ON COMPUTING RECIPROCALS OF POWER SERIES

H. T. Kung

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa.

September 1973

This research was supported in part by the National Science Foundation under Grant GJ32111 and the Office of Naval Research under Contract N00014-67-A-0314-0010, NR 044-422.

ABSTRACT

Root-finding iterations are used to compute reciprocals of power series. We show that Sieveking's algorithm is just Newton iteration applied in the field of power series. Let L_n denote the number of non-scalar multiplications needed to compute the first n+1 terms of the reciprocal. We show that

$$n+1 \le L_n \le 4n-\log_2 n.$$

We conjecture that

$$L_n = 4n - \text{lower order terms.}$$

1. INTRODUCTION

We consider the problem of computing reciprocals of power series. This problem is closely related to the problems of polynomial division, evaluation and interpolation. (For example, see Borodin (1973).) Let L_n denote the number of non-scalar multiplications needed to compute the first n+1 terms of the reciprocal of a power series. Recently Sieveking (1972) showed that

$$L_n \le 5n-2.$$

In this paper, we show:

(i) Root-finding iterations can be used in the field of power series. Sieveking's algorithm is just Newton iteration applied to the function $f(x) = x^{-1}-a$, $a \neq 0$, in the field of power series.

(ii) By modifying Sieveking's algorithm and analysis, Sieveking's bound is improved to

$$L_n \le 4n-\log_2 n.$$

(iii) We propose a new algorithm for computing reciprocals of power series, which is competitive with Sieveking's algorithm, and which is based on a third order iteration. The bound in (ii) can also be obtained by this new algorithm.

(iv) $L_n \ge n+1$ for all $n \ge 0$.

In Section 2 we define some basic notation and also prove results (i) and (ii). Results (iii) and (iv) are proven in Sections 3 and 4, respectively.

In Section 5 we give a general family of algorithms for computing reciprocals of power series. Any algorithm in the family can compute the first n+1 terms of the reciprocal of a power series in O(n) non-scalar multiplications (and can also compute them in O(n log n) arithmetic operations if Fast Fourier Transform is used for polynomial multiplication). We conjecture that Newton iteration and the third order iteration are optimal among all algorithms in the family.

2. NEWTON ITERATION

We will use notation of Sieveking (1972) and Strassen (1973). Let k be an infinite field, a_i, b_i, $i = 0,1,\ldots,\infty$, indeterminates over k, A an extension field of k, and t an indeterminate over A. Suppose that E and F are finite subsets of A and that we do computations in the field A. Let L(E mod F) denote the number of multiplications or divisions by units which are necessary to compute E starting from $k \cup F$ not counting multiplications by scalars in k. We shall prove the following theorem by using Newton iteration.

Theorem 2.1.

Suppose $A = k(a_0,a_1,\ldots)$, a_0 is a unit in A and

$$\sum_0^n a_i t^i \sum_0^n b_i t^i \equiv 1 \quad (t^{n+1}). \tag{2.1}$$

Then

$L(b_0,\ldots,b_n \bmod a_0,\ldots,a_n) \le 4n-\log_2 n$ for any $n \ge 1$.

We first use a technique of Strassen (1973) to prove the following

Lemma 2.1.

Suppose $A = k(a_0,b_0,a_1,b_1,\ldots)$ and

$$\sum_0^n a_i t^i \sum_0^m b_i t^i = \sum_0^{\ell-1} c_i t^i + \sum_\ell^{n+m} c_i t^i .$$

Then

$$L(c_\ell,\ldots,c_{n+m} \bmod a_0,\ldots,a_n,b_0,\ldots,b_m,c_0,\ldots,c_{\ell-1}) \le n+m-\ell+1$$

for any $n,m \ge 0$ and ℓ such that $n+m \ge \ell \ge 0$. ($\sum_0^{-1}$ is assumed to be zero.)

Proof of Lemma 2.1.

Let λ_j, $1 \le j \le n+m - \ell+1$, be any $n+m - \ell+1$ distinct nonzero elements in k. Observe that

$$(2.2) \quad \sum_{i=0}^{n+m-\ell} c_{\ell+i}\,\lambda_j^i = \lambda_j^{-\ell}\Big(\sum_{i=0}^{n} a_i\lambda_j^i \sum_{i=0}^{m} b_i\lambda_j^i - \sum_{0}^{\ell-1} c_i t^i\Big)$$

for $j = 1,\ldots,n+m - \ell+1$, and $\det(\lambda_j^i) \ne 0$. Hence $c_{\ell+i}$, $0 \le i \le n+m-\ell$, can be obtained by solving the linear system (2.2). Therefore,

$$L(c_\ell,\ldots,c_{n+m} \bmod a_0,\ldots,a_n,b_0,\ldots,b_m,c_0,\ldots,c_{\ell-1})$$

$$\le L\Big(\sum_{i=0}^{n} a_i\lambda_j^i \sum_{i=0}^{m} b_i\lambda_j^i,\ j = 1,\ldots,n+m - \ell+1, \bmod a_0,\ldots,a_n,b_0,\ldots,b_m\Big)$$

$$= n+m - \ell+1. \qquad \blacksquare$$

Proof of Theorem 2.1.

Denote $\sum_0^\infty a_i t^i$ by a and $\sum_0^\infty b_i t^i$ by b. Suppose that (2.1) holds for all n. Then b is the reciprocal of a with respect to the field A(t).

Define the function $f\colon A(t) - \{0\} \to A(t)$ by $f(x) = x^{-1}-a$. Thus b just the root of f. Applying Newton iteration to f, we obtain the iterat

function

$$(2.3) \quad \varphi(x) = x - f'(x)^{-1}f(x) = 2x-ax^2.$$

(In this paper, derivatives of f are defined by purely algebraic methods without employing any limit concept. For example, see van der Waerden (1953, §65).) It follows from (2.3) that

$$(2.4) \quad \varphi(x)-b = a(x-b)^2.$$

For notational simplicity, let L_n denote $L(b_0,\dots,b_n \bmod a_0,\dots,a_n)$. To prove the theorem it suffices to show that

(2.5) $L_{2n+1} \le L_n + 4n + 2 \quad \text{for } n \ge 0,$

(2.6) $L_{2n} \le L_n + 4n - 1 \quad \text{for } n \ge 1.$

Suppose that $b_0,\dots,b_n$ have already been computed. In (2.4) let x be taken to be $\sum_0^n b_i t^i$. Then

$$\varphi\left(\sum_0^n b_i t^i\right) - b \equiv 0 \qquad (t^{2n+2}),$$

or

(2.7) $$2 \sum_0^n b_i t^i - \sum_0^{2n+1} a_i t^i \left(\sum_0^n b_i t^i\right)^2 \equiv \sum_0^{2n+1} b_i t^i \qquad (t^{2n+2}).$$

Note that

$$\sum_0^n a_i t^i \sum_0^n b_i t^i \equiv 1 \quad (t^{n+1}).$$

We define $e_{n+1},\dots,e_{3n+1}$ by

(2.8) $$\sum_0^{2n+1} a_i t^i \sum_0^n b_i t^i = 1 + \sum_{n+1}^{3n+1} e_i t^i.$$

Then by (2.7) and (2.8),

$$-\sum_0^n e_{n+1+i} t^i \sum_0^n b_i t^i \equiv \sum_0^n b_{n+1+i} t^i \quad (t^{n+1}).$$

Therefore

(2.9) $$L_{2n+1} \le L_n + L(e_{n+1},\dots,e_{2n+1} \bmod a_0,\dots,a_{2n+1},b_0,\dots,b_n)$$
$$+ L(b_{n+1},\dots,b_{2n+1} \bmod e_{n+1},\dots,e_{2n+1},b_0,\dots,b_n).$$

By Lemma 2.1,

$$L_{2n+1} \le L_n + (2n+1) + (2n+1) = L_n + 4n+2.$$

We have shown (2.5). From (2.7) we also have

$$(2.10)\quad 2\sum_0^n b_i t^i - \sum_0^{2n} a_i t^i \left(\sum_0^n b_i t^i\right)^2 \equiv \sum_0^{2n} b_i t^i \quad (t^{2n+1}).$$

(2.6) follows in the same as (2.5) by starting with (2.10) instead of (2.7). ■

One can easily check that the algorithm proposed by Sieveking (1972) is just the Newton iteration stated above. However, because of (2.8), Lemma 2.1, and careful estimation of L_n from (2.5) and (2.6) we are able to obtain

$$L_n \le 4n - \log_2 n \quad \text{for } n \ge 1$$

rather than

$$L_n \le 5n-2 \qquad \text{for } n \ge 1$$

which is obtained by Sieveking (1972). One should also note that the idea of using Newton iteration to compute reciprocals has been known for a long
For example, Newton iteration is used to compute matrix inverses by Schulz (1933), to compute reciprocals of real numbers by Rabinowitz (196 and to compute integer reciprocals by S. A. Cook (see Knuth (1969, §4.33
In this section, we have shown that Newton iteration can be used successfully in the field of power series, and hence can compute reciprocals of power series. In fact, any root-finding iteration (Traub (1964)) can be used for the problem of computing reciprocals (see Section 5). Newton iteration is a second order iteration. In the next section we propose a new algorithm for computing reciprocals of power series, which is based on a third order iteration, and which is competitive with Sieveking's algorithm.

3. ANOTHER ALGORITHM FOR RECIPROCALS

We use notation defined in the previous section. Applying the third order iteration (Traub (1964)),

$$\bar{\varphi}(x) = x - f'(x)^{-1}f(x) - \frac{1}{2}[f'(x)^{-1}]^3 f''(x)f(x)^2,$$

to the function $f(x) = x^{-1}-a$, we get

(3.1) $\bar{\varphi}(x) = x(3-3ax + (ax)^2)$.

It is easy to show that

(3.2) $\bar{\varphi}(x) - b = a^2(x-b)^3$.

We shall now use $\bar{\varphi}$ to prove Theorem 2.1 for $n \geq 3$. Let L_n denote $L(b_0,\ldots,b_n \bmod a_0,\ldots,a_n)$ as before. Note that $L_1 \leq 3$ and $L_2 \leq 6$. It is not difficult to check that it suffices to prove that for $n \geq 1$,

(3.3) $L_{3n+2} \leq L_n + 8n + 5$,

(3.4) $L_{3n+1} \leq L_n + 8n + 1$,

(3.5) $L_{3n} \leq L_n + 8n - 3$.

Suppose that $b_0,\ldots,b_n$ have already been computed. In (3.2) let x be taken to be $\sum_0^n b_i t^i$ then

$$\bar{\varphi}(\sum_0^n b_i t^i) - b \equiv 0 \quad (t^{3n+3}),$$

or

(3.6) $(\sum_0^n b_i t^i)[3 - 3 \sum_0^{3n+2} a_i t^i \sum_0^n b_i t^i + (\sum_0^{3n+2} a_i t^i \sum_0^n b_i t^i)^2] \equiv \sum_0^{3n+2} b_i t^i \quad (t^{3n+3})$.

Note that

$$\sum_0^n a_i t^i \sum_0^n b_i t^i \equiv 1 \quad (t^{n+1}).$$

We define $e_{n+1},\ldots,e_{4n+2}$ by

$$\sum_0^{3n+2} a_i t^i \sum_0^n b_i t^i = 1 + \sum_{n+1}^{4n+2} e_i t^i.$$

Moreover, define $d_0,\ldots,d_n$ by

$$\Big(\sum_0^n e_{n+1+i} t^i\Big)^2 \equiv \sum_0^n d_i t^i \quad (t^{n+1}).$$

Then

$$(3.7)\quad 3 - 3 \sum_0^{3n+2} a_i t^i \sum_0^n b_i t^i + \Big(\sum_0^{3n+2} a_i t^i \sum_0^n b_i t^i\Big)^2$$

$$\equiv 3t^{n+1} \sum_0^{2n+1} e_{n+1+i} t^i + 1 + 2\,t^{n+1} \sum_0^{2n+1} e_{n+1+i} t^i + t^{2n+2} \sum_0^n d_i t^i \quad (t^{3n+3})$$

$$\overset{\text{def.}}{\equiv} 1 + t^{n+1} \sum_0^{2n+1} f_i t^i \quad (t^{3n+3}).$$

Define $g_0,\ldots,g_{3n+1}$ by

$$(3.8)\quad \sum_0^n b_i t^i \sum_0^{2n+1} f_i t^i = \sum_0^{3n+1} g_i t^i.$$

Then by (3.6), (3.7) and (3.8),

$$\sum_0^{2n+1} g_i t^i \equiv \sum_0^{2n+1} b_{n+1+i} t^i \quad (t^{2n+2}).$$

Therefore

$$L_{3n+2} = L_n + L(e_{n+1},\ldots,e_{3n+2} \bmod a_0,\ldots,a_{3n+2},b_0,\ldots,b_n)$$

$$+ L(d_0,\ldots,d_n \bmod e_{n+1},\ldots,e_{2n+1})$$

$$+ L(g_0,\ldots,g_{2n+1} \bmod e_{n+1},\ldots,e_{3n+2},d_0,\ldots,d_n,b_0,\ldots,b_n).$$

By Lemma 2.1,

$$L_{3n+2} \le L_n + (3n+2) + (2n+1) + (3n+2)$$
$$= L_n + 8n + 5.$$

We have shown (3.3). From (3.6) we also have

$$(3.9)\quad \left(\sum_0^n b_it^i\right)\left[3 - 3\sum_0^{3n+1} a_it^i \sum_0^n b_it^i + \left(\sum_0^{3n+1} a_it^i \sum_0^n b_it^i\right)^2\right] \equiv \sum_0^{3n+1} b_it^i \quad (t^{3n+2}),$$

$$(3.10)\quad \left(\sum_0^n b_it^i\right)\left[3 - 3\sum_0^{3n} a_it^i \sum_0^n b_it^i + \left(\sum_0^{3n} a_it^i \sum_0^n b_it^i\right)^2\right] \equiv \sum_0^{3n} b_it^i \quad (t^{3n+1}).$$

(3.4) and (3.5) follow in the same way as (3.3) by starting with (3.9) and (3.10), respectively instead of (3.6).

4. A LOWER BOUND

Under the hypotheses of Theorem 2.1, we show that

(4.1) $L(b_0,\dots,b_n \bmod a_0,\dots,a_n) \geq n+1$.

Suppose that $a_i = 0$, $i = 2,\dots,n$. Then it is clear that $b_i = (a_0^{-1}a_1)^i a_0^{-1}$, $i = 0,\dots,n$. (4.1) follows from the following

Lemma 4.1.

Suppose $A = k(a_0,a_1)$, a_0 is unit in A and

$$b_i = (a_0^{-1}a_1)^i \, a_0^{-1}, \quad i = 0,\dots,n.$$

Then

$$L(b_0,\dots,b_n \bmod a_0,a_1) \geq n+1.$$

Proof.

Consider an arbitrary algorithm which requires m non-scalar multiplications or divisions. Let $R_1,\dots,R_m$ denote the results of these multiplications or divisions. Then there exist $p_{i,j} \in k$, $q_i \in \{k_0a_0+k_1a_1 \mid k_0,k_1 \in k\} \cup k$, $i = 0,\dots,n$, $j = 1,\dots,m$ such that

$$\sum_{j=1}^{m} p_{i,j}R_j + q_i = b_i, \quad i = 0,\dots,n.$$

Suppose that $m < n+1$. Then there exist $r_i \in k$, $i = 0,\dots,n$, such that $r_i \neq 0$ for some i and

$$\sum_0^n r_i(b_i - q_i) = 0,$$

or

$$\sum_0^n r_i a_1^i a_0^{n-i} = \left(\sum_0^n r_i q_i\right) a_0^{n+1}.$$

This clearly implies that $r_i = 0$ for all $i = 0,\dots,n$ which is a contradiction. ■

5. A FAMILY OF ALGORITHMS FOR RECIPROCALS AND A CONJECTURE

Suppose $a = \sum_0^\infty a_i t^i$ and $b = \sum_0^\infty b_i t^i$ and that (2.1) is satisfied for all n. Let $A = k(a_0, a_1, \ldots)$. For any nonnegative integers d, $\ell_0, \ell_1, \ldots, \ell_d$ (not all zero) we define an algorithm which generates the sequence of iterates $\{x^{(k)}\}$ in A(t) by

$$\begin{aligned} x^{(k+1)} &= x^{(k)}(1-ax^{(k-1)})^{\ell_1}\cdots(1-ax^{(k-d)})^{\ell_d}\sum_{j=0}^{\ell_1-1}(1-ax^{(k)})^j \\ &\quad + x^{(k-1)}(1-ax^{(k-2)})^{\ell_2}\cdots(1-ax^{(k-d)})^{\ell_d}\sum_{j=0}^{\ell_1-1}(1-ax^{(k-1)})^j \\ &\quad + \cdots \\ &\quad\ \vdots \\ &\quad + x^{(k-d)}\sum_{j=0}^{\ell_d-1}(1-ax^{(k-d)})^j. \end{aligned} \tag{5.1}$$

Then it can be shown that

$$x^{(k+1)} = b - b(1-ax^{(k)})^{\ell_0}(1-ax^{(k-1)})^{\ell_1}\cdots(1-ax^{(k-d)})^{\ell_d}. \tag{5.2}$$

For all k define c_k to be the greatest integer such that

$$x^{(k)} \equiv b \quad (t^{c_k}).$$

Then from (5.2) it follows that

$$x^{(k+1)} \equiv b \qquad (t^{\ell_0 c_k + \cdots + \ell_d c_{k-d}}).$$

Hence

$$c_{k+1} \geq \sum_{i=0}^{d} \ell_i c_{k-i}. \tag{5.3}$$

Using (5.3) we can estimate the number of iteration steps necessary to compute $b_0, \ldots, b_n$ from $a_0, \ldots, a_n$ for any given n. Note that in computing $x^{(k+1)}$ by (5.1) we should use $\sum_0^{c_{k+1}} a_i t^i$ for $a(= \sum_0^\infty a_i t^i)$.

Example 5.1.

(i) if $d = 0$ and $\ell_0 = 2$ we have

$$x^{(k+1)} = x^{(k)}[1 + (1-ax^{(k)})] = 2x^{(k)} - ax^{(k)}x^{(k)}.$$

This is the Newton iteration (see (2.5)).

(ii) If $d = 0$ and $\ell_0 = 3$ we have

$$\begin{aligned} x^{(k+1)} &= x^{(k)}[1 + (1-ax^{(k)}) + (1-ax^{(k)})^2] \\ &= x^{(k)}[3 - 3ax^{(k)} + (ax^{(k)})^2]. \end{aligned}$$

This is the third order iteration $\bar{\varphi}$ in (3.1).

(iii) If $d = 1$ and $\ell_0 = \ell_1 = 1$ we have

$$\begin{aligned} x^{(k+1)} &= x^{(k)}(1-ax^{(k-1)}) + x^{(k-1)} \\ &= -ax^{(k-1)}x^{(k)} + x^{(k-1)} + x^{(k)}. \end{aligned}$$

One can check that this is the secant iteration applied to $f(x) = x^{-1}$ -

In fact, (5.1) represents the algorithm which is obtained by a general Hermite interpolatory iteration (Traub (1964)) applied to the function $f(x) = x^{-1} - a$. A special case of (5.1) (i.e., $d = 0$) was pointed out before by Rabinowitz (1961) for computing reciprocals of numbers. By the same techniques used in Sections 2 and 3 one could show that $L(b_0,\ldots,b_n \bmod a_0,\ldots,a_n)$ is bounded by a linear function of n by using any algorithm defined by (5.1). However, we believe that Newton iteration and the third order iteration are optimal among all algorithms defined by (5.1). More precisely, we state the following

Conjecture.

$L(b_0,\ldots,b_n \bmod a_0,\ldots,a_n) = 4n$ - lower order terms, for large n.

If we use the Fast Fourier Transform for polynomial multiplication, then it can be easily shown by techniques similar to those used in this paper that any algorithm defined by (5.1) is able to compute the first $n+1$ terms of the reciprocal of a power series in $O(n \log n)$ arithmetic operations.

ACKNOWLEDGMENT

I would like to thank Professor J. F. Traub for his comments on this paper.

References

Borodin, A.(1973), "On the Number of Arithmetics Required to Computer Certain Functions - Circa May 1973." Appears in Complexity of Sequentiāl and Parallel Numerical Algorithms, edited by J. F. Traub, Academic Press, New York.

Knuth, D. E. (1969), The Art of Computer Programming, Vol. II, Seminumerical Algorithms, Addison-Wesley, Reading, Mass.

Rabinowitz, P. (1961), "Multiple-Precision Division," Comm. ACM 4, p. 98.

Schulz, G. (1933), "Iterative Berechnung der reziproken Matrix," Z. Angew. Math. and Mech., 13, pp. 57-59.

Sieveking, M. (1972), "An Algorithm for Division of Power Series," Computing 10, pp. 153-156.

Strassen, V. (1973), "Vermeidung von Divisionen." To appear in J.Reine Angew.Math.

Traub, J. F. (1964), Iterative Methods for the Solution of Equations, Prentice-Hall.

van der Waerden, B. L. (1953), Modern Algebra, Vol. 1, Frederick Ungar Publishing Co.