An Efficient Derivative-Free Method for Solving Nonlinear Equations

D. LE
University of New South Wales, Australia

An algorithm is presented for finding a root of a real function. The algorithm combines bisection with second and third order methods using derivatives estimated from objective function values. Global convergence is ensured and the number of function evaluations is bounded by four times the number needed by bisection. Numerical comparisons with existing algorithms indicate the superiority of the new algorithm in all classes of problems.

Categories and Subject Descriptors: G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*convergence*

General Terms: Algorithms

Key Words and Phrases: Root finding

1. INTRODUCTION

One of the most frequently occurring problems in scientific work is to locate a real root α of a nonlinear equation

$$f(x) = 0. \tag{1}$$

In rare cases, it may be possible to express the root in closed form or to obtain the exact value of α such as in factorable polynominal. In general, however, we can hope to obtain only approximate solutions by relying on iterative methods. The function $f(x)$, may not be given explicitly, may not be differentiable or the derivatives may be difficult to compute, so a method which uses only computed values of f is generally more desirable.

Some of the more classical numerical methods for solving nonlinear equations without using derivatives include bisection, secant, and regula falsi (see [13] for a good survey). Many of these methods, such as the secant method, are fast in general but can be extremely slow for certain classes of functions and are likely to fail if the starting value is not sufficiently close to α. On the other hand, bisection is a safe method and in fact is optimal in a minimax sense, but its rate of convergence is relatively slow. The difficulty is to devise an algorithm that has

Author's address: Energy Systems Analysis Group, CSIRO Division of Energy Technology, Lucas Heights Research Laboratories, Private Mail Bag 7, Sutherland, New South Wales, 2232, Australia.

a high rate of convergence for well-behaved functions, yet can still guarantee to converge in a small number of function evaluations for all arbitrary functions.

Over the past few years, several authors have published efficient derivative-free algorithms which are based on the use of a method with superlinear convergence, but using the bisection method when necessary to retain the bracketing property. Some successful algorithms developed along this line include those of Dekker [6], Brent [2], and Bus and Dekker [4]. Various attempts have also been made at modifying the regula falsi method to improve its order of convergence, for example see [1], [7], and [8]. All of the above mentioned algorithms guarantee global convergence as well as having high asymptotic order of convergence. Among these algorithms, the two published by Bus and Dekker have the lowest upper bounds on the number of function evaluations required, namely $4n_b$ and $5n_b$ respectively, where n_b is the number of function evaluations needed by bisection. Brent's algorithm can require up to $(n_b + 1)^2 - 2$ function evaluations while those of Dekker [6], Dowell and Jarratt, [7] and Anderson and Bjorck [1] may require a prohibitively large number of function evaluations for certain classes of functions.

In a recent paper, Le [10] proposed three different algorithms, called LZ1, LZ2 and LZ3, for approximating a zero of an arbitrary function. Two completely new approaches were introduced in these algorithms. The first approach is based on the concept of cushion interpolation while the second involves bisecting the reduced interval limited by two most recent secants. The principal advantage of these algorithms is their extremely small upper bounds on the number of function evaluations required. Numerical results seem to indicate that LZ1–LZ3 could outperform existing methods for functions with multiple zeros while remaining comparable for general functions. However, no definite conclusions could be drawn because the stopping criteria used for LZ1–LZ3 were different from those used by various authors for other methods. In this paper we present a new algorithm, called LZ4, based on the combination of bisection with second and third order methods employing derivatives estimated from objective function values. The approach used in this algorithm is entirely different from those employed in LZ1–LZ3. The algorithm has a very high rate of convergence yet only requires at most $4n_b$ function evaluations. This bound is only slightly higher than those of LZ1–LZ3, is equal to that of Bus and Dekker [4], and compares favorably with all other existing methods. Section 2 contains a detailed discussion of this algorithm and a numerical comparison with existing algorithms is given in Section 3.

2. ALGORITHM LZ4

The new algorithm LZ4 uses a high-order method as its basic process but occasionally resorts to bisection to retain the bracketing property or to speed up convergence when the root appears to be multiple.

One way of obtaining a high-order iterative method for solving (1) is to use an approximating function based on Taylor polynomials. Assume that in a neighborhood of α, the function $y = f(x)$ has a unique inverse $x = \phi(y)$. If ϕ is sufficiently differentiable, the Taylor series expansion of $\phi(y)$ about a point y_n

is given by

$$x = \phi(y) = x_n + \sum_{j=1}^{m+1} \frac{(y - y_n)^j}{j!} \phi^{(j)}(y_n) + \frac{(y - y_n)^{m+2}}{(m + 2)!} \phi^{(m+2)}(\eta),$$

where x_n is the nth iterate, $\eta \in [y, y_n]$ and $\phi^{(j)}(y_n)$ is the jth derivative of ϕ at y_n. By replacing $x = \alpha = \phi(0)$ and dropping the error term, we arrive at the iteration formula

$$x_{n+1} = x_n + \sum_{j=1}^{m+1} \frac{(-1)^j}{j!} y_n^j \phi^{(j)}(y_n).$$

By expressing $\phi^{(j)}$ in terms of the derivatives of f, we have for $m = 0$ the Newton-Raphson iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{2}$$

For $m = 1$, the resulting iterative method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f''(x_n)[f(x_n)]^2}{2[f'(x_n)]^3}, \tag{3}$$

is of at least third order provided $f'(\alpha) \neq 0$.

Alternatively we can construct a Taylor polynomial of $f(x)$ at x_n,

$$p_j(x) = f(x_n) + \frac{(x - x_n)}{1!} f'(x_n) + \ldots + \frac{(x - x_n)^j}{j!} f^{(j)}(x_n).$$

Now let x_{n+1} be a root of $p_j(x) = 0$. For $j = 1$ we again have the familiar Newton-Raphson method. For $j = 2$, the following approximation formula is obtained.

$$x_{n+1} = x_n + \frac{-f'(x_n) \pm [f'(x_n)^2 - 2f(x_n)f''(x_n)]^{1/2}}{f''(x_n)}, \tag{4}$$

where the + or − sign is taken according to whether $f'(x_n)$ is positive or negative. This formula is at least third order if $f'(\alpha) \neq 0$ and $f''(\alpha) \neq 0$.

Although iteration formulas (3) and (4) are of the same order 3, it seems that (4) is more difficult to use because of the need to calculate a square root and to choose between positive and negative signs at each iteration. Furthermore, numerical study also shows that the algorithm LZ4 using equation (4) as its basic process is slightly inferior to that with equation (3), (except for quadratic functions) and accordingly the iteration formula (3) is preferable. It should be noted that in LZ4, all derivative terms required by the formulas developed above are replaced by their divided difference approximations using only objective function values (see step 4.3 in the description of the algorithm).

The description of the algorithm LZ4 is given below.

Let f be a real continuous function of one real variable, defined on the interval $[a, b]$ with $f(a) \cdot f(b) \leq 0$. The purpose of the algorithm LZ4 is to find an approximation $\hat{\alpha}$ to a zero α of f to within the required precision by using only

function evaluations. Depending on the context, an approximate solution may then mean either a point $\hat{\alpha}$ for which (1) is approximately satisfied, i.e., for which $|f(\hat{\alpha})|$ is small, or a point $\hat{\alpha}$ which is close to the solution α of (1). Thus there are two possible stopping criteria, the first relates to the objective function value while the second to the argument value. In LZ4, for any given small constant $\epsilon_y \geq 0$, the first stopping criterion is satisfied if a point $\hat{\alpha}$ is found with

$$|f(\hat{\alpha})| \leq \epsilon_y. \tag{5}$$

The second stopping criterion can be precisely stated by defining a positive tolerance function $\delta(x) = \epsilon_1|x| + \epsilon_2$ where ϵ_1 represents a relative tolerance for large $|x|$ and ϵ_2 an absolute tolerance. The algorithm LZ4 then terminates via the second stopping criterion when two distinct real numbers $x1$ and $x2$ are found satisfying

$$f(x1) \cdot f(x2) \leq 0 \qquad \text{and} \qquad |x1 - x2| \leq 2 \cdot \delta(\hat{\alpha}), \tag{6}$$

where $\hat{\alpha}$ here is chosen to be whichever of $x1$ or $x2$ corresponds to the smaller objective function value. Since f is continuous, the first condition of (6) ensures that there exists a zero in the closed interval $[x1, x2]$ while the second condition yields $\hat{\alpha}$ as the current best approximation of α and that the required tolerance has been reached. It should be noted that LZ4 will stop iterating if either one of the above stopping criteria is satisfied. Thus it is possible to set $\epsilon_y = 0$ so that only the second criterion comes into play.

At the beginning of a typical iteration of algorithm LZ4, three distinct points $x1$, $x2$, and $x3$ are available such that

$$f(x1) \cdot f(x2) \leq 0, \qquad f(x2) \cdot f(x3) \geq 0, \qquad x2 \in [x1, x3]. \tag{7}$$

These conditions state that there exists a zero α of f in the closed interval $[x1, x2]$, and that $x2$ and $x3$ lie on the same side of α. The points $x1$, $x2$, and $x3$ change from iteration to iteration, but there should be no confusion if we omit iterative subscripts. From the given interval $[a, b]$, the first set of points satisfying (7) is obtained as follows: Let $x1 = a$, $x2 = (a + b)/2$; if $f(x1) \cdot f(x2) > 0$ then $x3 = x1$, $x1 = b$ otherwise let $x3 = b$. Other necessary initializations are: Let int = 1, $d = 2 \cdot |a - b|$, $d1 = 2 \cdot d$, and $d2 = 2 \cdot d1$.

An iteration of LZ4 consists of the following steps:

(1) Let $z = x1 + (x2 - x1)/2$ and $d = |x1 - x2|$. Let $d1$ be the last value of d, $d2$ the last value of $d1$ and $d3$ the last value of $d2$. Moreover, let

$$\begin{aligned} u &= x1, \quad && \text{if } |f(x1)| < |f(x2)|, \\ &= x2, && \text{otherwise.} \end{aligned}$$

(This condition yields u as the current best approximation of α).

(2) Test for convergence. If $|f(u)| \leq \epsilon_y$ or $d \leq 2\delta(u)$ then the algorithm terminates with $\hat{\alpha} = u$. When high order convergence has set in, the current best estimate u may be much better than the bisection point of the remaining interval $[x1, x2]$, thus we prefer to return u as the approximation to α.

(3) Estimating derivatives
Let

$$\begin{aligned} c &= x1, && \text{if int} = 1 \text{ (interpolation)},\\ &= \text{previous value of } x3, && \text{if int} = 0 \text{ (extrapolation)}, \end{aligned}$$

where int is set equal to 1 initially and is defined clearly in step 4.5. It is thus seen that $\alpha \in [x2, x3, c]$ when int = 1, and as a consequence an interpolation is required to estimate α. An extrapolation using $x2$, $x3$ and c is necessary when int = 0. The three distinct points $x2$, $x3$ and c are then used to give estimates of the first and second derivatives of f at u.
Let

$$e = x2 + (c - x2)/2$$

and

$$ge = f[x2, c] = (f(x2) - f(c))/(x2 - c),$$

where $f[x2, c]$ denotes the first divided difference of f at $x2$ and c.

Define

$$h = 2(f[x2, x3] - ge)/(x3 - c), \tag{8}$$

so that h is twice the second divided difference of f at $x2$, $x3$ and c, and approximates the second derivative of the function at the point $p = (c + 2 \cdot x2 + x3)/4$. If h can be assumed constant in the neighborhood of p, then the derivative at any point x in this region can be estimated by

$$g(x) = ge + h(x - e). \tag{9}$$

(It is noted here that this method can obviously be expanded to include more points to estimate higher divided difference derivatives for higher order approximation formulas. However, in general there seems to be little gain in using data at more than two or three points.)

(4) Determine a new point w for function evaluation. The general strategy applied in LZ4 is that w is first calculated by formula (3); if $w \notin [z, u]$ then the Newton-Raphson method is employed to obtain a new value of w, if again $w \notin [z, u]$ then we use bisection.

(4.1) If $d > 0.595 \cdot d3$ or $(g(x1) \cdot g(x3) \leq 0$ and int = 1) then bisection is used, that is let $w = z$. The first condition determines the upper bound on the number of function evaluations while the second speeds up convergence for ill-behaved problems.

(4.2) If $|2 \cdot f(u) \cdot (g(u))^2 + h \cdot (f(u))^2| \geq |d \cdot (g(u))^3|$ then go to 4.3; otherwise let $w = u - f(u)/g(u) - h(f(u))^2/(2(g(u))^3)$ which is the estimate given by formula (3).
If $(w - u) \cdot \text{sign}\,(w - z) \leq 0$ then go to step 4.4; else execute 4.3.

(4.3) If $|f(u)| \geq |d \cdot g(u)/2|$, then use bisection; otherwise let $w = u - f(u)/g(u)$, which is the Newton-Raphson step (formula(2)), and go to 4.4.

(4.4) If $|w - u| < \delta(u)$ then let $w = u + \delta(u) \cdot \text{sign}\,(z - u)$. It is seen that care is taken to avoid overflow or division by zero when computing the

new point w by formulas (2) and (3). Although the first condition of step 4.2 already restricts $|w - u| < |z - u|$, there is no guarantee that w must lie in the interval $[z, u]$. If $|g(u)|$ is small and $|h|$ large, then the second derivative of the inverse function $|\phi^{(2)}| = |h/(g(u))^3|$ becomes very large resulting in an exceptionally strong curvature at u which in turn causes w to fall outside $[z, u]$. This explains the need to include the test at the end of step 4.2. The problem does not exist for formulas (2) and (4) since $|w - u| < |z - u|$ also means $w \in [z, u]$ in these cases.

(5) Reduce the search interval. Let the interpolation indicator int = 1.

(5.1) If $f(w) \cdot \text{sign}(f(x1)) < 0$ then go to 5.2; otherwise if $d \leq |x3 - w|$ then let $x3 = x1$, $x1 = x2$ and $x2 = w$; else let $x1 = w$; go to 4.1.

(5.2) If $d \leq |x3 - w|$ then let $x3 = x2$, $x2 = w$; otherwise let $c = x3$, $x3 = x2$, $x2 = w$ and int = 0; go to 4.1.

It is obvious from the definition of the algorithm, particularly step (4.4), that the new argument value w in each iteration is distinct from any existing point with the mutual distance bounded below by $\delta(u)$. Let I_i, for iteration $i = 1, 2, \ldots, n$ denotes the closed interval whose endpoints are $x1_i$ and $x2_i$. Then from the relation $f(x1_i) \cdot f(x2_i) \leq 0$ and the operations of steps (4) and (5), it follows immediately that I_i contains a zero α of f and $I_1 \supset I_2 \supset I_3 \ldots \supset I_i$. This shows that LZ4 will converge to a zero α of f as $i \to \infty$.

The maximum number of function evaluations needed by LZ4 is dictated by the first condition of step (4.1). With d_i denoting the length of I_i, the first condition of (4.1) can be presented in a general form as: "If $d_i > t \cdot d_{i-k}$ then do bisection," where k can be any positive integer. Assume for the moment that $t = 0.5$, then it is clear that the length of I_i is smaller than half the length of I_{i-k-1}. It follows therefore, that the number of function evaluations is bounded above by $(k + 1)n_b$. However, as that upper bound is based on the worst case, there is no requirement that t must be kept at 0.5 for other cases. In fact, t should be increased as much as possible to reduce the chance of performing bisection unnecessarily. In order to still guarantee the above bound of $(k + 1)n_b$, it can be shown that t must be limited by

$$(t)^{k+1} \leq (0.5)^k.$$

Therefore, the value of t can be as high as 0.595 if the required maximum number of function evaluations is $4n_b$. It should be mentioned that the performance of the algorithm is substantially reduced when decreasing this bound to $3n_b$ as bisection seems to be performed excessively, while no gain can be achieved by increasing the bound to $5n_b$.

The most complicated step in the new algorithm is step (4.2) which is itself hardly more complicated than the interpolation steps used in the algorithms of Brent [3] or of Bus and Dekker [4]. Moreover, slight differences in the complexities between the algorithms should be negligible in terms of computation time since only problems with expensive function evaluations would normally necessitate a proper selection of algorithms. This is one of the reasons why the number of function evaluations, instead of computation time, is often used as the criterion for comparison as it is less dependent on the computer and compiler systems or the programmer's coding ability.

It should be stressed here that LZ4 is not of third order because of the use of finite-difference approximated derivatives and also due to the involvement of the bisection as well as the finite-difference Newton-Raphson steps.

3. NUMERICAL STUDIES

It would be interesting to see the rate of convergence of the basic idea underlying the algorithm, namely the use of a third order method (formula (3)) with derivatives replaced by divided difference estimates. A test function obtained from Ralston and Rabinowitz [11] is used for this purpose

$$f(x) = \sin(x) - \frac{x}{2},$$

which is strictly concave in the interval under study $[\pi/2, \pi]$. The root is computed to five correct decimals and the sequences of estimates produced by three methods are shown in Table I. It should be noted that the values listed under LZ4 are not produced by the entire algorithm LZ4 as described in Section 2 above, but are the results of only formula (3) with approximated derivatives. As LZ4 requires 3 points to start with, the third point is chosen to be the midpoint of the interval $[\pi/2, \pi]$.

Although the total number of function evaluations required by the Newton-Raphson method is 5 compared to 6 by LZ4, of particular interest is the number of iterations which directly relate to the rate of convergence. It is seen that LZ4 requires one less iteration than the Newton-Raphson method which in turn needs one less iteration than the secant method. However, it is difficult to compare these three methods directly as they require different numbers of starting points.

Furthermore, as the performance of an iterative method depends greatly on the position(s) of the starting point(s), it might be expected that the rates of convergence of the three methods would be different for different values or orderings of the starting points. Surprisingly, in the following sensitivity analysis it turns out that the relative performance of these methods on this particular function seem to remain constant for all combinations of starting points tested so far. The number of iterations required by the Newton-Raphson method remains 4 when the starting point is altered to $3\pi/4$ and π. Similarly, the ordering of the starting points of the secant method has no effect on the number of iterations required by that method although the sequence of estimates is different. LZ4 also converges in 3 iterations when the third starting point is changed from $3\pi/4$ to the linear interpolation point using $\pi/2$ and π.

It should be mentioned that in this exercise the estimates h and $g(u)$ given by (8) and (9) converge to the real values of $f''(u)$ and $f'(u)$ as u approaches α although at a slower rate than that of u. Thus we can see that it is only necessary to estimate f'' and f' to very few significant digits to compute α to full single-precision accuracy.

Below are the results of a series of numerical comparisons between seven algorithms for solving nonlinear equations:

Algorithm LZ4, defined in Section 2;
Algorithm B, published by Brent [2];
Algorithms A, M and R, described in Bus and Dekker [4];

Table I. Sequences of Estimates to the Root of $f(x) = \sin(x) - x/2$

Starting point(s)	Secant $(\pi/2, \pi)$	Newton–Raphson $(\pi/2)$	LZ4 $(\pi/2, 3\pi/4, \pi)$	Iteration number
	1.75960	2.00000	1.85467	1
	1.84420	1.90100	1.89587	2
	1.90011	1.89551	1.89549	3
	1.89535	1.89549		4
	1.89549			5

Algorithm C, published by Anderson and Bjorck [1];
Algorithm D, a modified Davidenko-Broyden continuation method published by Swift and Lindfield [12].

All calculations have been carried out in single precision with 48 bits accuracy on the CDC Cyber 171 at the University of New South Wales. It should be mentioned that some exponent underflows have inevitably occurred during evaluations of function 11 for $n = 25$ and function 12.

As most of the performance data of other algorithms have been compiled from other authors' works (except those of algorithm B for function 2 and 3 where they have been obtained by the author), it is essential to use the same stopping criteria for LZ4 in order to achieve a fair and direct comparison. The following list shows all stopping criteria used; the governing criterion is basically equation (6), but different tolerance functions $\delta(x)$ have been chosen by different authors:

(a) $\delta(x) = 2 \times 16^{-7}|x| + 10^{-10}$; see [2].
(b) $\delta(x) = 10^{-8}(2|x| + 1)$; see [10].
(c) $\delta(x) = 10^{-7}(2|x| + 1)$; see [10].
(d) $\delta(x) = 10^{-14}(|x| + 1)$; see [4].
(e) $\delta(x) = 3 \times 10^{-8}(|x| + 1)$; see [1].
(f) $\delta(x) = 0.5 \times 10^{-8}(|x| + 1)$; see [12].

The test functions chosen for this numerical study are of varying difficulty and characteristics. They are the same as those used in [10] and many of them are obtained from various authors' works [2], [7], and [12].
The first set of test functions is supplied with an interval containing the zero:

(1) $f(x) = -2 \sum_{i=1}^{20} (2i - 5)^2/(x - i^2)^3$
in the interval $[n^2 + 10^{-9}, (n + 1)^2 - 10^{-9}]$ for $n = 1(1)19$
(2) $f(x) = axe^{bx}$ in the interval $[-9, 31]$, where
(i) $a = -40$ and $b = -1$
(ii) $a = -100$ and $b = -2$
(iii) $a = -200$ and $b = -3$
(3) $f(x) = x^n - a$, where
(i) $a = 0.2$ and $n = 4, 6, 8, 10, 12$ in the interval $[0, 5]$
(ii) $a = 1$ and $n = 4, 6, 8, 10, 12$ in the interval $[0, 5]$
(iii) $a = 1$ and $n = 8, 10, 12, 14$ in the interval $[-0.95, 4.05]$
(4) $f(x) = \sin(x) - 0.5$ in the interval $[0, 1.5]$
(5) $f(x) = 2xe^{-n} - 2e^{-nx} + 1$
in the interval $[0, 1]$ and $n = 1, 2, 3, 4, 5, 15, 20$

(6) $f(x) = (1 + (1 - n)^2)x - (1 - nx)^2$
in the interval [0, 1] and $n = 1, 2, 5, 10, 15, 20$
(7) $f(x) = x^2 - (1 - x)^n$
in the interval [0, 1] and $n = 1, 2, 5, 10, 15, 20$
(8) $f(x) = (1 + (1 - n)^4)x - (1 - nx)^4$
in the interval [0, 1] and $n = 1, 2, 4, 5, 8, 15, 20$
(9) $f(x) = (x - 1)e^{-nx} + x^n$
in the interval [0, 1] and $n = 1, 5, 10, 15, 20$
(10) $f(x) = (nx - 1)/((n - 1)x)$
in the interval [0.01, 1] and $n = 2, 5, 15, 20$
(11) $f(x) = x^n$ in the interval $[-1, 10]$ and $n = 3, 5, 7, 9, 19, 25$
(12) $f(x) = 0$, if $x = 0$
$= x \cdot \exp(-x^{-2})$, otherwise in the interval $[-1, 4]$.

The second set of test functions is supplied with only a single starting point. These sample problems were taken from [12] and are listed below for clarity:

(13) $f(x) = 2xe^{-n} - 2e^{-nx} + 1$ with starting value 0. and $n = 5$
(14) $f(x) = (1 + (1 - n)^4)x - (1 - nx)^4$
with starting value 0. and $n = 5$
(15) $f(x) = x^n$ with starting value -1. and $n = 3, 5, 7$
(16) $f(x) = (e^{-nx} - x - 0.5)/x^n$ with starting value 0.1185 and $n = 5$
(17) $f(x) = x^{-1/2} - 2 \log_{10}(nx^{1/2}) + 0.8$
with starting value 0.001 and $n = 5 \times 10^3, 5 \times 10^7$.

A simple interval locating procedure given by Swift and Lindfield has been used to supply an interval bracketing the zero for both algorithms B and LZ4. The initial stepsize for this procedure was set at 0.001 which is the same as that used by Swift and Lindfield. It has been noted in [10] that this interval locating procedure may diverge or fail to find an interval with end points of opposite sign function values. However, such a problem fortunately does not exist for the above test functions with the corresponding starting points.

The results for the first set of test functions are given in Tables II to VII while those for the second set are shown in Table VIII. The tables list the number of function evaluations needed by the various algorithms to find a root of the given function within the required precision.

It can be seen from all tables that the new algorithm LZ4 is markedly superior to all existing algorithms for a wide range of test functions. The saving on the number of function evaluations made by LZ4 compared to other algorithms can be as high as 5 function evaluations for simple zero functions and up to several hundreds for multiple zero functions. Although the theoretical maximum number of function evaluations for convergence required by LZ4 can be as high as $4n_b$, it has always been only slightly slower than bisection in practice. From the second part of Table V, Algorithm C appears to perform better than LZ4 and in fact better than all other algorithms on function 10. However, the rapid convergence of Algorithm C on this particular function can be explained by its relation to hyperbolic interpolation and thus cannot be seen to represent the general trend.

Because of the use of different stopping criteria, the results of LZ1–LZ3 were not included in the comparison study above. Although a direct comparison

Table II. Number of Function Evaluations for Function 1

	Stopping criterion (a)	
n	B	LZ4
1	14	10
2	8	10
3	14	9
4	12	9
5	12	9
6	11	9
7	11	9
8	11	9
9	10	9
10	10	9
11	10	9
12	10	9
13	10	9
14	10	9
15	10	9
16	10	9
17	10	9
18	9	9
19	9	9
Total	201	173

Table III. Number of Function Evaluations for Function 2

			Stopping criterion (b)	
Interval	a	b	B	LZ4
[−9, 31]	− 40	−1	16	12
	−100	−2	18	14
	−200	−3	19	14
Total			53	40

Table IV. Number of Function Evaluations for Function 3

			Stopping criterion (c)	
Interval	a	n	B	LZ4
[0, 5]	0.2	4	13	10
		6	15	10
		8	16	12
		10	16	11
		12	16	11
[0, 5]	1.	4	14	9
		6	14	11
		8	13	11
		10	15	11
		12	16	11
[−0.95, 4.05]	1.	8	14	10
		10	14	10
		12	15	11
		14	16	11
Total			207	149

Table V. Number of Function Evaluations for Functions 4–10

Function	Interval	n	Stopping criterion (d)						Stopping criterion (e)		
			A	M	R	B	C	LZ4	B	C	LZ4
4	[0, 1.5]	—	10	10	9	8	9	7	8	7	7
5	[0, 1]	1	9	9	7	8	7	7	7	6	7
		2	10	10	8	9	8	7			
		3	11	11	9	10	9	8			
		4	12	12	10	10	10	9			
		5							10	10	8
		15							11	12	10
		20							11	12	10
6	[0, 1]	1	10	9	8	8	9	7			
		2							9	8	6
		5	10	10	9	9	8	8	8	8	6
		10	9	9	9	9	8	5			
		15							7	7	6
		20							7	6	6
7	[0, 1]	1	9	10	8	9	9	8			
		2							3	3	2
		5	10	10	9	9	10	9	8	8	8
		10	11	11	11	10	11	9			
		15							10	11	9
		20							12	11	10
8	[0, 1]	1	10	10	8	9	9	8			
		2							10	9	8
		4	9	9	9	8	8	8			
		5							7	8	7
		8	7	7	8	7	8	7			
		15							6	7	6
		20							6	6	6
9	[0, 1]	1	9	9	8	9	9	9	8	7	7
		5	9	9	9	9	9	9	8	8	8
		10	10	10	10	9	10	10	8	9	9
		15							11	9	9
		20							12	10	10
10	[0.01, 1]	2							5	6	8
		5							12	7	9
		15							11	6	11
		20							13	6	11
Total			165	165	149	150	151	135	228	207	204

Table VI. Number of Function Evaluations for Function 11

Interval	n	Stopping criterion (d)					
		A	M	R	B	C	LZ4
[−1, 10]	3	117	151	91	147	118	57
	5	206	149	163	122	207	79
	7	293	161	206	138	294	50
	9	380	160	196	137	381	50
	19	802	179	206	141	759	50
	25	1320	159	174	123	961	41
Total		3118	959	1036	808	2720	327

Table VII. Number of Function Evaluations for Function 12

	Stopping criterion (d)					
Interval	A	M	R	B	C	LZ4
[−1, 4]	>5000	27	23	18	969	11

Table VIII. Number of Function Evaluations for Problems with 1 Starting Point

			Stopping criterion (f)		
Function	Starting value	n	D	B	LZ4
13	0.	5	12	14	13
14	0.	5	8	7	6
15	−1.	3	130	126	40
	−1.	5	141	86	21
	−1.	7	146	67	21
16	0.1185	5	20	13	11
17	0.001	5×10^3	18	13	12
	0.001	5×10^7	13	10	9
Total			488	336	133

could not be made, it appears that the new algorithm should be preferrable to LZ1–LZ3 for general functions due to its high rate of convergence. However when multiple zeros may exist, the algorithms LZ1–LZ3 could be chosen because of their smaller bounds on the number of function evaluations required, although numerical results show that LZ4 also performs extremely well for this class of functions.

It remains to be seen how LZ4 performs in cases where the function is hard to evaluate and subjected to rounding or other errors, for example, when the solutions of differential equations are involved. However, one might expect that the new algorithm would also perform well on those functions judging from its outstanding performance on functions with multiple zeros where rounding errors in evaluating f are quite severe.

4. CONCLUSIONS

A new algorithm for solving nonlinear equations has been presented. The algorithm features a combination of bisection with second and third order methods using derivatives estimated from objective function values. The maximum number of function evaluations required by the algorithm is bounded by four times the number needed by bisection. Numerical comparisons indicate that this algorithm is much faster than existing algorithms on both well- and ill-behaved functions and for both simple zeros as well as zeros with high multiplicity.

ACKNOWLEDGMENTS

The author wishes to thank Dr. G. Smith for his encouragement and stimulating discussions. Constructive comments made by the referees are also gratefully acknowledged.

REFERENCES

1. ANDERSON, N., AND BJORCK, 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. *Comput. J. 14* (1971), 422–425.
3. BRENT, R. P. *Algorithms for Minimisation Without Derivatives.* Prentice Hall, Englewood Cliffs, N.J. 1973.
4. BUS, J. C. P., AND DEKKER, T. J. Two efficient algorithms with guaranteed convergence for finding a zero of a function, *Trans. Math. Softw. 4* (Dec. 1975), 330–345.
5. DAHLQUIST, G., AND BJORCK, A. *Numerical Methods.* Prentice Hall, Englewood Cliffs, N.J., 1974.
6. 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, New York, (1969), 37–48.
7. DOWELL, M., AND JARRATT, P. A modified regula falsi method for computing the root of an equation. *BIT 11* (1971), 168–174.
8. DOWELL, M., AND JARRATT, P. The 'Pegasus' method for computing the root of an equation. *BIT 12* (1972), 503–508.
9. GONNET, G. H. On the structure of zero finders. *BIT 17* (1977), 170–183.
10. LE, D. Three new rapidly convergent algorithms for finding a zero of a function. *SIAM J. Sci. Stat. Comp. 6,* 1 (1985), 193–208.
11. RALSTON, A., AND RABINOWITZ, P., *First Course in Numerical Analysis.* McGraw-Hill, New York, (1978).
12. SWIFT, A., AND LINDFIELD, G. R. Comparison of a continuation method with Brent's method for the numerical solution of a single nonlinear equation. *Comput. J. 21* (1978) 359–362.
13. TRAUB, J. F. *Iterative Methods for the Solution of Equations.* Prentice Hall, Englewood Cliffs, N. J., (1964).

Received September 1983; revised July 1984; accepted May 1985