument to $f_2(y)$, we can evaluate $F = (F^{(-1)})^{(-1)}$ in time less than $t_n(E(F^{(-1)}))$. Hence, for large n we have

$$t_n(E(F)) < t_n(E(F^{(-1)})) < t_n(E(F)) \ , \tag{8.50}$$

a contradiction. Hence, $C(\alpha) \ge 1$.

Conjecture 8.1 *For all mp methods (using only function evaluations) which are well-defined and convergent for some reasonable class of functions with simple zeros,*

$$C(\alpha) \ge 1/(1 - 2^{-\alpha}) \ . \tag{8.51}$$

SUMMARY OF *mp* ZERO-FINDING METHODS

Of the methods described in this section, the most efficient are:

1. optimal inverse interpolation, if $1 \le \alpha \le 5.0608\ldots$ (equivalent to inverse quadratic interpolation, if $1 \le \alpha \le 4.6056\ldots$) ;
2. optimal secant method (method S_2), if $5.0608\ldots < \alpha \le 8.7143\ldots$;
3. optimal discrete Newton, if $8.7143\ldots < \alpha$.

For practical purposes, the inverse quadratic interpolation method is to be recommended, for it is easy to program, and its asymptotic $C_Q(\alpha)$ is always within 3.2% of the least constant for the methods above. Numerical values of the asymptotic constants, for various values of α, are given to $4D$ in Table 8.1. The smallest constant for each α is italicized.

TABLE 8.1 Aysmptotic constants for various *mp* methods

α	$C_N(\alpha)$	$C_S(1,\alpha)$	$C_S(2,\alpha)$	$C_Q(\alpha)$	$C_I(\alpha)$	$C_I(\frac{1}{2},\alpha)$
1.0	4.0000	3.0000	3.6823	*2.8085*	*2.8085*	3.0000
1.1	3.7489	2.8093	3.4256	*2.6484*	*2.6484*	2.8193
1.5	3.0938	2.2987	2.7241	*2.2108*	*2.2108*	2.3219
2.0	2.6667	1.9443	2.2209	*1.8954*	*1.8954*	1.9630
3.0	2.1071	1.5836	1.6935	*1.5586*	*1.5586*	1.5856
4.0	1.6988	1.3988	1.4248	*1.3789*	*1.3789*	1.3898
5.0	1.4260	1.2860	1.2694	1.2677	*1.2676*	1.2718
6.0	1.2529	1.2105	*1.1741*	1.1936	1.1930	1.1946
7.0	1.1469	1.1573	*1.1137*	1.1420	1.1410	1.1416
8.0	1.0838	1.1185	*1.0748*	1.1051	1.1039	1.1041
9.0	*1.0471*	1.0898	1.0495	1.0782	1.0770	1.0771
10.0	*1.0262*	1.0682	1.0328	1.0584	1.0573	1.0573
15.0	*1.0012*	1.0176	1.0043	1.0139	1.0134	1.0134
20.0	*1.0001*	1.0046	1.0006	1.0033	1.0032	1.0032

NOTE ADDED IN PROOF. Theorem 6.2 and its corollaries may be improved by a factor $\log(n)$, as described in [37] and [38].

References

[1] Brent, R.P. The computational complexity of iterative methods for systems of nonlinear equations. In *Complexity of Computer Computations* (edited by R.E. Miller and J.W. Thatcher). Plenum Press, New York, 1972, 61–71.

[2] Brent, R.P. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, New Jersey, 1973.

[3] Brent, R.P. Some efficient algorithms for solving systems of nonlinear equations. *SIAM Numer. Anal.* **10**, 327–344, 1973.

[4] Brent, R.P. On the precision attainable with various floating-point number systems. *IEEE Trans. Comp.* **C-22**, 601–607, 1973.

[5] Brent, R.P. Error analysis of algorithms for matrix multiplication and triangular decomposition using Winograd's identity. *Numer. Math.* **16**, 145–156, 1970.

[6] Brent, R.P. *Numerical solution of nonlinear equations.* Computer Sci. Dept., Stanford University, March 1975, 189 *pp.*

[7] Brent, R.P., Winograd, S. and Wolfe, P. Optimal iterative processes for rootfinding. *Numer. Math.* **20**, 327–341, 1973.

[8] Cook, S.A. and Aanderaa, S.O. On the minimum complexity of functions. *Trans. Amer. Math. Soc.* **142**, 291–314, 1969.

[9] Floyd, R.W. Unpublished notes.

[10] Hindmarsh, A.C. Optimality in a class of rootfinding algorithms. *SIAM J. Numer. Anal.* **9**, 205–214, 1972.

[11] Hopcroft, J.E. Complexity of computer computations. In *Information Processing 74.* North-Holland, Amsterdam, 1974, 620–626.

[12] Karatsuba, A. and Ofman, Y. Multiplication of multidigit numbers on automata (Russian). *Dokl. Akad. Nauk SSSR* **145**, 293-294, 1962.

[13] Knuth, D.E. *The Art of Compter Programming*, Vol. II, *Seminumerical Algorithms*. Addison Wesley, Reading, Massachusetts, 1969. Errata and addenda: Report CS 194, Computer Sci. Department, Stanford University, 1970.

[14] Kung, H.T. The computational complexity of algebraic numbers. *SIAM J. Numer. Anal.* (to appear).

[15] Kung, H.T. A bound on the multiplicative efficiency of iteration. *J. Computer & System Sciences* **7**, 334–342, 1973.

[16] Kung, H.T. and Traub, J.F. Optimal order of one-point and multipoint iteration. *J. ACM* **21**, 643–651, 1974.

[17] Kung, H.T. and Traub, J.F. Computational complexity of one-point and multipoint iteration. In *Complexity of Real Computations* (edited by R. Karp). Amer. Math. Soc., Providence, Rhode Island, 1974, 149–160.

[18] Kung, H.T. and Traub, J.F. Optimal order and efficiency for iterations with two evaluations. Tech. Report, Department of Computer Science, Carnegie-Mellon University, 1973.

[19] Morgenstern, J. The linear complexity of computation. *J. ACM* **20**, 305–306 (1973).

[20] Ostrowski, A.M. *Solution of Equations in Euclidean and Banach Spaces.* Academic Press, New York, 1973.

[21] Paterson, M.S. Efficient iterations for algebraic numbers. In *Complexity of Computer Computations* (edited by R.E. Miller and J.W. Thatcher). Plenum Press, New York, 1972, 41–52.

[22] Rissanen, J. On optimum root-finding algorithms. *J. Math. Anal. Applics.* **36**, 220–225, 1971.

[23] Schönhage, A. and Strassen, V. Schnelle Multiplikation grosser Zahlen. *Computing* **7**, 281–292, 1971.
[24] Schultz, M.H. The computational complexity of elliptic partial differential equations. In *Complexity of Computer Computations* (edited by R.E. Miller and J.W. Thatcher). Plenum Press, New York, 1972, 73–83.
[25] Shanks, D. and Wrench, J.W. Calculation of π to 100,000 decimals. *Math. Comp.* **16** 76–99, 1962.
[26] Shanks, D. and Wrench, J.W. Calculation of e to 100,000 decimals. *Math. Comp.* **23** 679–680, 1969.
[27] Traub, J.F. *Iterative Methods for the Solution of Equations.* Prentice-Hall, Englewood Cliffs, New Jersey, 1964.
[28] Traub, J.F. Computational complexity of iterative processes. *SIAM J. Computing* **1**, 167–179, 1972.
[29] Traub, J.F. Numerical mathematics and computer science. *Comm. ACM* **15**, 537–541, 1972.
[30] Traub, J.F. Optimal iterative processes: theorems and conjectures. In *Information Processing 71*. North-Holland, Amsterdam, 1972, 1273–1277.
[31] Traub, J.F. Theory of optimal algorithms. In *Software for Numerical Mathematics* (edited by D.J. Evans). Academic Press, 1974.
[32] Traub, J.F. An introduction to some current research in numerical computational complexity. Tech. Report, Department of Computer Science, Carnegie-Mellon University, 1973.
[33] Winograd, S. On the time required to perform addition. *J. ACM* **12**, 277–285, 1965.
[34] Winograd, S. On the time required to perform multiplication. *J. ACM* **14**, 793–802, 1967.
[35] Wozniakowski, H. Generalized information and maximal order of iteration for operator equations. *SIAM J. Numer. Anal.* **12**, 121–135, 1975.
[36] Wozniakowski, H. Maximal stationary iterative methods for the solution of operator equations. *SIAM J. Numer. Anal.* **11**, 934–949, 1974.
[37] Brent, R.P. Fast multiple-precision evaluation of elementary functions. *J. ACM* **23**, 242–251, 1976.
[38] Brent, R.P. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. In *Analytic Computational Complexity* (edited by J.F. Traub). Academic Press, New York, 1975, 59–73.

Postscript (September 1999)

Historical Notes

This paper was retyped (with minor corrections) in LaTeX during August 1999. It is available electronically in compressed postscript format from `ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Richard.Brent/rpb032.ps.gz`

The related paper Brent [38] is available electronically in compressed postscript format from `ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Richard.Brent/rpb028.ps.gz`

The paper Kung [14] appeared in *SIAM J. Numer. Anal.* **12** (1975), 89–96.

Sharper Results

Some of the constants given in Table 7.1 can be improved, e.g. C_{DM}, C_{RM}, C_{DS}, C_{RS}. One source of improvement is given in a report by Karp and Markstein[2].

For example, consider C_{DM}. We want to compute an n-bit approximation to b/a. If $x_i \to 1/a$ as in (7.4) and we define $y_i = bx_i$, then $y_i \to b/a$. Also, if x_i satisfies the recurrence (7.4), then y_i satisfies

$$y_{i+1} = y_i - x_i(ay_i - b) \ . \tag{7.4$'$}$$

Note that (7.4') is self-correcting because of the computation of the residual $ay_i - b$. Suppose x_i has (relative) precision $n/2$. If we approximate $y_i = bx_i$ using an $\frac{n}{2}$-bit multiplication, compute the residual $ay_i - b$ using an n-bit multiplication, then its product with x_i using an $\frac{n}{2}$-bit multiplication, we can apply (7.4') to obtain y_{i+1} with relative precision n. Assuming x_i is obtained in time $\sim 3M(n/2) \sim \frac{3}{2}M(n)$ (see (7.5)), the time to obtain y_{i+1} is $\sim \frac{7}{2}M(n)$, i.e. $C_{DM} \le 3.5$, which is sharper than the bound $C_{DM} \le 4.0$ given in Table 7.1.

Similarly, we can obtain $C_{RM} \le 4.25$, which is sharper than the bound $C_{RM} \le 5.5$ given in Table 7.1. If $x_i \to a^{-1/2}$ and $y_i = ax_i \to \sqrt{a}$, we compute a precision $n/2$ approximation x_i in time $\sim \frac{9}{2}M(n/2)$ as in Section 7, then apply a final second-order iteration for

$$y_{i+1} = y_i - x_i(y_i^2 - a)/2 \tag{7.9$'$}$$

(derived by multiplying (7.9) by a and using (7.7)) to obtain a precision n approximation y_{i+1} to $\sqrt{a}$.

As a corollary, the time required for an arithmetic-geometric mean iteration [37,38] is reduced from $\sim 6.5M(n)$ to $\sim 5.25M(n)$.

The Definition of n-bit Multiplication

Our $t_n(M)$ (see Sections 1–3) is essentially the time required to compute the most significant n bits in the product of two n-bit numbers. In Brent [38], $t_n(M)$ is written as $M(n)$. A related but subtly different function is $M^*(n)$, defined as the time required to compute the full $2n$-bit

[2] Alan H. Karp and Peter Markstein, *High Precision Division and Square Root*, HP Labs Report 93-93-42 (R.1), June 1993, Revised October 1994. Available electronically from `http://www.hpl.hp.com/techreports/93/HPL-93-42.html`

product of n-bit numbers[3]. Paul Zimmermann[4] observed that smaller constants can sometimes be obtained in row $Y = M$ of Table 7.1 if we use $M^*(n)$ instead of $M(n)$. (We denote these constants by C_{XM^*} to avoid confusion with the C_{XM} of Table 7.1.) For example, $C_{DM^*} < 3.5$ and $C_{RM^*} < 4.25$.

It is an open question whether

$$M(n) \sim M^*(n) \text{ as } n \to \infty ;$$

with the best available multiplication algorithms (those based on the FFT) this is true[5].

Final Comments

Daniel Bernstein[6] observed that the time required to compute n-bit square roots can be reduced further if the model of computation is relaxed so that redundant FFTs can be eliminated. Similar remarks apply to division, exponentiation etc (and to operations on power series).

In conclusion, 25 years after the paper was written (in 1974), improvements can still be found, and the last word is yet to be written!

[3]In Brent [37] we (confusingly) used the notation $M(n)$ for $M^*(n)$.

[4]Personal communication, 1999.

[5]Similar remarks apply if we consider computing the product of two polynomials of degree $n-1$, and ask either for the first n terms in the product or the complete product. Although the first computation is faster (by a factor of about two) if the classical order n^2 algorithms are used, it is not significantly faster if FFT-based algorithms are used.

[6]Personal communication, 1999.