Two Efficient Algorithms with Guaranteed Convergence for Finding a Zero of a Function

J.C.P. BUS

Mathematical Centre, The Netherlands

and

T.J. DEKKER

University of Amsterdam, The Netherlands

Two algorithms are presented for finding a zero of a real continuous function defined on a given interval. The methods used are mixtures of linear interpolation, rational interpolation, and bisection. The asymptotic behavior of these algorithms is completely satisfactory. The number of function evaluations needed to find a zero of a function is bounded by four or five times the number needed by bisection and is usually considerably smaller.

Key Words and Phrases: zero finding, linear interpolation, rational interpolation, nonlinear equation
CR Categories: 5.13, 5.15

1. INTRODUCTION

Our starting point is an algorithm, published by Dekker [3], for finding a zero of a real function defined on a given interval.

Section 2 contains a detailed discussion of this algorithm, which we call algorithm A. The method used in algorithm A is a mixture of linear interpolation and bisection. For this algorithm, convergence is guaranteed and the asymptotic behavior is completely satisfactory. However, the number of function evaluations required by this algorithm may be prohibitively large, in particular, when the zero appears to be multiple. Therefore, Brent [2] proposed a modified algorithm (called algorithm B in Section 5). For this algorithm the upper bound of the number of function evaluations needed equals $(t + 1)^2 - 2$, where t is the number of function evaluations needed by bisection.

A version of this paper was presented at Mathematical Software II, a conference held at Purdue University, West Lafayette, Indiana, May 29–31, 1974.
Authors' addresses: J.C.P. Bus, Mathematical Centre, Tweede Boerhaavestraat 49, Amsterdam, The Netherlands; T.J. Dekker, University of Amsterdam, Roetersstraat 15, Amsterdam, The Netherlands.

In Section 3 we present a modified algorithm (algorithm M) having the same asymptotic order of convergence as algorithm A, but requiring at most $4t$ function evaluations. This is achieved by inserting steps in which rational interpolation (see Jarratt and Nudds [5]) or bisection is performed. Anderson and Björck [1] present an algorithm (called algorithm C in Section 5) which uses also linear interpolation and rational interpolation. This algorithm, however, may require as many function evaluations as algorithm A.

In Section 4 we present another algorithm (algorithm R) having a higher asymptotic order of convergence and requiring at most $5t$ function evaluations. This algorithm has a similar strategy but uses rational interpolation instead of linear interpolation.

In Section 5 we compare some numerical results of the algorithms mentioned, and in Section 6 we give some conclusions.

A description of our algorithms in the form of Algol 60 procedures is given in the Appendix.

2. ALGORITHM A

For a detailed description of algorithm A, together with a discussion on its properties and an Algol 60 procedure, see [3].

2.1 Data

Given a real continuous function f of one real variable, two distinct argument values x_0 and x_1 satisfying $f(x_0) \times f(x_1) \leq 0$, and a positive tolerance function δ of one real variable satisfying $0 < \tau \leq \delta(x)$, where τ is a given positive constant (for instance, $\delta(x) \equiv \tau$ defines an absolute tolerance τ and $\delta(x) = \alpha \mid x \mid + \tau$ defines a relative tolerance α when $\mid x \mid$ is large).

2.2 Results

The purpose of algorithm A (and of algorithms M and R presented in Sections 3 and 4) is to find two (distinct) real numbers x and y satisfying

$$f(x) \times f(y) \leq 0, \quad |f(x)| \leq |f(y)|, \quad |x - y| \leq 2\delta(x). \tag{2.2.1}$$

Since f is continuous, the first condition ensures that there exists a zero, z, of f in the closed interval with endpoints x and y; the second condition yields that x is the "best" approximation of z; the third condition states that the required tolerance has been reached.

2.3 Definition of Algorithm A

From the data mentioned in Section 2.1, algorithm A produces two argument values x and y satisfying (2.2.1). This is achieved by calculating in succession the argument values x_i (for $i = 2, \ldots, n$), and a_i, b_i, and c_i (for $i = 1, \ldots, n$) as defined in A1 and A2 below, where n and the results delivered are defined in A3.

A1 (initialization, $i = 1$). If $|f(x_1)| \leq |f(x_0)|$, then $b_1 = x_1$ and $a_1 = c_1 = x_0$; otherwise $b_1 = x_0$ and $a_1 = c_1 = x_1$.

A2 (iteration step, $i = 2, \dots, n$). Let the linear interpolation formula be defined, for $a \neq b$, by

$$\begin{aligned} l = l(b,a) &= b - f(b)(b-a)/(f(b) - f(a)) && \text{if } f(b) \neq f(a), \\ &= \infty && \text{if } f(b) = f(a) \neq 0, \\ &= b && \text{if } f(b) = f(a) = 0. \end{aligned} \quad (2.3.1)$$

Let, moreover,

$$h = h(b,c) = b + \operatorname{sign}(c-b) \times \delta(b), \quad (2.3.2)$$

$$m = m(b,c) = \tfrac{1}{2}(b+c), \text{ and} \quad (2.3.3)$$

$$\begin{aligned} v = v(l,b,c) &= l && \text{if } l \text{ is between } h(b,c) \text{ and } m(b,c), \\ &= h(b,c) && \text{if } |l-b| \leq \delta(b), \\ &= m(b,c) && \text{otherwise.} \end{aligned} \quad (2.3.4)$$

Then the new iterate x_i is calculated according to the formula

$$x_i = v(\lambda_i, b_{i-1}, c_{i-1}), \quad (2.3.5)$$

where $\lambda_i = l(b_{i-1}, a_{i-1})$. Furthermore, let k be the largest (nonnegative) integer satisfying $k < i$ and $f(x_k) \times f(x_i) \leq 0$. Then b_i, c_i, and a_i are defined by

$$b_i = x_i,\ c_i = x_k,\ a_i = b_{i-1} \quad \text{if } |f(x_i)| \leq |f(x_k)|; \quad (2.3.6)$$

$$b_i = x_k,\ a_i = c_i = x_i \quad \text{otherwise.} \quad (2.3.7)$$

A3 (termination). Let n be the smallest positive integer satisfying

$$|b_n - c_n| \leq 2\delta(b_n). \quad (2.3.8)$$

Then the algorithm terminates for $i = n$ and delivers as results

$$x = b_n,\ y = c_n. \quad (2.3.9)$$

2.4 Additional Definitions and Remarks

2.4.1 Let J_i, for $i = 1, 2, \dots, n$, denote the closed interval whose endpoints are b_i and c_i. Then, from the *invariant relations* $f(b_i) \times f(c_i) \leq 0$ and $|f(b_i)| \leq |f(c_i)|$, it follows that J_i contains a zero z of f and that b_i is the best approximation of z obtained up to and including step i.

2.4.2 The iterates x_i $(i = 1, 2, \dots, n)$ are all distinct and their mutual distances are at least τ. Hence $|b_i - a_i| \geq \tau$ for $i = 1, 2, \dots, n$, so that λ_i and x_i in (2.3.5) are well defined for $i = 1, 2, \dots, n$.

2.4.3 If $a_{i-1} = c_{i-1}$, for certain i, then the argument values a_{i-1} and b_{i-1}, used to calculate λ_i in (2.3.5), are on different sides of z and we call the ith step a (*linear*) *intrapolation step*; otherwise a_{i-1} and b_{i-1} are on the same side of z and we call the ith step a (*linear*) *extrapolation step*.

2.4.4 Obviously, algorithm A uses the function values $f(x_i)$, for $i = 0, 1, \dots, n$. So, the number of function evaluations needed equals $n + 1$.

2.5 Properties

Algorithm A has the following properties [3].

2.5.1 If the given function f has a continuous second derivative in J_1 and a unique simple zero in this interval, then the asymptotic order of convergence of algorithm A equals the largest root, p_1, of the equation $x^2 - x - 1 = 0$; thus $p_1 = \frac{1}{2}(1 + \sqrt{5}) \cong 1.618$.

2.5.2 The number of function evaluations needed is bounded above by T, where $T = |x_1 - x_0|/\tau$. As Brent [2] shows, this upper bound may indeed be attained.

2.6 Discussion

If $f(x_i) \times f(b_{i-1}) \leq 0$ for certain i, then

$$|b_i - c_i| = |x_i - b_{i-1}| \leq \tfrac{1}{2}|b_{i-1} - c_{i-1}|;$$

otherwise

$$\tfrac{1}{2}|b_{i-1} - c_{i-1}| \leq |b_i - c_i| = |x_i - c_{i-1}| \leq |b_{i-1} - c_{i-1}| - \tau.$$

So, we may have (very) slow convergence only if the latter case occurs frequently.

If f has a continuous second derivative, z is a simple zero of f (i.e., $f'(z) \neq 0$), and a and b are sufficiently close to z to ensure that $f'(\eta) \neq 0$ for η in the smallest interval containing a, b, and z, then $l = l(b, a)$, obtained by the linear interpolation formula (2.3.1), satisfies [3]

$$l - z = (b - z)(a - z)K(\xi, \eta), \tag{2.6.1}$$

where $K(\xi, \eta) = \frac{1}{2}f''(\xi)/f'(\eta)$, and ξ and η lie in the smallest interval containing a, b, and z. Hence if $|b_{i_0} - c_{i_0}|$ is sufficiently small for certain i_0, then the iterates x_i converge to z and the values $|f(x_i)|$ decrease monotonically for $i \geq i_0$ as long as

$$\delta(x_i) < |l(x_i, x_{i-1}) - x_i|. \tag{2.6.2}$$

Condition (2.6.2) ensures that, for $i \geq i_0$, the tolerance function does not influence the ith iteration step. Henceforth in this section (where we consider the asymptotic behavior of algorithm A) we take $i \geq i_0$ and assume that condition (2.6.2) holds for all $i \geq i_0$. (In fact we consider the process that is obtained if the tolerance function δ tends uniformly to zero on the interval J_1; see also the proof of Theorem 3.3.2). Then, by A2, we have $b_i = x_i$, $a_i = x_{i-1}$, and $c_i = x_k$. Let $\epsilon_i = b_i - z (= x_i - z)$ denote the error of the ith iterate. Then (2.3.5) and (2.6.1) yield

$$\epsilon_{i+1} = \epsilon_i \epsilon_{i-1} K(\xi_i, \eta_i), \tag{2.6.3}$$

where ξ_i and η_i lie in the smallest interval containing b_i, b_{i-1}, and z. Consequently, if $f''(z) \neq 0$, we have $K(z, z) \neq 0$. Hence, for sufficiently large i, $K(\xi_i, \eta_i)$ has the same sign as $K(z, z)$. Therefore the sign of $K(z, z)$ and of two successive errors ϵ_i and ϵ_{i-1} completely determine the signs of the subsequent errors. Then simple checking yields that, when $f''(z) \neq 0$, there are only the following two (essentially different) possibilities for the asymptotic behavior:

(1) the iteration consists of consecutive cycles of the form IIE, i.e. two intrapolation steps followed by one extrapolation step;
(2) the iteration consists of consecutive extrapolation steps.

In the first case, the length of J_i is smaller than 0.25 times the length of J_{i-3}. So, in this case, we find a small upper bound (viz. $\frac{3}{2}t$) for the number, N, of function evaluations needed. In the second case, convergence may be very slow (N may attain the upper bound T). Therefore, we modify algorithm A such that more than two consecutive extrapolation steps can no longer occur in an iteration, while an iteration consisting of consecutive cycles of the form IIE remains undisturbed.

3. ALGORITHM M

3.1 Definition

From the data mentioned in Section 2.1, algorithm M pıoduces two argument values x and y satisfying (2.2.1). This is achieved by calculating in succession the argument values x_i, d_i (for $i = 2, \ldots, n$), and a_i, b_i, c_i (for $i = 1, \ldots, n$) as defined in A1 (see Section 2.3) and M2 (below), where n and the results delivered are defined in A3 (see Section 2.3).

M2 (iteration step, $i = 2, \ldots, n$). Let $j = j_i$ be the largest positive integer satisfying $j = 1$ or, if $1 < j < i$, then

$$| b_j - c_j | \leq \tfrac{1}{2} | b_{j-1} - c_{j-1} |. \qquad (3.1.1)$$

Then the new iterate x_i is calculated as follows (for the definitions of h, m, and λ_i, see A2). Let

$$\begin{aligned} w = w(l, b, c) &= l && \text{if } l \text{ is between } h(b, c) \text{ and } m(b, c), \\ &= h(b, c) && \text{if } | l - b | \leq \delta(b) \text{ and } l \text{ lies not outside the} \\ & && \text{interval bounded by } b \text{ and } m(b, c), \\ &= m(b, c) && \text{otherwise.} \qquad (3.1.2) \end{aligned}$$

Then

$$\begin{aligned} x_i &= w(\lambda_i, b_{i-1}, c_{i-1}) && \text{if } j_i \geq i - 2, \\ &= w(\rho_i, b_{i-1}, c_{i-1}) && \text{if } j_i = i - 3, \\ &= m(b_{i-1}, c_{i-1}) && \text{otherwise,} \qquad (3.1.3) \end{aligned}$$

where ρ_i is defined as follows: for $a \neq b$, let

$$f[a, b] = (f(a) - f(b))/(a - b) \qquad (3.1.4)$$

(i.e. the first divided difference of f at a and b); for distinct a, b, and d, using the abbreviations $\alpha = f[b, d] \times f(a)$ and $\beta = f[a, d] \times f(b)$, define

$$\begin{aligned} r = r(b, a, d) &= b - \beta(b - a)/(\beta - \alpha) && \text{if } \beta \neq \alpha, \\ &= \infty && \text{if } \beta = \alpha \neq 0, \\ &= 0 && \text{if } \beta = \alpha = 0; \qquad (3.1.5) \end{aligned}$$

then

$$\rho_i = r(b_{i-1}, a_{i-1}, d_{i-1}). \qquad (3.1.6)$$

Furthermore, let k be the largest (nonnegative) integer satisfying $k < i$ and

$f(x_k) \times f(x_i) \leq 0$, then b_i, c_i, a_i, and d_i are defined by

$$b_i = x_i, \; c_i = x_k, \; a_i = b_{i-1} \quad \text{if } |f(x_i)| \leq |f(x_k)|; \tag{3.1.7}$$

$$b_i = x_k, \; a_i = c_i = x_i \quad \text{otherwise}; \tag{3.1.8}$$

$$d_i = a_{i-1} \quad \text{if } b_i = x_i \text{ or } b_i = b_{i-1};$$

$$d_i = b_{i-1} \quad \text{otherwise}. \tag{3.1.9}$$

3.2 Additional Definitions and Remarks

The definitions and remarks given in Section 2.4 are also valid for algorithm M.

3.2.1 Formula (3.1.5) is obtained by 3-point rational interpolation, where the interpolating function is $\phi(x) = (x - r)/(px + q)$ and the parameters p, q, and r are determined such that $\phi(x) = f(x)$ for $x = a, b, d$ (see also [5]).

3.2.2 In addition to paragraph 2.4.2, it is obvious that for all $i \geq 2$ the argument values b_i, a_i, and d_i are distinct and have a mutual distance which is bounded below by τ. So ρ_i and x_i in (3.1.6) and (3.1.3) are well defined.

3.2.3 In addition to paragraph 2.4.3 we speak about *rational interpolation* if $x_i = w(\rho_i, b_{i-1}, c_{i-1})$. Moreover, if in this case b_{i-1} and a_{i-1} lie on different sides of z, then we call the ith step a *rational intrapolation step*; otherwise we call the ith step a *rational extrapolation step.*

3.2.4 Comparing the definitions of w and v, given in (3.1.2) and (2.3.4), respectively, we note that $w(l, b, c) \neq v(l, b, c)$ only if $|l - b| \leq \delta(b)$ and l lies outside the interval bounded by b and $m(b, c)$. We have replaced v by w in algorithm M because we think it is preferable from a theoretical point of view, and it sometimes yields better results.

3.3 Properties

We state and prove the following two theorems on algorithm M.

3.3.1 THEOREM. *Let data be given as mentioned in Section* 2.1. *Then the number of function evaluations needed by algorithm M to obtain two values x and y satisfying* (2.2.1) *is bounded by* $4t$, *where* $t = {}^2\log(|x_1 - x_0|/\tau)$.

(Note that t is the number of function evaluations needed by bisection.)

PROOF. This follows from the definition of the algorithm, in particular from formulas (3.1.1) and (3.1.2). A bisection step is performed whenever none of the last three steps has reduced the length of the interval by a factor less than or equal to 0.5. Hence the length of J_i is smaller than half the length of J_{i-4}, which proves the theorem. $\square$

3.3.2 THEOREM. *Let data be given as mentioned in Section* 2.1. *Let, moreover, the given function f have a continuous fourth derivative and a unique simple zero, z, in the interval* J_1. *Then the asymptotic order of convergence of algorithm M, finding an approximation of z, equals* p_1.

(For definitions of J_1 and p_1 see paragraphs 2.4.1 and 2.5.1.)

PROOF. Let

$$c_k = f^{(k)}(z)/k! \qquad k > 0. \tag{3.3.1}$$

Then $c_1 \neq 0$, because z is a simple zero of f by assumption. We need more terms in the error formula (2.6.1). By straightforward calculation, using Newton's interpolation formula and the assumption that f has a continuous fourth derivative, we find

$$l - z = (b - z)(a - z)[K_0 - K_1(b - z + a - z) + O(|b - z| + |a - z|)^2], \tag{3.3.2}$$

where $K_0 = c_2/c_1$ and $K_1 = (c_2/c_1)^2 - c_3/c_1$. Similarly, for the 3-point rational interpolation formula (3.1.5) we find [5]:

$$r - z = (b - z)(a - z)(d - z)[K_1 + O(|b - z| + |a - z| + |d - z|)]. \tag{3.3.3}$$

From (3.3.3) it follows that the asymptotic order of convergence of the 3-point rational interpolation formula equals p_2, where p_2 is the largest root of the equation $x^3 - x^2 - x - 1 = 0$; hence $p_2 \approx 1.839$ (cf. [5]).

We consider the asymptotic order of convergence of the iteration process that is obtained if we let the tolerance function δ tend uniformly to zero on the interval J_1. (We assume, of course, that exact arithmetic is used.) This limit process is a well defined iteration process which does, however, not terminate. (Here we use the fact that the divided difference $f[a, b]$ converges to $f'(a)$ when b converges to a). The intervals J_i $(i = 1, 2, \ldots)$ are monotonically nonincreasing (i.e. $J_{i+1} \subset J_i$, for all i) and the length of the interval J_i converges to zero for i tending to infinity. (Indeed the length decreases by a factor less than or equal to 0.5 in every four steps; cf. the proof of Theorem 3.3.1). We choose i_0 such that $f'(x) \neq 0$ for $x \in J_{i_0}$.

From the definition of the algorithm, in particular (3.1.1) and (3.1.3), and from the error formulas (3.3.2) and (3.3.3), we know that an integer $i_1 \geq i_0$ exists, such that

(a) for all $i > i_1$ satisfying $j_i > i - 3$, a bisection step is performed to obtain the ith iterate x_i (i.e. $x_i = m(b_{i-1}, c_{i-1})$); so, $|f(x_i)| > |f(b_{i-1})|$ and $f(x_i) \times f(b_{i-1}) \leq 0$; in this case a_i, b_i, and c_i are chosen according to (3.1.8), and the $(i + 1)$-th step will be an intrapolation step;

(b) for all $i > i_1$ satisfying $j_i \leq i - 3$, we have $|f(x_i)| \leq |f(b_{i-1})|$ and $|x_i - z| \leq |x_{i-1} - z|$; now b_i, a_i, and c_i are obtained by (3.1.7); substituting $\epsilon_k = b_k - z$ for arbitrary k in (3.3.2) and (3.3.3), we obtain

$$\lambda_i - z = \epsilon_{i-1}\epsilon_{i-2}[K_0 - K_1(\epsilon_{i-1} + \epsilon_{i-2}) + O(|\epsilon_{i-1}| + |\epsilon_{i-2}|)^2], \tag{3.3.4}$$

$$\rho_i - z = \epsilon_{i-1}\epsilon_{i-2}\epsilon_{i-3}[K_1 + O(|\epsilon_{i-1}| + |\epsilon_{i-2}| + |\epsilon_{i-3}|)]. \tag{3.3.5}$$

We distinguish two cases.

(A) There exists an $i_2 \geq i_1$ such that $j_i \geq i - 3$ for all $i \geq i_2$. Then, for all $i \geq i_2$, the iterate x_i is obtained by linear interpolation (with asymptotic order of convergence equal to p_1) or by 3-point rational interpolation (with asymptotic order of convergence equal to $p_2 > p_1$). This leads immediately to the required result.

(B) For each $i_2 \geq i_1$, there exists an $i \geq i_2$ such that $j_i < i - 3$.

We distinguish two subcases.

(B.1) $c_2 \neq 0$. So $K_0 \neq 0$. Hence an integer $\nu \geq i_2$ exists such that $j_\nu < \nu - 3$ and K_0 in formula (3.3.4) dominates. Consequently, using (a), the $(\nu + 1)$-th step is an intrapolation step and the sign of $\epsilon_i (i > \nu)$ is determined by the sign of ϵ_ν,

$\epsilon_{\nu-1}$, and K_0. Then it is easily checked that, from the $(\nu + 1)$-th step, the iteration consists of consecutive cycles of the form IIE, i.e. two linear intrapolation steps followed by one linear extrapolation step. This contradicts our assumption (B).

(B.2) $c_2 = 0$. Then also $K_0 = 0$.

We again distinguish two subcases.

(B.2.1) $c_3 \neq 0$. So $K_1 \neq 0$. Hence, as in (B.1), an integer $\nu \geq i_2$ exists such that the $(\nu + 1)$-th step is a linear intrapolation step and the term $K_1(\epsilon_{i-1} + \epsilon_{i-2})$ in formula (3.3.4) and the term K_1 in (3.3.5) dominate. Consequently, the sign of $\epsilon_i(i > \nu)$ is completely determined by the sign of ϵ_ν, $\epsilon_{\nu-1}$, and K_1. (Note that $\epsilon_i(i > \nu)$ equals either $\lambda_i - z$ or $\rho_i - z$ and that a rational extrapolation step always yields an iterate on the other side of z. So this step is always followed by a linear intrapolation step.) It can be shown that from the $(\nu + 1)$-th step the iteration consists of either only linear intrapolation steps (viz. when $K_1 > 0$) or cycles of the form IEE′, i.e. a linear intrapolation step, a linear extrapolation step, and a rational extrapolation step. This also contradicts our assumption (B).

(B.2.2) $c_3 = 0$. Then also $K_1 = 0$ and the most unfavorable situation is an iteration consisting of consecutive cycles of the form IEE′B, i.e. a linear intrapolation step, a linear extrapolation step, a rational extrapolation step, and a bisection step. Let the ith step be a bisection step yielding argument values $a_i = c_i = x_i$ and $b_i = x_{i-1}$. Then $a_i - z = O(1)$ and, according to (3.3.4) and (3.3.5), the cycle IEE′B asymptotically yields:

I: $\epsilon_{i+1} = \lambda_{i+1} - z = O(\epsilon_i(c_i - z)^3) = O(\epsilon_i)$,

E: $\epsilon_{i+2} = \lambda_{i+2} - z = O(\epsilon_{i+1}\epsilon_i^3) = O(\epsilon_i^4)$,

E′: $\epsilon_{i+3} = \rho_{i+3} - z = O(\epsilon_{i+2}\epsilon_{i+1}\epsilon_i^2) = O(\epsilon_i^7)$,

B: $\epsilon_{i+4} = \epsilon_{i+3} = O(\epsilon_i^7)$ and $a_{i+4} = c_{i+4} = x_{i+4}$.

So in this case the effective asymptotic order of convergence equals $\sqrt[4]{7} \cong 1.626$, which is greater than p_1. This completes the proof of the theorem. □

4. ALGORITHM R

4.1 Definition

From the data mentioned in Section 2.1, algorithm R produces two argument values x and y satisfying (2.2.1), by successively calculating argument values x_i and d_i (for $i = 2, \ldots, n$) and a_i, b_i, and c_i (for $i = 1, \ldots, n$) as defined in A1 (see Section 2.3) and R2 (below), where n and the results delivered are defined in A3 (see Section 2.3).

R2 (iteration step, $i = 2, \ldots, n$). Let j_i be defined as in M2. Then the new iterate x_i is calculated as follows (for the definitions of λ_i and m, see A2, and for the definitions of w and ρ_i, see M2):

$$\begin{aligned} x_i &= w(\lambda_i, b_{i-1}, c_{i-1}) && \text{if } i = 2, \\ &= w(\rho_i, b_{i-1}, c_{i-1}) && \text{if } i \geq 3 \text{ and } j_i \geq i - 3, \\ &= w(2\rho_i - b_{i-1}, b_{i-1}, c_{i-1}) && \text{if } i \geq 3 \text{ and } j_i = i - 4, \\ &= m(b_{i-1}, c_{i-1}) && \text{otherwise.} \end{aligned} \tag{4.1.1}$$

Furthermore, b_i, c_i, a_i, and d_i are defined as in M2.

4.2 Additional Definitions and Remarks

The definitions and remarks in Sections 2.4 and 3.2 are also valid for algorithm R.

4.2.1 In algorithm M a bisection step is performed ($x_i = m(b_{i-1}, c_{i-1})$) when $j_i = i - 4$, but in algorithm R a bisection step is performed when $j_i = i - 5$. The reason for this difference lies in the different asymptotic behavior of the algorithms M and R. Using 3-point rational interpolation, the errors satisfy (3.3.3). Assuming $K_1 \neq 0$, then the iteration may asymptotically consist of consecutive cycles of the form IIEE, i.e. two intrapolation steps followed by two extrapolation steps (see also the proof of Theorem 4.3.2). We do not want to disturb such an asymptotic behavior. So we have to allow two consecutive extrapolation steps in algorithm R. Therefore, in algorithm R, we modify the third of three consecutive extrapolation steps ($j_i = i - 4$) by doubling the step length obtained with rational interpolation, and a bisection step is inserted if $j_i < i - 4$.

4.2.2 In addition to paragraphs 2.4.3 and 3.2.3, we call an iteration step a *modified extrapolation step* if $x_i = w(2\rho_i - b_{i-1}, b_{i-1}, c_{i-1})$.

4.3 Properties

We state and prove the following two theorems on algorithm R.

4.3.1 THEOREM. *Let data be given as mentioned in Section* 2.1. *Then the number of function evaluations needed by algorithm R to produce two argument values x and y satisfying* (2.2.1) *is at most* 5*t*.

(For the definition of t see Section 3.3.1.)

PROOF. This follows immediately from the definition of the algorithm. □

4.3.2 THEOREM. *Let data be given as mentioned in Section* 2.1. *Let, moreover, the given function f have a continuous fifth derivative and a unique simple zero, z, in the interval J_1. Then the asymptotic order of convergence of algorithm R, to find an approximation of z, equals p_2.*

(For the definition of J_1 see paragraph 2.4.1, and for the definition of p_2 see the proof of Theorem 3.3.2.)

PROOF. This proof is similar to that of Theorem 3.3.2. Let c_k, $k > 0$, be defined by (3.3.3). Then $c_1 \neq 0$ by assumption. As in the proof of Theorem 3.3.2, we consider the asymptotic order of convergence of the iteration process that is obtained if we let the tolerance function δ tend uniformly to zero on the interval J_1. The length of the intervals J_i converges to 0 for i tending to infinity. So we may choose i_0 such that $f'(x) \neq 0$ for all $x \in J_{i_0}$. From the definition of the algorithm and the error formula (3.3.3) we may conclude that an integer $i_1 \geq i_0$ exists such that

(a) for all $i \geq i_1$, satisfying $j_i = i - 4$, a modified extrapolation step is performed; then, using the notation $\epsilon_k = b_k - z$ for arbitrary k, we obtain the following error formula:

$$\epsilon_i = 2\rho_i - b_{i-1} - z = -\epsilon_{i-1}[1 + O(\epsilon_{i-2}\epsilon_{i-3})]; \tag{4.3.1}$$

hence $f(x_i) \times f(b_{i-1}) \leq 0$ and the next step will be an intrapolation step;

(b) for all $i \geq i_1$, satisfying $j_i \geq i - 3$, the relations $|f(x_i)| \leq |f(b_{i-1})|$ and $|x_i - z| \leq |b_{i-1} - z|$ hold; consequently, b_i, a_i, and c_i are obtained by (3.1.7).

Note that for all $i \geq i_1$, the inequality $j_i \geq i - 4$ holds because of (a). So no bisection steps occur.

Instead of (3.3.5), we need a more elaborate error formula for this proof, which can be obtained by straightforward calculation using the assumption that f has a continuous fifth derivative.

$$\rho_i - z = \epsilon_{i-1}\epsilon_{i-2}\epsilon_{i-3}[K_1 + K_2(\epsilon_{i-1} + \epsilon_{i-2} + \epsilon_{i-3}) + O(|\epsilon_{i-1}| + |\epsilon_{i-2}| + |\epsilon_{i-3}|)^2], \tag{4.3.2}$$

where K_1 is defined by (3.3.2) and $K_2 = c_2c_3/c_1^2 - c_4/c_1$. We distinguish two cases.

(A) There exists an integer $i_2 \geq i_1$ such that $j_i \geq i - 3$ for all $i \geq i_2$. Then, for all $i \geq i_2$, the iterate x_i is obtained by rational interpolation (with asymptotic order of convergence equal to p_2). This proves the required result.

(B) For each $i_2 \geq i_1$, there exists an $i \geq i_2$ such that $j_i = i - 4$. Hence the ith step is a modified step.

We distinguish two subcases.

(B.1) $K_1 \neq 0$. By assumption (B) we may choose an integer $\nu \geq i_2$ such that the νth step is a modified extrapolation step and the term K_1 in (4.3.2) dominates. Consequently, using (a), the $(\nu + 1)$-th step is an intrapolation step and the sign of $\epsilon_i (i > \nu)$ is completely determined by the sign of ϵ_k $(k = \nu, \nu - 1, \nu - 2)$ and K_1. Then it is easily checked that, from the $(\nu + 1)$-th step, the iteration can only consist of cycles of the form I or IE, when $K_1 > 0$, and IIEE, when $K_1 < 0$; here I denotes a rational intrapolation step and E denotes a rational extrapolation step. This contradicts our assumption (B).

(B.2) $K_1 = 0$. Then the most unfavorable situation is an iteration consisting of cycles IEEE′, i.e. a rational intrapolation step, two rational extrapolation steps, and a modified extrapolation step. Then, according to (4.3.2), we have $\epsilon_i = -\epsilon_{i-1} + O(\epsilon_{i-1}\epsilon_{i-2}\epsilon_{i-3}^2)$, and the cycle IEEE′ yields:

$$\text{E}': \quad \epsilon_i = -\epsilon_{i-1} + O(\epsilon_{i-1}\epsilon_{i-2}\epsilon_{i-3}^2) = O(\epsilon_{i-1});$$

$$\text{I}: \quad \epsilon_{i+1} = O(\epsilon_i\epsilon_{i-1}\epsilon_{i-2}^2) = O(\epsilon_{i-1}^2\epsilon_{i-2}^2);$$

$$\begin{aligned}\text{E}: \quad \epsilon_{i+2} &= \epsilon_{i+1}\epsilon_i\epsilon_{i-1}[K_2(\epsilon_{i+1} + O(\epsilon_{i-1}\epsilon_{i-2}\epsilon_{i-3}^2)) + O(\epsilon_{i-1}^2)]\\ &= O(\epsilon_{i-1}^5\epsilon_{i-2}^2(\epsilon_{i-2}\epsilon_{i-3}^2 + \epsilon_{i-1}));\end{aligned}$$

$$\text{E}: \quad \epsilon_{i+3} = O(\epsilon_{i+2}\epsilon_{i+1}\epsilon_i^2) = O(\epsilon_{i-1}^9\epsilon_{i-2}^4(\epsilon_{i-2}\epsilon_{i-3}^2 + \epsilon_{i-1})).$$

Using similar relations for the $(i + 4)$-th up to the $(i + 7)$-th iteration step we obtain

$$\epsilon_{i+7} = O(\epsilon_{i+3}^9\epsilon_{i+2}^4(\epsilon_{i+2}\epsilon_{i+1}^2 + \epsilon_{i+3})) < O(\epsilon_{i+3}^9\epsilon_{i-1}^{29}).$$

Therefore, the effective asymptotic order of convergence is at least equal to $\sqrt[4]{\zeta}$, where ζ denotes the largest positive root of the equation $x^2 - 9x - 29 = 0$, which approximately equals 11.52. So $\sqrt[4]{\zeta} \cong 1.842$, which is larger than p_2. This completes the proof of the theorem. □

Remark. In fact, for analytic functions having a simple zero, it can be shown that no modified steps will asymptotically occur in the iteration of algorithm R. So the asymptotic order of convergence of algorithm R is as large as that of an iteration process using 3-point rational interpolation throughout.

5. NUMERICAL RESULTS

We have compared five algorithms for calculating a zero of a function of one variable:

Algorithm A, published by Dekker [3] and described in Section 2;
Algorithm M, defined in Section 3;
Algorithm R, defined in Section 4;
Algorithm B, published by Brent [2] (see Section 1);
Algorithm C, published by Anderson and Bjorck [1] (see Section 1).

For testing these algorithms we have chosen four groups of test functions.

I. Some functions with a simple zero in the interval considered. These functions are [4]:
1. $f(x) = \sin(x) - 0.5$ in the interval $[0, 1.5]$;
2. $f(x) = 2x\, exp(-n) + 1 - 2\, exp(-nx)$ in the interval $[0, 1]$ and $n = 1, 2, 3, 4$;
3. $f(x) = (1 + (1 - n)^2)x - (1 - nx)^2$ in the interval $[0, 1]$, and $n = 1, 5, 10$; these functions have one turning point in $[0, 1]$;
4. $f(x) = x^2 - (1 - x)^n$ in the interval $[0, 1]$, and $n = 1, 5, 10$; these functions have one inflexion in $[0, 1]$;
5. $f(x) = (1 + (1 - n)^4)x - (1 - nx)^4$ in the interval $[0, 1]$, and $n = 1, 4, 8$; these functions have one turning point and one inflexion in $[0, 1]$;
6. $f(x) = (x - 1)exp(-nx) + x^n$ in the interval $[0, 1]$, and $n = 1, 5, 10$; this is a family of curves increasingly close to the x-axis for large n.

II. Some functions of the form $f(x) = x^n + ax + b$, where $n = 3, 5, 9, 19$, and
1. $a = 1$ and $b = 0$;
2. $a = 0$ and $b = 10-4$;
3. $a = 1$ and $b = 10-4$.

These functions have a simple zero and an inflexion point of the order $n - 1$ or n at the zero or in its neighborhood.

III. Some simple polynomials with a multiple zero. $f(x) = x^n$ in the interval $[-1, 10]$, and $n = 3, 5, 7, 9, 19, 25$. These functions have a zero of multiplicity n.

IV. A function given by Brent [2] for which all the derivatives vanish at the zero of the function ("multiplicity ∞"). This function is defined by

$$\begin{aligned} f(x) &= 0 && \text{if } x = 0, \\ &= x\, exp(-x^{-2}) && \text{otherwise.} \end{aligned}$$

The interval is chosen to be $[-1, 4]$.

The testing has been performed on a Cyber 73 computer, which has a machine precision of 48 bits. In all examples the tolerance function is chosen to be $\delta(x) = |x| \times 10-14 + 10-14$.

The results for these groups of test functions are given in Tables I–IV. In these tables we give the number of function evaluations needed by the various algorithms to find a zero of the given function within the given precision.

Table I shows that algorithm M behaves almost the same as algorithm A for simple zeros, while algorithms R, B, and C are slightly better. The better results for algorithm R are due to the use of the higher order rational interpolation formula (3.1.5) throughout. The better behavior of algorithms B and C is caused by re-

Table I. Test Functions of Group I

		Number of function evaluations				
Function	n	A	M	R	B	C
1	—	10	10	9	8	9
2	1	9	9	7	8	7
	2	10	10	8	9	8
	3	11	11	9	10	9
	4	12	12	10	10	10
3	1	10	9	8	8	9
	5	10	10	9	9	8
	10	9	9	9	9	8
4	1	9	10	8	9	9
	5	10	10	9	9	10
	10	11	11	11	10	11
5	1	10	10	8	9	9
	4	9	9	9	8	8
	8	7	7	8	7	8
6	1	9	9	8	9	9
	5	9	9	9	9	9
	10	10	10	10	9	10
Total		165	165	149	150	151

Table II. Test Functions of Group II

			Number of function evaluations				
a	b	n	A	M	R	B	C
1	0	3	11	12	11	15	12
		5	10	10	10	14	12
		9	10	13	11	16	12
		19	10	13	13	16	12
0	10^{-4}	3	21	26	17	26	21
		5	22	26	18	27	23
		9	23	27	19	25	24
		19	23	27	19	24	24
1	10^{-4}	3	11	12	11	14	12
		5	10	10	10	14	11
		9	10	10	11	16	11
		19	10	13	13	16	11
Total			171	199	163	223	185

Tables III and IV. Test Functions of Groups III and IV

	Number of function evaluations				
n	A	M	R	B	C
			Table III		
3	117	151	91	147	118
5	206	149	163	122	207
7	293	161	206	138	294
9	380	160	196	137	381
19	802	179	206	141	759
25	1320	159	174	123	961
Total	3118	959	1036	808	2720
			Table IV		
	>5000	27	23	18	969

```
Boolean procedure zeroin(x, y, fx, tolx);
real x, y, fx, tolx;
begin integer ext;
    real c, fc, b, fb, a, fa, d, fd, fdb, fda, w, mb,
    tol, m, p, q;
    b:= x; fb:= fx; a:= x:= y; fa:= fx;
interpolate: c:= a; fc:= fa; ext:= 0;
extrapolate: if abs(fc) < abs(fb) then
    begin if c  ≠ a then begin d:= a; fd:= fa end;
        a:= b; fa:= fb; b:= x:= c; fb:= fc; c:= a; fc:= fa
    end interchange;
    tol:= tolx; m:= (c + b) × 0.5; mb:= m - b;
    if abs(mb) > tol then
    begin if ext > 2 then w:= mb else
        begin tol:= tol × sign(mb);
            p:= (b - a) × fb; if ext ≤ 1 then
            q:= fa - fb else
            begin fdb:= (fd - fb) / (d - b);
                fda:= (fd - fa) / (d - a);
                p:= fda × p; q:= fdb × fa - fda × fb
            end; if p < 0 then
            begin p:= -p; q:= -q end;
            w:= if p × 1 = 0 ∨ p ≤ q × tol then tol else
            if p < mb × q then p / q else mb
        end; d:= a; fd:= fa; a:= b; fa:= fb;
        x:= b:= b + w; fb:= fx;
        if (if fc ≥ 0 then fb ≥ 0 else fb ≤ 0) then
        goto interpolate else
        begin ext:= if w = mb then 0 else ext + 1;
            goto extrapolate
        end
    end; y:= c;
    zeroin:= if fc ≥ 0 then fb < 0 else fb ≥ 0
end zeroin;
```

Fig. 1

placing each linear extrapolation step by an inverse quadratic interpolation step (in algorithm B, see [2]) or a rational extrapolation step (in algorithm C, see [1]). Hence in algorithms R, B, and C we save roughly 10 percent of the number of function evaluations at the cost of slightly more complicated calculations.

From Table II we see that algorithms R, C, and M are better than algorithm B for finding a simple zero of a function with a high order inflexion point at or near the zero.

Finally, Tables III and IV show clearly that algorithm A and also algorithm C are not efficient for calculating multiple zeros. They may cause a computer program to run out of time very quickly.

6. CONCLUSIONS

From the results given in Section 5 it is obvious that algorithms A and C are not efficient for practical use on a computer if the multiplicity of the zero is not known in advance.

```
Boolean procedure zeroinrat(x, y, fx, tolx);
real x, y, fx, tolx;
begin integer ext; boolean first;
    real b, fb, a, fa, d, fd, c, fc, fdb, fda, w,
    mb, tol, m, p, q;
    b:= x; fb:= fx; a:= x:= y; fa:= fx; first:= true;
interpolate: c:= a; fc:= fa; ext:= 0;
extrapolate: if abs(fc) < abs(fb) then
    begin if c ≠ a then begin d:= a; fd:= fa end;
        a:= b; fa:= fb; b:= x:= c; fb:= fc; c:= a; fc:= fa
    end interchange;
    tol:= tolx; m:= (c + b) × .5; mb:= m - b;
    if abs(mb) > tol then
    begin if ext > 3 then w:= mb else
        begin tol:= tol × sign(mb);
            p:= (b - a) × fb; if first then
            begin q:= fa - fb; first:= false end else
            begin fdb:= (fd - fb) / (d - b);
                fda:= (fd - fa) / (d - a);
                p:= fda × p; q:= fdb × fa - fda × fb
            end; if p < 0 then
            begin p:= -p; q:= -q end;
            if ext = 3 then p:= p × 2;
            w:= if p × 1 = 0 v p ≤ q × tol then tol else
            if p < mb × q then p / q else mb
        end; d:= a; fd:= fa; a:= b; fa:= fb;
        x:= b:= b + w; fb:= fx;
        if (if fc ≥ 0 then fb ≥ 0 else fb ≤ 0) then
        goto interpolate else
        begin ext:= if w = mb then 0 else ext + 1;
            goto extrapolate
        end
    end; y:= c;
    zeroinrat:= if fc ≥ 0 then fb ≤ 0 else fb ≥ 0
end zeroinrat;
```

Fig. 2

Although in most cases the results of algorithm B are slightly better than those of algorithm M, this is only due to the use of a more complicated formula in roughly 30 percent of the iteration steps. Moreover, there are examples (see Table II) for which algorithm M requires fewer function evaluations than algorithm B. So for rather simple functions, whose evaluation is cheap with respect to the calculations performed in one iteration step of algorithm M, we recommend the use of algorithm M; we also recommend the use of algorithm M because the upper bound of the number of function evaluations needed is better than for algorithm B (see Theorem 3.3.1). Algorithm R is to be preferred for more expensive functions, because of the higher asymptotic order of convergence of the interpolation formula used in this algorithm (see Theorem 4.3.2). This statement is affirmed by the numerical results given in Section 5. For functions having poles near the zero we also advise the use of algorithm R, because of the special character of the interpolating function used in this algorithm.

APPENDIX. ALGOL 60 PROCEDURES

In this Appendix we give the text of two Algol 60 procedures (Figures 1 and 2), implementing algorithms M and R, defined in Sections 3 and 4.

The heading of the procedure implementing algorithm M reads:

Boolean procedure *zeroin*(*x*,*y*,*fx*,*tolx*);
real *x*,*y*,*fx*,*tolx*;

The heading of the procedure implementing algorithm R reads:

Boolean procedure *zeroinrat*(*x*,*y*,*fx*,*tolx*);
real *x*,*y*,*fx*,*tolx*;

The meaning of the formal parameters is:

x, y: real variables;
entry: the endpoints of the interval J_1 (see paragraph 2.4.1);
exit: if the value of the procedure identifier is **true**, then the values of x and y satisfy (2.2.1);

fx: real expression depending on x; the actual value of *fx* should be equal to the function value at the point given by the actual value of x;

tolx: real expression depending on x; the actual value of *tolx* should be equal to the value of the tolerance function at the point given by the actual value of x.

The procedure identifier will have the value **true** on exit if two argument values x and y are found which satisfy (2.2.1); otherwise the value of the procedure identifier will be **false** on exit. The last case can only occur if, on entry, the values of x and y do not satisfy $f(x) \times f(y) \leq 0$. Note that in the procedures we have written "**if** $p \times 1 = 0 \vee$" instead of "**if** $p = 0 \vee$." This is done because of the poor arithmetic of the Cyber 73 for values around the smallest positive representable number. On this computer, it can occur that the Boolean expression $p = 0$ has the value **false**, while the expressions $p/1$ and $p \times 1$ have the value 0. So replacing the expression $p = 0$ by $p \times 1 = 0$ removes the difficulty, at least in the cases we checked.

NOTE ADDED IN PROOF

Instead of "$p \times 1 = 0$," it would be preferable to write "$p \leq$ dwarf," where dwarf is a machine constant roughly equal to the smallest positive representable number and, more precisely, defined such that underflow and anomalies of the kind stated above can occur only for numbers which are smaller in magnitude. We chose the (machine-dependent) formulation $p \times 1 = 0$, because there is not yet an international agreement on the definition of dwarf and/or related machine constants.

REFERENCES

1. ANDERSON, N., AND BJÖRCK, A. A new high order method of regula falsi type for computing a root of an equation. *BIT 13* (1973), 253–264.
2. BRENT, R.P. An algorithm with guaranteed convergence for finding a zero of a function. *Computer J. 14* (1971), 422–425.
3. DEKKER, T.J. Finding a zero by means of successive linear interpolation. In *Constructive Aspects of the Fundamental Theorem of Algebra,* B. Dejon and P. Henrici, Eds., Wiley Interscience, London, 1969.
4. DOWELL, M., AND JARRATT, P. A modified regula falsi method for computing the root of an equation. *BIT 11* (1971), 168–174.
5. JARRATT, P., AND NUDDS, D. The use of rational functions in the iterative solution of equations on a digital computer. *Computer J. 8* (1965), 62–65.

Received March 1974