Numer. Math. 22, 341—348 (1974)

On Computing Reciprocals of Power Series*

H. T. Kung

Received October 10, 1973

Abstract. It is shown that root-finding iterations can be used in the field of power series. As a consequence, we obtain a class of new algorithms for computing reciprocals of power series. In particular, we show that the recent Sieveking algorithm for computing reciprocals is just Newton iteration. Moreover, if L_n is the number of non scalar multiplications needed to compute the first $n+1$ terms of the reciprocal of a power series, we show that

$$n+1 \leqq L_n \leqq 4n - \log_2 n$$

and 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 nonscalar multiplications needed to compute the first $n+1$ terms of the reciprocal of a power series. Recently Sieveking (1972) showed that

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

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

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

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

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

The idea of using Newton iteration to compute reciprocals has been known for a long time. Newton iteration was used to compute reciprocals of real numbers even in early desk or manual computations. See, for instance, Böhm (1955). Moreover, Newton iteration has been used to compute matrix inverses by Schulz

* This research was supported in part by the National Science Foundation under Grant GJ 32111 and the Office of Naval Research under Contract N00014-67-A-0314-0010, NR044-422.

(1933) and to compute integer reciprocals by S. A. Cook (see Knuth (1969, § 4.33)).

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 the coefficients of the power series are complex numbers and if the 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, $a_i, b_i, i = 0, 1, \ldots$, 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 $L(E \bmod F)$ 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. In this paper, A will be taken to be one of $k(a_0, a_1, \ldots)$, $k(a_0, b_0, a_1, b_1, \ldots)$ or $k(a_0)$. One should note that all algorithms presented in this paper do not require the commutativity relation among $a_0, b_0, a_1, b_1, \ldots$. Therefore, a_i, b_i could be, for example, matrices. 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*

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

Then

$$L(b_0, \ldots, b_n \bmod a_0, \ldots, a_n) \leqq 4n - \log_2 n \quad \textit{for any} \quad n \geqq 1.$$

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*

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

Then

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

for any, $n, m \geqq 0$ *such that* $n + m \geqq l$. $\left(\sum_0^{-1} \textit{is assumed to be zero.}\right)$

Proof of Lemma 2.1. Let $\lambda_j, 1 \leqq j \leqq n+m-l+1$, be any $n+m-l+1$ distinct nonzero elements in k. Observe that

$$\sum_{i=0}^{n+m-l} c_{l+i} \lambda_j^i = \lambda_j^{-l} \left(\sum_{i=0}^n a_i \lambda_j^i \sum_{i=0}^m b_i \lambda_j^i - \sum_0^{l-1} c_i \lambda_j^i \right) \tag{2.2}$$

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

$$\begin{aligned} &L(c_l, \ldots, c_{n+m} \bmod a_0, \ldots, a_n, b_0, \ldots, b_m, c_0, \ldots, c_{l-1}) \\ &\leqq L\left(\sum_{i=0}^{n} a_i \lambda_j^i \sum_{i=0}^{m} b_i \lambda_j^i, j=1, \ldots, n+m-l+1, \bmod a_o, \ldots, a_n, b_0, \ldots, b_m\right) \\ &= n+m-l+1. \quad \blacksquare \end{aligned}$$

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

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

(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

$$\varphi(x) - b = a(x-b)^2. \tag{2.4}$$

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

$$L_{2n+1} \leqq L_n + 4n + 2 \quad \text{for} \quad n \geqq 0, \tag{2.5}$$

$$L_{2n} \leqq L_n + 4n - 1 \quad \text{for} \quad n \geqq 1. \tag{2.6}$$

Suppose that $b_0, \ldots, b_n$ 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

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

Note that

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

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

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

Then by (2.7) and (2.8),

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

Therefore, by the transitivity principle of Strassen,

$$\begin{aligned} L_{2n+1} \leqq L_n &+ L(e_{n+1}, \ldots, e_{2n+1} \bmod a_0, \ldots, a_{2n+1}, b_0, \ldots, b_n) \\ &+ L(b_{n+1}, \ldots, b_{2n+1} \bmod e_{n+1}, \ldots, e_{2n+1}, b_0, \ldots, b_n). \end{aligned} \tag{2.9}$$

By Lemma 2.1,

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

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

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

In the same way (2.6) follows 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

$$L_n \leqq 4n - \log_2 n \quad \text{for} \quad n \geqq 1$$

rather than

$$L_n \leqq 5n - 2 \quad \text{for} \quad n \geqq 1$$

which is obtained by Sieveking (1972). 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)),

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

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

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

It is easy to show that

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

We shall now use $\bar{\varphi}$ to prove Theorem 2.1 for $n \geqq 3$. Let L_n denote

$$L(b_0, \ldots, b_n \bmod a_0, \ldots, a_n)$$

as before. Note that $L_1 \leqq 3$ and $L_2 \leqq 6$. It is not difficult to check that it suffices to prove that for $n \geqq 1$,

$$L_{3n+2} \leqq L_n + 8n + 5, \tag{3.3}$$

$$L_{3n+1} \leqq L_n + 8n + 1, \tag{3.4}$$

$$L_{3n} \leqq L_n + 8n - 3. \tag{3.5}$$

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

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

or

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

Note that

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

We define $e_{n+1}, \ldots, e_{3n+2}$ by

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

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

$$\left(\sum_0^n e_{n+1+i} t^i\right)^2 \equiv \sum_0^n d_i t^i \qquad (t^{n+1}).$$

Then define $f_0, \ldots, f_{2n+1}$ by

$$\begin{aligned} 3 - 3\sum_0^{3n+2} a_i t^i \sum_0^n b_i t^i &+ \left(\sum_0^{3n+2} a_i t^i \sum_0^n b_i t^i\right)^2 \\ &\equiv 3t^{n+1}\sum_0^{2n+1} e_{n+1+i} t^i + 1 + 2t^{n+1}\sum_0^{2n+1} e_{n+1+i} t^i + t^{2n+2}\sum_0^n d_i t^i \qquad (t^{3n+3}) \\ &\equiv 1 + t^{n+1}\sum_0^{2n+1} f_i t^i \qquad (t^{3n+3}). \end{aligned} \tag{3.7}$$

Define $g_0, \ldots, g_{2n+1}$ by

$$\sum_0^n b_i t^i \sum_0^{2n+1} f_i t^i \equiv \sum_0^{2n+1} g_i t^i \qquad (t^{2n+2}). \tag{3.8}$$

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

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

Therefore

$$\begin{aligned} L_{3n+2} = L_n &+ L(e_{n+1}, \ldots, e_{3n+2} \bmod a_0, \ldots, a_{3n+2}, b_0, \ldots, b_n) \\ &+ L(d_0, \ldots, d_n \bmod e_{n+1}, \ldots, e_{2n+1}) \\ &+ L(g_0, \ldots, g_{2n+1} \bmod e_{n+1}, \ldots, e_{3n+2}, d_0, \ldots, d_n, b_0, \ldots, b_n). \end{aligned}$$

By Lemma 2.1.

$$\begin{aligned} L_{3n+2} &\leq L_n + (3n+2) + (2n+1) + (3n+2) \\ &= L_n + 8n + 5. \end{aligned}$$

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

$$\left(\sum_0^n b_i t^i\right)\left[3 - 3\sum_0^{3n+1} a_i t^i \sum_0^n b_i t^i + \left(\sum_0^{3n+1} a_i t^i \sum_0^n b_i t^i\right)^2\right] \equiv \sum_0^{3n+1} b_i t^i \qquad (t^{3n+2}), \tag{3.9}$$

$$\left(\sum_0^n b_i t^i\right)\left[3 - 3\sum_0^{3n} a_i t^i \sum_0^n b_i t^i + \left(\sum_0^{3n} a_i t^i \sum_0^n b_i t^i\right)^2\right] \equiv \sum_0^{3n} b_i t^i \qquad (t^{3n+1}). \tag{3.10}$$

In the same way (3.4) and (3.5) follow 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

$$L(b_0, \ldots, b_n \bmod a_0, \ldots, a_n) \geqq n+1. \tag{4.1}$$

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

Lemma 4.1. *Suppose $A = k(a_0)$, a_0 is unit in A and*

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

Then

$$L(b_0, \ldots, b_n \bmod a_0) \geqq n+1.$$

Proof. Consider an arbitrary algorithm which requires m non scalar multiplications or divisions. Let $R_1, \ldots, R_m$ denote the results of these multiplications or divisions. Then there exist $p_{i,j} \in k$, $q_i \in \{k_0 a_0 + k_1 | k_0, k_1 \in k\}$ $i = 0, \ldots, n$, $j = 1, \ldots, m$ such that

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

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

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

or

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

This clearly implies that $r_i = 0$ for all $i = 0, \ldots, n$ 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 $d, l_0, l_1, \ldots, l_d$ (not all zero) we define an algorithm which generates the sequence of iterates $\{x^{(h)}\}$ in $A(t)$ by

$$\begin{aligned} x^{(h+1)} = {} & x^{(h)} (1 - a x^{(h-1)})^{l_1} \ldots (1 - a x^{(h-d)})^{l_d} \sum_{j=0}^{l_0 - 1} (1 - a x^{(h)})^j \\ & + x^{(h-1)} (1 - a x^{(h-2)})^{l_2} \ldots (1 - a x^{(h-d)})^{l_d} \sum_{j=0}^{l_1 - 1} (1 - a x^{(h-1)})^j \\ & + \cdots \\ & \vdots \\ & + x^{(h-d)} \sum_{j=0}^{l_d - 1} (1 - a x^{(h-d)})^j. \end{aligned} \tag{5.1}$$

Then it can be shown that

$$x^{(h+1)} = b - b(1 - a x^{(h)})^{l_0} (1 - a x^{(h-1)})^{l_1} \ldots (1 - a x^{(h-d)})^{l_d}. \tag{5.2}$$

For all h define c_h to be the greatest integer such that

$$x^{(h)} \equiv b \qquad (t^{c_h}).$$

Then from (5.2) it follows that

$$x^{(h+1)} \equiv b \qquad (t^{l_0 c_h + \cdots + l_d c_{h-d}}).$$

Hence

$$c_{h+1} \geqq \sum_{i=0}^{d} l_i c_{h-i}. \tag{5.3}$$

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 $x^{(h+1)}$ by (5.1) we should use $\sum_0^{c_{h+1}} a_i t^i$ for $a \left(= \sum_0^{\infty} a_i t^i\right)$.

Example 5.1. (i) if $d=0$ and $l_0=2$ we have

$$x^{(h+1)} = x^{(h)}[1+(1-a x^{(h)})] = 2x^{(h)} - a x^{(h)} x^{(h)}.$$

This is the Newton iteration (see (2.3)).

(ii) If $d=0$ and $l_0=3$ we have

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

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

(iii) If $d=1$ and $l_0=l_1=1$ we have

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

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

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 the coefficients of a power series are complex numbers and 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

Böhm, C.: Perfezionamento di un processo iterativo atto alla divisione automatica. La Ricerca Scientifica **25**, 2077–2080 (1955)

Borodin, A.: On the number of arithmetics required to computer certain functions — Circa May 1973, in: Complexity of sequential and parallel numerical algorithms, edited by J. F. Traub, p. 149–180. New York: Academic Press 1973

Knuth, D. E.: The art of computer programming, vol. II, Seminumerical algorithms. Reading, Mass.: Addison-Wesley 1969

Rabinowitz, P.: Multiple-precision division. Comm. ACM **4**, 98 (1961)

Schulz, G.: Iterative Berechnung der reziproken Matrix. Z. Angew. Math. and Mech. **13**, 57–59 (1933)

Sieveking, M.: An algorithm for division of power series. Computing **10**, 153–156 (1972)

Strassen, V.: Vermeidung von Divisionen. J. Reine Angew. Math. **264**, 184–202 (1973)

Traub, J. F.: Iterative methods for the solution of equations. Englewood Cliffs, N. J.: Prentice-Hall 1964

van der Waerden, B. L.: Modern algebra, vol. 1. New York: Frederick Ungar Publishing Co. 1953

H. T. Kung
Dept. of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa. 15213/U.S.A.