2. R. Floyd, The syntax of programming languages – a survey. *IEEE Transactions on Electronic Computers* **EG-13** (4), 346–353 (1964).
3. F. Jalili, A general incremental evaluator for attribute grammars. *Science of Computer Programming* **5**, 83–96 (1985).
4. N. Jones and M. Madsen, Attribute influenced LR parsing. In *Semantics Directed Compiler Generation*. Lecture Notes in Computer Sciences no. 94, pp. 393–407. Springer, Heidelberg (1980).
5. D. E. Knuth, Semantics of context free languages, *Mathematical Systems Theory*, **2** (2), 127–145 (1968) and **5** (1), 95–96 (1971).
6. D. R. Milton, *Syntactic Specification and Analysis with Attributed Grammars*. Technical Report 304, Computer Science Department, University of Wisconsin, Madison (1977).
7. G. Papakonstantinou and J. Kontos, Knowledge representation with attribute grammars. *The Computer Journal* **29** (3), 241–246 (1986).
8. K. J. Raiha, Bibliography on attribute grammars. *SIG-PLAN Notices* **15** (3), 35–44 (1980).
9. M. Sideri, S. Efremidis, G. Papakonstantinou and E. Skordalakis, Error recovery using attribute grammars. *First European Workshop on Fault Diagnostics, Reliability and Related Knowledge-Based Approaches*.
10 M. Sideri, S. Efremidis and G. Papakonstantinou, *Implementation of the System SDP*. Research Report, National Technical University of Athens (1986).
11. D. Watt, Rule splitting and attribute directed parsing. In *Semantics Directed Compiler Direction*. Lecture Notes in Computer Science no. 94, pp. 363–392 (1980).

Rapidly Converging Iterative Formulae for Finding Square Roots and their Computational Efficiencies

A derivation is given of rapidly converging iterative formulae for finding square roots which include, as special cases, some recently published examples. Their computational efficiencies are investigated for sequential and parallel implementation. It is concluded that the most efficient method is equivalent to sequential application of the Newton Raphson formula; a simple modification is suggested which brings the advantage of root bracketing at little extra computational cost.

Received June 1988

1. Introduction

Recently there has been renewed interest in the calculation of square roots.[1,2,3] The Newton Raphson method is well known and has second-order convergence; some aspects of its computer implementation have been discussed by Bentley.[1] A third-order method has been given by Moler and Morrison,[2] and a family of related methods was presented by Dubrulle.[3] A question arises as to their computational efficiencies for both sequential and parallel implementation. Below we derive formulae with kth-order convergence (for any integer $k > 1$) from elementary numerical methods and study their complexities; the study applies to the methods of Moler and Morrison[2] and Dubrulle,[3] since they use special cases of our formulae.

2. The iterative formulae

We wish to find a function $F_k(x)$ which generates an iterative sequence $\{x_n\}$ from a starting value x_0 by means of the equation

$$x_{n+1} = F_k(x_n), \quad n \geqslant 0 \tag{1}$$

such that

$$x_n \to \pm\sqrt{N} = \pm h, \tag{2}$$

say, as n tends to infinity with convergence of order k, the sign of the limit being the same as that of x_0. Thus we want

$$F_k(\pm h) = \pm h \tag{3}$$

and

$$F_k^{(i)}(\pm h) = 0, \quad i = 1, 2, \ldots, k-1. \tag{4}$$

Let us impose the condition that $F_k(x)$ be antisymmetric with the same sign as x, so that there is no possibility of generating a sequence which oscillates between positive and negative roots. The conditions (3) and (4) suggest functions involving the kth power of $(x \pm h)$. We can verify easily that the formulae

$$F_k^{\pm}(x) = h\frac{[(h+x)^k \pm (h-x)^k]}{[(h+x)^k \mp (h-x)^k]} \tag{5}$$

satisfy our conditions. They only involve h^2, that is N, and clearly have the same sign as x. They lead to stable iterative schemes because they contain no cancellation terms or terms which may produce rounding errors (provided that k is not too large, leading to large binomial coefficients).

3. Convergence

Consider

$$E = N - [F_k^{\pm}(x)]^2 = \frac{\mp 4N(N-x^2)^k}{[(h+x)^k \mp (h-x)^k]^2}. \tag{6}$$

The quantity E gives a measure of the deviation of x from the root, and Equation (6) enables us to study the behaviour of its sign. If k is even the sign of E does not change. The convergence is monotonic from above or below, save for possibly the initial estimate according to the choice of upper or lower sign in Equation (5). With the lower sign and k odd the sign of E does not change, and the convergence is monotonic from above if the initial estimate exceeds the root and from below otherwise. The most interesting situation occurs with the upper sign and k odd, for then the sign of E alternates so that any pair of consecutive values of the sequence $\{x_n\}$ brackets the appropriate (positive or negative) root.

Because use of Equation (5) leads to high-order convergence we assume that convergence is guaranteed provided the starting value is close enough to the root. Let us examine the range of possible starting values for guaranteed convergence. We need only consider convergence to the positive root with a positive starting value, because the iterative formulae have the same signs as x. From Equations (1) and (5) we find

$$x_{n+1} = h(1 \pm r_n^k)/(1 \mp r_n^k) \tag{7}$$

where

$$r_n = (h - x_n)/(h + x_n). \tag{8}$$

Hence for the $(n+1)$th error

$$h - x_{n+1} = \mp 2hr_n^k/(1 \mp r_n^k). \tag{9}$$

We find after manipulation from Equation (7) with n replaced by $n-1$ and Equation (8)

$$r_n = \mp r_{n-1}^k. \tag{10}$$

Repeated application of Equation (10) yields r_n in terms of r_0, and we see from Equation (9) that if $k > 1$ the error decreases as n increases provided that r_0 is less than unity in magnitude, which it is for any (non-zero) positive starting value. Thus we get convergence for any positive starting value with $k > 1$; the nearer the starting value is to the root the faster is the convergence. For the special case $k = 1$ the sequence remains fixed at the starting value (lower sign) or oscillates between it and N times its reciprocal (upper sign).

From Equation (5) we find

$$\begin{aligned} F_k^+(x) &\cong N/kx \quad \text{if } x \text{ is small,} \\ &\cong kN/x \quad \text{if } x \text{ is large for odd } k, \\ &\cong x/k \quad \text{if } x \text{ is large for even } k. \end{aligned} \tag{11}$$

The reader may easily see the corresponding behaviour of $F_k^-(x)$ by noting that it is $N/F_k^+(x)$. The convergence is initially slow if we start far from the root. In the problem of evaluating Pythagorean sums, $\sqrt{(a^2+b^2)}$, of use in computer graphics applications, a starting value which leads to rapid convergence is the larger of the magnitudes of a and b. For such sums Dubrulle[3] has derived, differently, an iterative formula which is the same as $F_k^-(x)$.

4. Computational efficiency

We need not consider the case with k equal to unity, since it does not lead to a converging iterative process. For other values of k we seek the number of operations required to evaluate the iterative formulae. The formula $F_k^+(x)$ is closely related to the Newton Raphson formula and we examine it first. For even k ($2m$, say) this function can be expressed as a constant times x plus a ratio of polynomials each of degree $m-1$ in x^2, the ratio being divided by x; the division by x can be incorporated in the numerator polynomial. For odd k ($2m+1$, say) the function has the form of a ratio of polynomials each of degree m in x^2, the ratio being divided by x; this division can also be incorporated in the numerator. We must, at least, calculate a polynomial of degree $k/2-1$ in x^2 for even k and degree $(k-1)/2$ for odd k.

Traub,[4] in a study of the complexity of iterative processes, stated a measure of the multiplicative efficiency as the ratio of $\log_2(k)$ to the number of multiplications and/or divisions required, not including those by constants. For the Newton Raphson process ($k = 2$) the efficiency is unity, but for any value of k other than a power of two, even with as efficient as possible evaluation of polynomials (e.g. Paterson and Stockmeyer)[5] we find below that it is less than unity. Kung[6] has shown that unity is the upper limit of the efficiency for iterative processes which converge to quadratically irrational numbers.

It is obvious that calculation of $F_k^+(x)$ by direct evaluation of the polynomials has an efficiency below unity except for $k = 2$, the Newton Raphson formula. However, these

functions satisfy a recurrence relation which is helpful to their evaluation. From Equation (5) we see that

$$F^+_{i+j}(x) = \frac{[F^+_i(x)\,F^+_j(x)+N]}{[F^+_i(x)+F^+_j(x)]},\ i,j \geqslant 1. \quad (12)$$

It is of interest to note that certain convergents to periodic continued fractions representing quadratic surds also satisfy this recurrence relation as shown by Frank;[7] such convergents, where the surd consists of a square root only, can be thought of as special cases of the estimates generated by $F^+_k(x)$. If we set $i = j$ we obtain from Equation (12) an expression of the same form as the Newton Raphson formula. Hence if k is a power of two, 2^l say, we can evaluate $F^+_k(x)$ by l applications of Newton's method with an efficiency of unity. If k is not a power of two the efficiency cannot be exactly unity because of the logarithm in its definition, and hence must be less than unity in this case.[6] Hence the greatest computational efficiency is obtained when k is a power of two, where the iterative process is equivalent to multiple applications of the Newton Raphson method.

When several processors are available the calculation of the polynomials may be done in parallel. Furthermore, if the processors are arranged in a tree-like structure the evaluation of each polynomial can be broken down into parts to be evaluated concurrently in a logarithmic cascade (Kogge).[8] We generalise the efficiency such that $\log_2(k)$ is divided by the number of steps involving division and multiplication (except by constants) where at each step several multiplications and/or divisions may be done concurrently; we do not consider the cost of the processors here. We have seen that we should at least choose k even for greatest efficiency. The number of steps is equal to the number required to evaluate one polynomial plus one for the ratio, which is the lowest integer upper bound of $1+\log_2(k-2)$ for k not two; the efficiency is maximised to unity only for $k = 4$. For $k = 2$ the efficiency is unity but extra processors are unnecessary. The use of several processors does not produce an efficiency greater than that of the sequential application of the Newton Raphson process; except for $k = 2$ and 4 even with parallel processing we cannot achieve the same efficiency. The multiple application of the Newton process cannot be accelerated in time by parallel processing because each application of it must await the completion of the previous one.

It was observed in Section 2 that we obtain an algorithm with a root-bracketing property by using the function $F^+_k(x)$ with k odd. In the light of the previous paragraph let us choose $k = 2^l+1$; we use l applications of the Newton Raphson method to obtain $F^+_{2^l}(x)$ and, noting that $F^+_1(x) = N/x$ from Equation (5), we have

$$F^+_k(x) = \frac{N[F^+_{2^l}(x)+x]}{[xF^+_{2^l}(x)+N]} \quad (13)$$

The efficiency falls from unity by around $2/l$ if l is not too small; in a parallel implementation it falls by around $1/l$ giving little advantage. The extra cost incurred to obtain root bracketing is not great.

We turn to the evaluation of the function $F^-_k(x)$. For even k it is best evaluated as N divided by $F^+_k(x)$, this single extra operation making its use slightly less efficient. Sequential Newton Raphson, l times, followed by division yields an efficiency around $1-1/l$. For odd k the function is the product of x and a ratio of polynomials of degree $(k-1)/2$ in x^2; the multiplier x can be absorbed in the denominator polynomial. The efficiency is not greatly enhanced by parallel computation. In general the function $F^-_k(x)$ is less useful than $F^+_k(x)$.

5. Conclusion

Iterative processes based on formula (5) have kth order convergence to $\sqrt{N}$ for any positive starting value and to $-\sqrt{N}$ for any negative starting value. The convergence is slow at first if the magnitude of the starting value is very small or very large. The processes are stable. No significant gain is obtained by the use of parallel processors (even when measured by latency alone). The most efficient formula is equivalent to multiple applications of the Newton Raphson method. It can be modified simply at little extra computational cost to give root bracketing (Equation (13)).

The formulae (5) include a number of recently published iterative expressions.[2,3,9] The above remarks apply to all of them. Those of Morrison and Moler[2] and Dubrulle[3] are equivalent to $F^-_k(x)$. The formula I derived differently[9] is equivalent to $F^-_k(x)$ for k odd and $F^+_k(x)$ for k even; I withdraw the suggestion[9] that parallel computation of it would be beneficial!

M. J. JAMIESON
Department of Computing Science,
University of Glasgow,
Glasgow G12 8QQ.

References

1. J. Bentley, Programming pearls – birth of a cruncher. *Communications of the Association for Computing Machinery* **29** (1986), 1155–1161.
2. C. Moler and D. Morrison, Replacing square roots by Pythagorean sums. *IBM Journal of Research and Development* **27** (1983), 577–581.
3. A. A. Dubrulle, A class of numerical methods for the computation of Pythagorean sums. *IBM Journal of Research and Development* **27** (1983), 582–589.
4. J. F. Traub, Computational complexity of iterative processes. *SIAM Journal on Computing* **1** (1972), 167–179.
5. M. S. Paterson and L. J. Stockmeyer, On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing* **2** (1973), 60–66.
6. H. T. Kung, The computational complexity of algebraic numbers. *SIAM Journal on Numerical Analysis* **12** (1975), 89–96.
7. E. Frank, On continued fractions and binomial quadratic surds. *Numerische Mathematik* **4** (1962), 85–95.
8. P. M. Kogge, *The Architecture of Pipelined Computers*, McGraw-Hill, Washington, New York, London (1981).
9. M. J. Jamieson, A note on the convergence of an iterative scheme for solving a quadratic equation. *The Computer Journal* **30** (1987), 189–190.

Announcement

10–14 JULY 1989

ECOOP '89

European Conference on Object-Oriented Programming
East Midlands Conference Centre, University of Nottingham

ECOOP '89 will be the third annual European Conference on Object-Oriented Programming. The conference will present the latest research in the object-oriented paradigm, and will provide a focus for discussion and exploration of practical applications of object-oriented systems. There will be one-day tutorials and three days of presentations of invited and refereed papers, with associated workshops and panel sessions. One half-day session will focus on papers describing object-oriented research from Esprit projects.

The topics to be addressed at ECOOP '89 include but are not limited to: Languages, Theory, Implementation, Applications, Databases, User interfaces, Design, Tools and environments, Concurrency and Teaching.

Other activities

Proposals are invited for workshops, panels, demonstrations, videos or other activities covering issues relevant to the object-oriented paradigm. Send a one- or two-page proposal, including the organiser's name, affiliation, address and phone number to the Programme Chairman before 27 February 1989.

Call for Exhibitors

If you or your company is interested in reserving exhibition space at ECOOP '89, please advise Julia Allen.

Conference venue

The East Midlands Conference Centre is situated on the campus of Nottingham University, two miles away from the city centre and its InterCity rail network. The M1 motorway is five minutes drive away, while the East Midlands International Airport can be reached in 20 minutes. The high-speed InterCity 125 train takes about 1½ hours to London.

For accommodation, there are comfortable study bedrooms available within the University campus. Alternatively Nottingham has numerous 3- and 4-star hotels.

Programme Chairman
Steve Cook, *Queen Mary College*

General Enquiries
Julia Allen, *British Informatics Society Ltd, 13 Mansfield Street, London W1M 0BP, UK*