Computers Math. Applic. Vol. 17, No. 10, pp. 1423–1427, 1989
Printed in Great Britain. All rights reserved

0097-4943/89 $3.00 + 0.00
Copyright © 1989 Pergamon Press plc

FAST COMPUTATION OF THE Nth ROOT

S.-G. CHEN† and P. Y. HSIEH
Department of Electrical and Computer Engineering, State University of New York at Buffalo, 201 Bell Hall, Amherst, NY 14260, U.S.A.

(*Received* 23 *May* 1988)

Abstract—A new class of iterative methods for computing a differentiable function is proposed, which is based on Páde approximation to Taylor's series of the function. It leads to a faster algorithm than Newton's method for $x^{1/N}$ and a different interpretation of Newton's method. This algorithm uses 3rd degree approximation of continued fraction expansion (CFE) to Taylor's series for $x^{1/N}$, with adaptive expansion point for every iteration. Its major computational cost is $O[(2\log_2 N)(\log_4 M)]$ multiplications asymptotically, M is the number of precision bits desired, as opposed to existing bound of $O[(2\log_2 N)(\log_2 M)]$ for Newton's method. A new periodic CFE for special case $x^{1/2}$ is obtained which is simple and free from error propagation. New algorithm is not applied to the square root case for N is too small to be practical; instead, the 3rd degree approximation is used as the starting value for Newton's method which has high precision and low computational cost.

1. INTRODUCTION

Newton's method [1] is a fast way to compute Nth root of a number x for its quadratic convergence rate as shown in conditions (1) and (2):

$$r_{i+1} = r_i + \frac{1}{N}\left(\frac{x}{r_i^{N-1}} - r_i\right) \tag{1}$$

and

$$\epsilon_{i+1} = \frac{r_{i+1} - x^{1/N}}{r_{i+1}} \approx \frac{N-1}{2x^{1/N}} \frac{r_i^2}{r_{i+1}} \left(\frac{r_i - x^{1/N}}{r_i}\right)^2. \tag{2}$$

For a normalized floating-point number $y = x2^p$

$$y^{1/N} = x^{1/N} 2^{p/N} = [x2^{-N+(p \bmod N)}]^{1/N} 2^{\lceil p/N \rceil},$$

where $1 > x \geq 1/2$ and p is the exponent. The value inside the square bracket can be computed from equation (1). The relative error ϵ_{i+1} is approximately proportional to ϵ_i^2. Despite of its convergence rate, every iteration for Newton's method needs $O(2\log_2 N)$ multiplications and one division, regardless of additional operations. The adjustments of exponent and mantissa parts has another disadvantage which calls for a wider range and more accurate starting approximations for Newton's method, may incur additional computational penalty and introduce error propagation when N is large.

Another way to compute $y^{1/N}$ using equation (1) without the mentioned problems is avoiding the shifting of mantissa, namely, $y^{1/N} = x^{1/N} 2^{p/N}$. This involves computation of $x^{1/N}$ which is confined to the fixed range of [1/2, 1), and $2^{p/N}$ which is equivalent to the evaluation of an exponential function. The penalty from this exponential function may be offset by the elimination of complicated starting approximations [2]. For the following discussions, the latter approach is adopted; we will only concern with the calculation of $x^{1/N}$. $x^{1/N}$ can be also be computed from Newton's iteration for $x^{-1/N}$ [3] as in equation (3) without division except one final inversion and initial inversions for $1/N$. Still, equation (3) does not have the advantage over equation (1), because multiplication and division have almost the same speed in most of the CPUs. Both equations (1) and (3) require

†Present address: Microelectronics and Information Science and Technology Research Center, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 300, Republic of China.

$O(2\log_2 N)$ multiplications asymptotically. Total of $O[(2\log_2 N)(\log_2 M)]$ multiplications are needed for result with M-bit precision.

$$r_{i+1} = r_i + r_i(1 - xr_i^N)/N. \tag{3}$$

A class of methods for computing differentiable function $f(x)$ which is based on Páde approximation to Taylor's series of the function is proposed in Section 2. These methods include algorithms with quadratic, 3rd and 4th degree convergence rates for evaluating $f(x)$. Using this approach, the quadratic algorithm derived turns out to be the same as Newton's method, but with different interpretation. The 4th degree algorithm for $x^{1/N}$ will be discussed thoroughly in Section 3, which is the fastest of all the algorithms derived in terms of multiplication steps needed. This algorithm uses the 3rd degree Páde approximation of Stieltjés-type [4] to Taylor's series of $x^{1/N}$ with adaptive expansion points. The algorithm requires precomputations of several constants, and $O(2\log_2 N)$ multiplications for an iteration which is the same as that for equation (1) but converges two times faster than Newton's method. In Section 4, a special case $x^{1/2}$ is considered. Because $N = 2$ is too small that the new algorithm is impractical to be applied to. Instead, a new identity [5] for $x^{1/2}$ in terms of continued fraction expansion (CFE) is given which is used as the starting approximation for Newton's method. Section 5 is the conclusion.

2. A NEW CLASS OF ITERATIVE METHODS FOR COMPUTING $f(x)$

Given a differentiable function $f(x)$, its Taylor's series and Páde approximation

$$f(x) = f(x_0) + f^{(1)}(x_0)(x - x_0) + \frac{f^{(2)}}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}}{n!}(x - x_0)^n + \cdots \approx \frac{N(x, x_0)}{D(x, x_0)}, \tag{4}$$

where the rational polynomial

$$\frac{N(x, x_0)}{D(x, x_0)}$$

is Páde's is approximant. The computation of $f(x)$ can be carried out as follows which have quadratic, 3rd or 4th degree convergence rates according to the degree sum of $N(\cdot)$ and $D(\cdot)$ is one, two or three respectively.

(a) Define iteration

$$r_{i+1} = \frac{N(x, x_i)}{D(x, x_i)},$$

where $x_i = g(r_i)$, $g(y)$ is the inverse function of $f(x)$ and $g(y) = x$.

(b) Repeat (a) until some satisfied r_n, then set $f(x) = r_n$.

For the quadratic algorithm, we set $N(x, x_0) = f(x_0) + f^{(1)}(x_0)(x - x_0)$, and $D(\cdot)$ equal to 1. The error between $f(x)$ and its approximant is

$$\epsilon = \frac{f^{(2)}}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}}{n!}(x - x_0)^n + \cdots. \tag{5}$$

The rationale here is that we use Páde approximation as an iteration equation for a better truncated approximation to Taylor's series, and for every iteration we calculate a new expansion point for both Taylor's series and its Páde approximant to achieve a fast convergence rate. This approach is different from that of Newton's method; for the latter case has to set up an equation, then locates its root by equating it to zero and to the first degree Taylor's series. There are many ways to set up the equations for Newton's method. However, for the methods discussed here, it is obviously unique and is a subset of Newton's method for the quadratic algorithm as we shall see. The problem encountered in this method is the evaluation of the inverse function of $g(y)$ which limits its use to a small class of functions as is in the case of Newton's method when setting up a root-locating equation which also involves the inverse function.

If summation of the terms in equation (5) with degrees greater than two is less than or the same

order of magnitude as the 2nd order term, as is justified by Taylor's theroem [6] if $x - x_0 \leq 1/2$, then the algorithm converges quadratically. By the same token, we will have 3rd degree algorithm if we set $D(\cdot) = 1$ and

$$N(x, x_0) = f(x_0) + f^{(1)}(x_0)(x - x_0) + \frac{f^{(2)}(x_0)}{2}(x - x_0)^2,$$

or $D(x, x_0) = d_0 + (x - x_0)$ and $N(x, x_0) = c_0 + c_1(x - x_0)$. The latter approximant can be rearranged as

$$\frac{N(\cdot)}{D(\cdot)} = k_0 + \frac{k_1}{k_2 + (x - x_0)}$$

which has less computational cost and better approximation than the former case. For the 4th degree algorithm, we set $D(x, x_0) = d_0 + d_1(x - x_0)$ and $N(x, x_0) = c_0 + c_1(x - x_0) + c_2(x - x_0)^2$; rearranging it, we have

$$\frac{N(\cdot)}{D(\cdot)} = k_0 + k_1(x - x_0) + \frac{k_2}{k_3 + (x - x_0)}.$$

The major computational cost for every iteration of this 4th degree algorithm is one division, one multiplication, calculation of the inverse function for $f(x)$ and updating of the constants k_0 through k_3. For the quadratic algorithm, every iteration needs one multiplication, calculation of the inverse function $g(y)$ and updating of the constant $f^{(1)}(x_0)$. If constant updating is trivial compared with the rest of the computational costs, then the 4th degree algorithm may be more attractive than the quadratic algorithms, and this is the case for $x^{1/N}$.

According to the procedures described, the iterative equation for the quadratic algorithm of $x^{1/N}$ is

$$r_{i+1} = r_i + \frac{1}{N}[g(r_i)]^{(1/N)-1}[x - g(r_i)],$$

which is the same form as Newton's method if we expand $g(r_i)$. If N is replaced by $-N$, we will arrive at condition (2). Similarly, if $N = -1$, an identical form for x^{-1} of Newtons's method is derived.

3. NEW ALGORITHM FOR $x^{1/N}$ WITH 4th DEGREE CONVERGENCE RATE

Given a normalized number, x, $1 > x \geq 1/2$, Taylor's series for $x^{1/N}$ is

$$x^{1/N} = x_0^{1/N}\left[1 + \sum_{i=1}^{\infty}\prod_{j=0}^{i-1}\left(\frac{1}{N} - j\right)z^i/i!\right], \tag{6}$$

where

$$z = (x - x_0)/x_0.$$

Consider the 3rd degree Páde approximation of Stieltjés-type to equation (6)

$$x^{1/N} \approx x_0^{1/N}\left(1 + \frac{a_1 z}{1 + \dfrac{a_2 z}{1 + a_3 z}}\right), \tag{7}$$

where

$$a_1 = \frac{1}{N}, \quad a_2 = \left(1 - \frac{1}{N}\right)\Big/2, \quad a_3 = \left(1 + \frac{1}{N}\right)\Big/6.$$

The difference ϵ between equations (7) and (6) is

$$\epsilon = \frac{x_0^{1/N}}{72N}\left(1-\frac{1}{N}\right)\left(\frac{1}{N}-2\right)\left(\frac{1}{N}+1\right)z^4 + O(z^5). \tag{8}$$

Rearranging condition (7)

$$x^{1/N} \approx x_0^{1/N}\left[1+\frac{4.5(N-1)}{(2N-1)^2}+\frac{N+1}{2N(2N-1)}z-\frac{13.5N(N-1)}{(2N-1)^3}\left(z+\frac{3N}{2N-1}\right)^{-1}\right]. \tag{9}$$

Define iteration for the Nth root

$$r_{i+1} = r_i\left[1+\frac{4.5(N-1)}{(2N-1)^2}+\frac{N+1}{2N(2N-1)}z_i-\frac{13.5N(N-1)}{(2N-1)^3}\left(z+\frac{3N}{2N-1}\right)^{-1}\right], \tag{10}$$

where

$$z_i = \frac{x - r_i^N}{r_i^N}.$$

From equation (8), the relative error ϵ_{i+1} defined as

$$\frac{r_{i+1} - x^{1/N}}{r_{i+1}}$$

is proportional to z_i^4

$$\epsilon_{i+1} \approx \frac{r_i z_i^4}{36Nr_{i+1}}. \tag{11}$$

However, for Newton's method

$$\epsilon_{i+1} \approx \frac{r_i z_i^2}{2Nr_{i+1}}. \tag{12}$$

In addition to the 4th degree convergence rate, iteration (10) has four more precision bits produced than Newton's method does. To evaluate equation (10) iteratively for a result with M-bit precision, precalculations of the constants in equation (10) have to be performed. After that, every iteration needs two divisions, two multiplications and calculation of r_i^N, which takes $O(4+2\log_2 N)$ multiplications. Total of $O[(4+2\log_2 N)(\log_4 M)]$ multiplications for a result with M-bit precision. For Newton's methods (1), it requires two divisions and $O[(2\log_2(N-1)]$ multiplications for every iteration, total of $O\{[2+2\log_2(N-1)](\log_2 M)\}$ multiplications. The proposed algorithm turns out to be two times faster than Newton's method asymptotically.

4. A SPECIAL CASE: $x^{1/2}$

For $N = 2$, based on Páde approximation of Stieltjés-type, a new CFE for square-root can be derived [5] which is

$$x^{1/2} = x_0^{1/2}\left[1+\cfrac{\frac{z}{2}}{1+\cfrac{\frac{z}{4}}{1+\frac{z}{4}\cdots}}\right]. \tag{13}$$

This equation can be easily programmed with a few statements and evaluated recursively. Although it is free from error propagation, it converges linearly. Because $N = 2$ is too small, the algorithm given in the previous section is not economical; it requires two divisions and two multiplications for every iteration which is much inferior to only one division for Newton's method. If the 3rd degree approximation of condition (9) is rearranged [5], a starting approximation with at least 10-bit precision for Newton's method is obtained which needs only one division and a few

complement and shift operations as shown

$$x^{1/2} \approx x_0^{1/2} \left[1 + \frac{\frac{z}{2}}{1 + \frac{\frac{z}{4}}{1 + \frac{z}{4}}} \right]$$

$$= \begin{cases} 1 - \left[\dfrac{1}{1 - \frac{z}{2}} - \left(1 - \frac{z}{2}\right) \right] \Big/ 2, & \text{if exponent } p \text{ is even, } z = 1 - x, x_0 = 1; \\ \left\{ 1 - \left[\dfrac{1}{1 - \frac{z}{2}} - \left(1 - \frac{z}{2}\right) \right] \Big/ 2 \right\} \Big/ 2, & \text{if exponent } p \text{ is odd, } z = 1 - 2x, x_0 = 1/4. \end{cases} \quad (14)$$

The worst case starting approximation with 10-bit precision is when $x = 0.5$ or $x = 1$ corresponding to p is even or odd, which is equal to 17/24. The difference between 17/24 and $0.5^{1/2}$ is equal to 0.0012265521. It has better performance than that of Chebyshev's starting approximation [7] in terms of computational complexity except for the form of

$$a + \frac{b}{c + x}$$

yet condition (14) has trivial coefficients.

5. CONCLUSION

The 4th degree algorithm for $x^{1/N}$ is better than Newton's method only when N and M are sufficiently large. Similar results can be shown for the 3rd degree algorithm which is better than Newton's method asymptotically. More rigorous analysis can be done to locate the border-line values N and M, error propagation and computational complexities for both Newton's method and the proposed method. Especially when N is large that the computation of r_i^N may incur divergence of the result.

REFERENCES

1. W. A. Watson, T. Philipson and P. J. Oates, *Numerical Analysis, the Mathematics of Computing* (2nd edn). Arnold, London (1981).
2. G. D. Taylor, Optimal starting approximations for Newton's method. *J. Approx. Theory* **3**, 156–163 (1970).
3. R. P. Brent, A FORTRAN multiple-precision arithmetic package. *ACM Trans. Math. Soft.* **4**(1), 57–70 (1978).
4. G. A. Baker Jr, *Essential of Pade Approximants*. Academic Press, New York (1975).
5. S.-G. Chen and P. Hsieh, A new square root algorithm and its applications (in press).
6. W. Cheney and D. Kincaid, *Numerical Mathematics and Computing* (2nd edn). Brooks–Cole, Belmont, Calif. (1985).
7. I. Ninomiya, Best rational starting approximation and improved iteration for the square root. *Maths. Comput.* **24**(110), 391–404 (1970).