Algorithm 748: Enclosing Zeros of Continuous Functions

G. E. ALEFELD
Universität Karlsruhe

F. A. POTRA
University of Iowa
and
YIXUN SHI
Bloomsburg University

Two efficient algorithms for enclosing a zero of a continuous function are presented. They are similar to the recent methods, but together with quadratic interpolation they make essential use of inverse cubic interpolation as well. Since asymptotically the inverse cubic interpolation is always chosen by the algorithms, they achieve higher-efficiency indices: 1.6529... for the first algorithm, and 1.6686... for the second one. It is proved that the second algorithm is optimal in a certain family. Numerical experiments show that the two new methods compare well with recent methods, as well as with the efficient solvers of Dekker, Brent, Bus and Dekker, and Le. The second method from the present article has the best behavior of all 12 methods especially when the termination tolerance is small.

Categories and Subject Descriptors: G.1.0 [**Numerical Analysis**]: General—*numerical algorithms*; G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*convergence*; *iterative methods*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Asymptotic efficiency index, enclosing method, inverse cubic interpolation, quadratic interpolation, simple root

1. INTRODUCTION

In a recent paper, Alefeld and Potra [1992] proposed three efficient methods for enclosing a simple zero x_* of a continuous function f. Starting with an initial enclosing interval $[a_1, b_1] = [a, b]$, the methods produce a sequence of

---

Authors' addresses: G. E. Alefeld, Institut für Angewandte Mathematik, Universität Karlsruhe, D-7500 Karlsruhe 1, Germany; F. A. Potra, Department of Mathematics, University of Iowa, Iowa City, IA 52242; Y. Shi, Department of Mathematics and Computer Science, Bloomsburg University, Bloomsburg, PA 17815.

intervals $\{[a_n, b_n]\}_{n=1}^{\infty}$, such that

$$x_* \in [a_{n+1}, b_{n+1}] \subseteq [a_n, b_n] \subseteq \cdots \subseteq [a_1, b_1] = [a, b] \tag{1}$$

$$\lim_{n \to \infty} (b_n - a_n) = 0. \tag{2}$$

The asymptotic efficiency indices of each of the three methods in the sense of Ostrowski [1973] are $2^{1/2} = 1.4142\ldots$, $4^{1/3} = 1.5874\ldots$, and $((3 + (13^{1/2}))/2)^{1/3} = 1.4892\ldots$, respectively. Subsequently, Alefeld et al. [1993] improved the methods of Alefeld and Potra and obtained two new enclosing methods having asymptotic efficiency indices $(1 + (2^{1/2}))^{1/2} = 1.5537$ and $(1 + (5^{1/2}))/2 = 1.6180\ldots$, respectively. The numerical experiments presented by Alefeld et al. show that the five methods mentioned above are about as efficient as the equation solvers of Brent [1972], Dekker [1969], and Le [1985]. The second method in Alefeld et al. has the best behavior of all eight methods.

Although there are many enclosing methods for solving the equation

$$f(x) = 0, \tag{3}$$

where f is continuous on $[a, b]$ and has a simple zero x_* in $[a, b]$, most of them do not have nice asymptotic convergence properties of the diameters $\{(b_n - a_n)\}_{n=1}^{\infty}$. For example, in case of Dekker's method, the diameters $b_n - a_n$ may remain greater than a relative large positive quantity until the last iteration when a "δ-step" is taken. In case of Le's [1985] Algorithm LZ4, the convergence properties of $\{(b_n - a_n)\}_{n=1}^{\infty}$ have not been proved except that the total number of function evaluations is bounded by four times of that needed by the bisection method, which is also an upper bound for the number of function evaluations required by the second method to be presented in this article.

Bus and Dekker [1975] published two improved versions of Dekker's [1969] method and proved that the upper bounds of the number of function evaluations are four or five times of that needed by the bisection method. However, for those two methods, as well as for Brent's method, the Illinois method, the Anderson-Björck method, Regula Falsi, Snyder's method, the Pegasus method, and so on, only the convergence rate of $\{|x_n - x_*|\}_{n=1}^{\infty}$, where x_n is the current estimate of x_*, has been studied and not the convergence rate of the diameters $(b_n - a_n)$. However, finding the rate of convergence of the sequence of the diameters is extremely important because in most algorithms for solving nonlinear equations the stopping criterion is constructed in terms of the diameter of the enclosing interval.

In case f is convex on $[a, b]$, the classical Newton-Fourier method [Ostrowski 1973, p. 248], Schmidt's [1971] method and the methods of Alefeld and Potra [1988] produce a sequence of enclosing intervals whose diameters are superlinearly convergent to zero. The highest asymptotic efficiency index of those methods, $1.5537\ldots$, is attained by a method of Schmidt and a slight modification of this method due to Alefeld-Potra. The convexity assumption was eventually removed in the methods of Alefeld and Potra

[1992], and the methods of Alefeld et al. [1993]. The second method in Alefeld et al. achieves the efficiency index $(1 + (5^{1/2}))/2 = 1.6180\ldots$ which was, up to that moment, the highest efficiency index for a general nonlinear equation solver with superlinear convergence of the diameters of the enclosing intervals and without any convexity requirements on f. The methods of Alefeld and Potra [1992] and Alefeld et al. are based on "double-length secant steps" and on appropriate use of quadratic interpolation and are briefly described in the next section.

We propose two methods which further improve the methods of Alefeld et al. [1992]. The improvements are achieved by employing inverse cubic interpolation instead of quadratic interpolation whenever possible. We show in Section 5 that asymptotically the inverse cubic interpolations will always be chosen by the algorithm. Our first method requires at most 3 while our second method requires at most 4 function evaluations per iteration. Asymptotically our first method requires only 2 and our second method only 3 function evaluations per iteration. For our first method, $\{(b_n - a_n)\}_{n=1}^{\infty}$ converges to zero with R-order at least $1 + (3^{1/2}) = 2.732\ldots$, while for our second method $\{(b_n - a_n)\}_{n=1}^{\infty}$ converges to zero with R-order at least $2 + (7^{1/2}) = 4.646\ldots$. Hence the corresponding efficiency indices are $(1 + (3^{1/2}))^{1/2} = 1.6529\ldots$ and $(2 + (7^{1/2}))^{1/3} = 1.6686\ldots$, respectively. We also show that our second method is optimal in a certain class of algorithms.

Section 3 describes our subroutine for inverse cubic interpolation, and Section 4 presents the major algorithms of this article. In Section 5 the convergence results are proved, and in Section 6 numerical experiments are presented. We compare the two methods of this article with the methods in Alefeld and Potra [1992] and Alefeld et al. [1993], with the methods of Brent [1972] and Dekker [1969] which are used in many standard software packages, with the Algorithms M and R of Bus and Dekker [1975], and with the Algorithm LZ4 of Le [1985]. The numerical results show that the two methods of the present article compare well with the other 10 methods. The second method in this article has the best behavior among all methods especially when the termination tolerance is small.

2. SOME RECENT ENCLOSING METHODS

In this section we briefly describe the recently developed enclosing algorithms of Alefeld and Potra [1992] and their improvements proposed by Alefeld et al. [1993] for enclosing a simple zero x_* of a continuous function f in $[a, b]$ where $f(a)f(b) < 0$. In all, there are three methods proposed in Alefeld and Potra and two methods proposed in Alefeld et al. "Double-length secant step" is used by all five methods, and quadratic interpolation techniques are applied in all but the first method of Alefeld and Potra. In the present article we call those methods Algorithms 2.1–2.5 and summarize their asymptotic convergence properties in the following table, where NFM stands for "the maximum number of function evaluations required per iteration," NFA for "the number of function evaluations required asymptotically per iteration," and AEI for "asymptotic efficiency index" (the values of AEI are rounded to the given number of digits).

Algorithm	Method	NFM	NFA	AEI
2.1	Method 1 of Alefeld and Potra [1992]	3	2	1.4142
2.2	Method 2 of Alefeld and Potra [1992]	4	3	1.5874
2.3	Method 3 of Alefeld and Potra [1992]	3	3	1.4892
2 4	Method 1 of Alefeld et al. [1993]	3	2	1.5537
2.5	Method 2 of Alefeld et al. [1993]	4	3	1.6180

We first list out two subroutines that are called by Algorithms 2.1–2.5 as well as by Algorithms 4.1 and 4.2 in Section 4. We assume throughout that f is continuous on $[a, b]$ and that $f(a)f(b) < 0$. We consider a point $c \in (a, b)$.

Subroutine *bracket*$(a, b, c, \bar{a}, \bar{b})$ (or *bracket*$(a, b, c, \bar{a}, \bar{b}, d)$)
If $f(c) = 0$, then print c and stop;
If $f(a)f(c) < 0$, then $\bar{a} = a$, $\bar{b} = c$, $(d = b)$;
If $f(b)f(c) < 0$, then $\bar{a} = c$, $\bar{b} = b$, $(d = a)$.

After calling the above subroutine, we will have a new interval $[\bar{a}, \bar{b}] \subset [a, b]$ with $f(\bar{a})f(\bar{b}) < 0$. Furthermore, if *bracket*$(a, b, c, \bar{a}, \bar{b}, d)$ is called, then we will have a point $d \notin [\bar{a}, \bar{b}]$ such that if $d < \bar{a}$ then $f(\bar{a})f(d) > 0$; otherwise $f(d)f(\bar{b}) > 0$.

Subroutine *Newton-Quadratic*(a, b, d, r, k)
Set $A = f[a, b, d]$, $B = f[a, b]$;
If $A = 0$, then $r = a - B^{-1}f(a)$;
If $Af(a) > 0$, then $r_0 = a$, else $r_0 = b$;
For $i = 1, 2, \ldots, k$ do:

$$\begin{aligned} r_i &= r_{i-1} - \frac{P(r_{i-1})}{P'(r_{i-1})} \\ &= r_{i-1} - \frac{B(r_{i-1})}{B + A(2r_{i-1} - a - b)} \end{aligned} \tag{4}$$

$r = r_k$.

The above subroutine has a, b, d, and k as inputs and r as output. It is assumed that $d \notin [a, b]$ and that $f(d)f(a) > 0$ if $d < a$ and $f(d)f(b) > 0$ if $d > b$. k is a positive integer, and r is an approximation of the unique zero z of the quadratic polynomial,

$$P(x) = P(a, b, d)(x) = f(a) + f[a, b](x - a) + f[a, b, d](x - a)(x - b)$$

in $[a, b]$ where $f[a, b] = (f(b) - f(a))/(b - a)$, and $f[a, b, d] = (f[b, d] - f[a, b])/(d - a)$; note that $P(a) = f(a)$ and $P(b) = f(b)$. Hence $P(a)P(b) < 0$.

The following five algorithms describe the methods in Alefeld and Potra [1992] and Alefeld et al. [1993], where $\mu < 1$ is a positive parameter which is usually chosen as $\mu = 0.5$.

Algorithm 2.1 (*Alefeld and Potra* [1992])

set $a_1 = a$, $b_1 = b$, for $n = 1, 2, \ldots,$ do:

2.1.1 $c_n = a_n - f[a_n, b_n]^{-1} f(a_n)$;

2.1.2 call *bracket*$(a_n, b_n, c_n, \bar{a}_n, \bar{b}_n)$;

2.1.3 if $|f(\bar{a}_n)| < |f(\bar{b}_n)|$, then set $u_n = \bar{a}_n$, else set $u_n = \bar{b}_n$;

2.1.4 set $\bar{c}_n = u_n - 2f[\bar{a}_n, \bar{b}_n]^{-1} f(u_n)$;

2.1.5 if $|\bar{c}_n - u_n| > 0.5(\bar{b}_n - \bar{a}_n)$,
then $\hat{c}_n = 0.5(\bar{b}_n + \bar{a}_n)$, else $\hat{c}_n = \bar{c}_n$;

2.1.6 call *bracket*$(\bar{a}_n, \bar{b}_n, \hat{c}_n, \hat{a}_n, \hat{b}_n)$;

2.1.7 if $\hat{b}_n - \hat{a}_n < \mu(b_n - a_n)$,
then $a_{n+1} = \hat{a}_n$, $b_{n+1} = \hat{b}_n$,
else call *bracket*$(\hat{a}_n, \hat{b}_n, 0.5(\hat{a}_n + \hat{b}_n), a_{n+1}, b_{n+1})$.

Algorithm 2.2 (*Alefeld and Potra* [1992])

set $a_1 = a$, $b_1 = b$, for $n = 1, 2, \ldots$ do:

2.2.1 $c_n = a_n - f[a_n, b_n]^{-1} f(a_n)$;

2.2.2 call *bracket*$(a_n, b_n, c_n, \tilde{a}_n, \tilde{b}_n)$;

2.2.3 $\tilde{c}_n$ = the unique zero of $P(a_n, b_n, c_n)(x)$ in $[\tilde{a}_n, \tilde{b}_n]$;

2.2.4 call *bracket*$(\tilde{a}_n, \tilde{b}_n, \tilde{c}_n, \bar{a}_n, \bar{b}_n)$;

2.2.5–2.2.9: same as 2.1.3–2.1.7.

Algorithm 2.3 (*Alefeld and Potra* [1992])

set $a_1 = a$, $b_1 = b$, for $n = 1, 2, \ldots$ do:

2.3.1 $c_n = 0.5(a_n + b_n)$;

2.3.2–2.3.6: same as 2.2.2–2.2.6;

2.3.7 call *bracket*$(\bar{a}_n, \bar{b}_n, \bar{c}_n, a_{n+1}, b_{n+1})$.

Algorithm 2.4 (*Alefeld et al.* [1993])

2.4.1 set $a_1 = a$, $b_1 = b$, $c_1 = a_1 - f[a_1, b_1]^{-1} f(a_1)$;

2.4.2 call *bracket*$(a_1, b_1, c_1, a_2, b_2, d_2)$;

For $n = 2, 3, \ldots,$ do:

2.4.3 call *Newton-Quadratic*$(a_n, b_n, d_n, c_n, 2)$;

2.4.4 call *bracket*$(a_n, b_n, c_n, \bar{a}_n, \bar{b}_n, \bar{d}_n)$;

2.4.5 if $|f(\bar{a}_n)| < |f(\bar{b}_n)|$, then set $u_n = \bar{a}_n$, else set $u_n = \bar{b}_n$;

2.4.6 set $\bar{c}_n = u_n - 2f[\bar{a}_n, \bar{b}_n]^{-1} f(u_n)$;

2.4.7 if $|\bar{c}_n - u_n| > 0.5(\bar{b}_n - \bar{a}_n)$,
then $\hat{c}_n = 0.5(\bar{b}_n + \bar{a}_n)$, else $\hat{c}_n = \bar{c}_n$;

2.4.8 call *bracket*$(\bar{a}_n, \bar{b}_n, \hat{c}_n, \hat{a}_n, \hat{b}_n, \hat{d}_n)$;

2.4.9 if $\hat{b}_n - \hat{a}_n < \mu(b_n - a_n)$,
then $a_{n+1} = \hat{a}_n$, $b_{n+1} = \hat{b}_n$, $d_{n+1} = \hat{d}_n$
else call *bracket*$(\hat{a}_n, \hat{b}_n, 0.5(\hat{a}_n + \hat{b}_n), a_{n+1}, b_{n+1}, d_{n+1})$.

Algorithm 2.5 (*Alefeld et al.* [1993])

2.5.1–2.5.2: same as 2.4.1–2.4.2;
For $n = 2, 3, \ldots,$ do:
2.5.3 call *Newton-Quadratic*$(a_n, b_n, d_n, c_n, 2)$;
2.5.4 call *bracket*$(a_n, b_n, c_n, \tilde{a}_n, \tilde{b}_n, \tilde{d}_n)$;
2.5.5 call *Newton-Quadratic*$(\tilde{a}_n, \tilde{b}_n, \tilde{d}_n, \tilde{c}_n, 3)$;
2.5.6 call *bracket*$(\tilde{a}_n, \tilde{b}_n, \tilde{c}_n, \bar{a}_n, \bar{b}_n, \bar{d}_n)$;
2.5.7–2.5.11: same as 2.4.5–2.4.9.

3. A BASIC SUBROUTINE

In this section we describe a subroutine for approximating a zero of f by using the inverse cubic interpolation. This subroutine will be called by the algorithms described in the next section. Assume that f is continuous on a closed interval I, that f has a zero in I, and that a, b, c, d are four numbers in I. If $f(a)$, $f(b)$, $f(c)$, and $f(d)$ are four distinct values, then the inverse interpolation polynomial at $(a, f(a))$, $(b, f(b))$, $(c, f(c))$, and $(d, f(d))$ is given by the formula

$$\begin{aligned} IP(y) = a &+ (y - f(a))f^{-1}[f(a), f(b)] \\ &+ (y - f(a))(y - f(b))f^{-1}[f(a), f(b), f(c)] \\ &+ (y - f(a))(y - f(b))(y - f(c))f^{-1}[f(a), f(b), f(c), f(d)], \end{aligned} \tag{5}$$

where

$$f^{-1}[f(a), f(b)] = \frac{b - a}{f(b) - f(a)},$$

$$f^{-1}[f(a), f(b), f(c)] = \frac{f^{-1}[f(b), f(c)] - f^{-1}[f(a), f(b)]}{f(c) - f(a)},$$

and

$$f^{-1}[f(a), f(b), f(c), f(d)] = \frac{f^{-1}[f(b), f(c), f(d)] - f^{-1}[f(a), f(b), f(c)]}{f(d) - f(a)}.$$

Notice that the polynomial $IP(y)$ in (5) can always be constructed as long as $f(a)$, $f(b)$, $f(c)$, and $f(d)$ are distinct, even if f is not invertible. Then we may always compute $\bar{x} = IP(0)$, which is an "approximate solution" of $f(x) = 0$ although $\bar{x}$ may lie outside of I. We are interested in the case where $f(a)$, $f(b)$, $f(c)$, and $f(d)$ are distinct and where $\bar{x}$ is in I. We will prove that this will always happen asymptotically.

In case f is continuously differentiable with $f'(x) \neq 0$ for all $x \in I$ and $f(a)f(b) < 0$ for some $[a, b] \subseteq I$, $f^{-1}(x)$ exists, and a simple root x_* of

$f(x) = 0$ lies in $[a, b]$. In this case, if we further assume that $f^{(4)}(x)$ exists and is continuous on I, then

$$|\bar{x} - x_*| = |IP(0) - f^{-1}(0)| \leq |f(a)||f(b)||f(c)||f(d)| \frac{\max_{y \in f(I)} \left| \left[f^{-1}(y) \right]^{(4)} \right|}{4!}. \tag{6}$$

Since

$$\left[f^{-1}(y) \right]^{(4)} = \frac{10 f'(x) f''(x) f'''(x) - 15[f''(x)]^3 - [f'(x)]^2 f^{(4)}(x)}{[f'(x)]^7}$$

for all $y \in f(I)$ with $x = f^{-1}(y) \in I$, we deduce that

$$|\bar{x} - x_*| \leq M |f(a)||f(b)||f(c)||f(d)|, \tag{7}$$

where

$$M = \frac{10 M_1 M_2 M_3 + 15 M_2^3 + M_1^2 M_4}{(m_1)^7} \tag{8}$$

with $M_1 = \max_{x \in I} |f'(x)|$, $M_2 = \max_{x \in I} |f''(x)|$, $M_3 = \max_{x \in I} |f'''(x)|$, $M_4 = \max_{x \in I} |f^{(4)}(x)|$, and $m_1 = \min_{x \in I} |f'(x)|$. We mention that $m_1 > 0$ because I is assumed to be a closed interval. The following procedure for calculating $\bar{x} = IP(0)$ is a slight modification of the Aitken-Neville interpolation algorithm that avoids unnecessary roundoff errors, as described in Stoer and Bulirsch [1980].

Subroutine *ipzero*$(a, b, c, d, \bar{x})$
set

$$Q_{11} = (c - d) \frac{f(c)}{f(d) - f(c)},$$

$$Q_{21} = (b - c) \frac{f(b)}{f(c) - f(b)},$$

$$Q_{31} = (a - b) \frac{f(a)}{f(b) - f(a)},$$

$$D_{21} = (b - c) \frac{f(c)}{f(c) - f(b)},$$

$$D_{31} = (a - b) \frac{f(b)}{f(b) - f(a)},$$

$$Q_{22} = (D_{21} - Q_{11}) \frac{f(b)}{f(d) - f(b)},$$

$$Q_{32} = (D_{31} - Q_{21})\frac{f(a)}{f(c) - f(a)},$$

$$D_{32} = (D_{31} - Q_{21})\frac{f(c)}{f(c) - f(a)},$$

$$Q_{33} = (D_{32} - Q_{22})\frac{f(a)}{f(d) - f(a)},$$

$\bar{x} = a + (Q_{31} + Q_{32} + Q_{33})$, end.

4. ALGORITHMS

In this section we present two algorithms for enclosing a simple zero x_* of a continuous function f in $[a, b]$ where $f(a)f(b) < 0$. These two algorithms are improvements of Algorithm 2.4 and Algorithm 2.5. They call the subroutines *bracket* and *Newton-Quadratic* as described in Section 2, as well as the subroutine *ipzero* from the previous section. The basic idea is that we will make use of $\bar{x} = IP(0)$ whenever it is computable and lies inside the current enclosing interval, which is always the case asymptotically. The first algorithm requires at most 3 while asymptotically 2 function evaluations per iteration, and the second algorithm requires at most 4 while asymptotically 3 function evaluations per iteration. Under certain assumptions the first algorithm has an asymptotic efficiency index $(1 + (3^{1/2}))^{1/2} = 1.6529\ldots$, and the second algorithm has an asymptotic index $(2 + (7^{1/2}))^{1/3} = 1.6686\ldots$. We also show that in a certain sense our second algorithm is an optimal procedure. In the following algorithms, $\mu < 1$ is a positive parameter which is usually chosen as $\mu = 0.5$.

Algorithm 4.1

4.1.1 set $a_1 = a$, $b_1 = b$, $c_1 = a_1 - f[a_1, b_1]^{-1}f(a_1)$;
4.1.2 call *bracket*$(a_1, b_1, c_1, a_2, b_2, d_2)$;
For $n = 2, 3, \ldots,$ do:
4.1.3 if $n = 2$ or $\prod_{i \neq j}(f_i - f_j) = 0$ where $f_1 = f(a_n)$, $f_2 = f(b_n)$, $f_3 = f(d_n)$, and $f_4 = f(e_n)$,
  then call *Newton-Quadratic*$(a_n, b_n, d_n, c_n, 2)$,
 else
  call *ipzero*$(a_n, b_n, d_n, e_n, c_n)$,
  if $(c_n - a_n)(c_n - b_n) \geq 0$
  then call *Newton-Quadratic*$(a_n, b_n, d_n, c_n, 2)$,
 endif;
4.1.4 call *bracket*$(a_n, b_n, c_n, \bar{a}_n, \bar{b}_n, \bar{d}_n)$;
4.1.5 if $|f(\bar{a}_n)| < |f(\bar{b}_n)|$, then set $u_n = \bar{a}_n$, else set $u_n = \bar{b}_n$;
4.1.6 set $\bar{c}_n = u_n - 2f[\bar{a}_n, \bar{b}_n]^{-1}f(u_n)$;
4.1.7 if $|\bar{c}_n - u_n| > 0.5(\bar{b}_n - \bar{a}_n)$,
  then $\hat{c}_n = 0.5(\bar{b}_n + \bar{a}_n)$, else $\hat{c}_n = \bar{c}_n$;

4.1.8 call $bracket(\bar{a}_n, \bar{b}_n, \hat{c}_n, \hat{a}_n, \hat{b}_n, \hat{d}_n)$;
4.1.9 if $\hat{b}_n - \hat{a}_n < \mu(b_n - a_n)$,
then $a_{n+1} = \hat{a}_n$, $b_{n+1} = \hat{b}_n$, $d_{n+1} = \hat{d}_n$, $e_{n+1} = \bar{d}_n$,
else
$e_{n+1} = \hat{d}_n$,
call $bracket(\hat{a}_n, \hat{b}_n, 0.5(\hat{a}_n + \hat{b}_n), a_{n+1}, b_{n+1}, d_{n+1})$,
endif.

Algorithm 4.2
4.2.1–4.2.2: same as 4.1.1–4.1.2;
For $n = 2, 3, \ldots,$ do:
4.2.3 if $n = 2$ or $\prod_{i \neq j}(f_i - f_j) = 0$ where $f_1 = f(a_n)$, $f_2 = f(b_n)$, $f_3 = f(d_n)$, and $f_4 = f(e_n)$,
then call *Newton-Quadratic*$(a_n, b_n, d_n, c_n, 2)$,
else
call *ipzero*$(a_n, b_n, d_n, e_n, c_n)$,
if $(c_n - a_n)(c_n - b_n) \geq 0$
then call *Newton-Quadratic*$(a_n, b_n, d_n, c_n, 2)$,
endif;
4.2.4 set $\tilde{e}_n = d_n$, call $bracket(a_n, b_n, c_n, \tilde{a}_n, \tilde{b}_n, \tilde{d}_n)$;
4.2.5 if $\prod_{i \neq j}(\tilde{f}_i - \tilde{f}_j) = 0$ where $\tilde{f}_1 = f(\tilde{a}_n)$, $\tilde{f}_2 = f(\tilde{b}_n)$, $\tilde{f}_3 = f(\tilde{d}_n)$, $\tilde{f}_4 = f(\tilde{e}_n)$,
then call *Newton-Quadratic*$(\tilde{a}_n, \tilde{b}_n, \tilde{d}_n, \tilde{c}_n, 3)$,
else
call *ipzero*$(\tilde{a}_n, \tilde{b}_n, \tilde{d}_n, \tilde{e}_n, \tilde{c}_n)$,
if $(\tilde{c}_n - \tilde{a}_n)(\tilde{c}_n - \tilde{b}_n) \geq 0$
then call *Newton-Quadratic*$(\tilde{a}_n, \tilde{b}_n, \tilde{d}_n, \tilde{c}_n, 3)$,
endif;
4.2.6 call $bracket(\tilde{a}_n, \tilde{b}_n, \tilde{c}_n, \bar{a}_n, \bar{b}_n, \bar{d}_n)$;
4.2.7–4.2.11: same as 4.1.5–4.1.9.

The following theorem contains a basic property of the above algorithms, whose proof is straightforward and hence will be omitted.

THEOREM 4.3. *Let f be continuous on $[a, b]$, $f(a)f(b) < 0$, and consider either Algorithm 4.1 or Algorithm 4.2. Then either a zero of f is found in a finite number of iterations, or the sequence of the intervals $\{[a_n, b_n]\}_{n=1}^{\infty}$ satisfies both (1) and (2) where x_* is a zero of f in $[a, b]$.*

5. CONVERGENCE THEOREMS

From the previous section it is clear that the intervals $\{[a_n, b_n]\}_{n=1}^{\infty}$ produced by either Algorithm 4.1 or Algorithm 4.2 satisfy $b_{n+1} - a_{n+1} \leq \mu_1(b_n - a_n)$ for $n \geq 2$, where $\mu_1 = \max\{\mu, 0.5\}$. Since $\mu_1 < 1$, that shows at least linear convergence. In what follows we show that under certain smoothness assumptions Algorithm 4.1 and Algorithm 4.2 produce intervals whose diameters $\{(b_n - a_n)\}_{n=1}^{\infty}$ converge to zero with R-orders at least $1 + 3^{1/2} = 2.732\ldots$ and $2 + 7^{1/2} = 4.646\ldots$, respectively. First, we have the following two lemmas.

LEMMA 5.1 (ALEFELD-POTRA [1992]). *Assume that f is continuously differentiable in $[a, b]$, that $f(a)f(b) < 0$, and that x_* is a simple root of $f(x) = 0$ in $[a, b]$. Suppose that Algorithm 4.1 (or Algorithm 4.2) does not terminate after a finite number of iterations. Then there is an n_3 such that for all $n > n_3$, $\bar{c}_n$ and u_n in step 4.1.6 (or in step 4.2.8) satisfy*

$$f(\bar{c}_n)f(u_n) < 0. \tag{9}$$

LEMMA 5.2. *Under the hypothesis of Lemma 5.1, assume that f is four times continuously differentiable on $[a, b]$. Then:*

(1) *For Algorithm 4.1, there is $r_1 > 0$ and n_1 such that c_n in step 4.1.3 will always be obtained by calling ipzero for all $n > n_1$, and*

$$|f(c_n)| \leq r_1(b_n - a_n)^2(b_{n-1} - a_{n-1})^2, \qquad \forall n > n_1. \tag{10}$$

(2) *For Algorithm 4.2, there is $r_2 > 0$ and n_2 such that c_n in step 4.2.3 and $\tilde{c}_n$ in step 4.2.5 will always be obtained by calling ipzero for all $n > n_2$, and*

$$|f(\tilde{c}_n)| \leq r_2(b_n - a_n)^4(b_{n-1} - a_{n-1})^3, \qquad \forall n > n_2. \tag{11}$$

PROOF. By Theorem 4.1, $x_* \in (a_n, b_n)$, and

$$b_n - a_n \to 0. \tag{12}$$

Since x_* is a simple zero, $f'(x_*) \neq 0$. Therefore, when n is big enough $f'(x) \neq 0$ for all $x \in [a_n, b_n]$. For simplicity, we assume that $f'(x) \neq 0$ for all $x \in [a, b]$. With this assumption, f is strictly monotone on $[a, b]$, and hence f_i $(i = 1, 2, 3, 4)$ in step 4.1.3 are four distinct values. Therefore, the subroutine *ipzero* will always be called in step 4.1.3, and now we need only to prove that c_n calculated from *ipzero* satisfies $c_n \in (a_n, b_n)$ whenever n is large enough.

From (7) we see that

$$\begin{aligned} |c_n - x_*| &\leq M|f(a_n)||f(b_n)||f(d_n)||f(e_n)| \\ &\leq M(M_1)^4(b_n - a_n)^2(b_{n-1} - a_{n-1})^2 \end{aligned} \tag{13}$$

where M and M_1 are as defined in (7) and (8) with the interval I replaced by $[a, b]$. Since $x_* \in (a, b)$, there is an $\epsilon > 0$ such that $[x_* - \epsilon, x_* + \epsilon] \subset (a, b)$. Hence (13) and (12) imply that there is an $\bar{n}$ such that

$$c_n \in [x_* - \epsilon, x_* + \epsilon] \subset (a, b), \qquad \forall n \geq \bar{n}. \tag{14}$$

Therefore the following inequality

$$|f(c_n)| \le M_1 |c_n - x_*| \tag{15}$$

holds for $n \ge \bar{n}$, and as a result we have

$$|f(c_n)| \le M_1 |c_n - x_*| \le M(M_1)^4 (b_n - a_n)(b_{n-1} - a_{n-1})^2 |f(a_n)|$$

as well as

$$|f(c_n)| \le M_1 |c_n - x_*| \le M(M_1)^4 (b_n - a_n)(b_{n-1} - a_{n-1})^2 |f(b_n)|.$$

Equation (12) enables us again to choose an $n_1 \ge \bar{n}$ such that $c_n \in (a, b)$ for all $n \ge n_1$ and

$$|f(c_n)| < \min\{|f(a_n)|, |f(b_n)|\}, \qquad \forall n \ge n_1. \tag{16}$$

Since f is strictly monotone over $[a, b]$, and $f(a_n)f(b_n) < 0$, (16) implies that $c_n \in (a_n, b_n)$ whenever $n \ge n_1$. Therefore c_n in step 4.1.3 will always be obtained from *ipzero* for all $n \ge n_1$, and now (10) follows immediately from (13) and (15) with $r_1 = M(M_1)^5$.

A similar argument can be applied to show that there is an n_2 such that c_n in step 4.2.3 and $\tilde{c}_n$ in step 4.2.5 will always be obtained from *ipzero* for all $n \ge n_2$. For $n \ge n_2$ we can write,

$$\begin{aligned}
|f(\tilde{c}_n)| &\le M_1 |\tilde{c}_n - x_*| \\
&\le M_1 M |f(\tilde{a}_n)||f(\tilde{b}_n)||f(\tilde{d}_n)||f(\tilde{e}_n)| \\
&= M_1 M |f(a_n)||f(b_n)||f(c_n)||f(d_n)| \\
&\le (M_1)^4 M (b_n - a_n)^2 (b_{n-1} - a_{n-1}) |f(c_n)| \\
&\le (M_1)^9 M^2 (b_n - a_n)^4 (b_{n-1} - a_{n-1})^3
\end{aligned}$$

which proves (11) with $r_2 = (M_1)^9 M^2$. □

The following two theorems show the asymptotic convergence properties of Algorithm 4.1 and Algorithm 4.2, respectively.

THEOREM 5.3. *Under the assumptions of Lemma* 5.2, *the sequence of diameters* $\{(b_n - a_n)\}_{n=1}^{\infty}$ *produced by Algorithm* 4.1 *converges to zero, and there is an* $L_1 > 0$ *such that*

$$b_{n+1} - a_{n+1} \le L_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1})^2, \qquad \forall n = 2, 3, \ldots. \tag{17}$$

Moreover, there is an N_1 *such that for all* $n > N_1$ *we have*

$$a_{n+1} = \hat{a}_n \quad \text{and} \quad b_{n+1} = \hat{b}_n.$$

Hence when $n > N_1$, *Algorithm* 4.1 *requires only two function evaluations per iteration.*

PROOF. As in the proof of Lemma 5.2 we assume without loss of generality that $f'(x) \ne 0$ for all $x \in [a, b]$. Take N_1 such that $N_1 > \max\{n_1, n_3\}$. Then by Lemma 5.1, (9) holds for all $n > N_1$. From steps 4.1.6–4.1.8 of Algorithm 4.1 and the fact that $u_n, \bar{c}_n \in [\bar{a}_n, \bar{b}_n]$ we deduce that

$$\hat{b}_n - \hat{a}_n \le |\bar{c}_n - u_n|, \qquad \forall n > N_1. \tag{18}$$

From step 4.1.6 we also see that

$$\begin{aligned} |\bar{c}_n - u_n| &= \left|2f[\bar{a}_n, \bar{b}_n]^{-1} f(u_n)\right| \\ &\le \frac{2}{m_1}|f(u_n)|, \end{aligned} \tag{19}$$

where m_1 is as defined in (8) with the interval I replaced by $[a, b]$. Finally, since $c_n \in \{\bar{a}_n, \bar{b}_n\}$, we have that $|f(u_n)| \le |f(c_n)|$. Combining that with (18) and (19) we have

$$\hat{b}_n - \hat{a}_n \le \frac{2}{m_1}|f(c_n)|, \qquad \forall n > N_1. \tag{20}$$

Now by Lemma 5.2, $|f(c_n)| \le r_1(b_n - a_n)^2(b_{n-1} - a_{n-1})^2$, so that

$$\hat{b}_n - \hat{a}_n \le \frac{2}{m_1} r_1 (b_n - a_n)^2 (b_{n-1} - a_{n-1})^2, \qquad \forall n > N_1. \tag{21}$$

Since $\{(b_n - a_n)\}_{n=1}^{\infty}$ converges to zero, if N_1 is large enough then

$$\hat{b}_n - \hat{a}_n < \mu(b_n - a_n), \qquad \forall n > N_1.$$

This shows that for all $n > N_1$ we will have $a_{n+1} = \hat{a}_n$ and $b_{n+1} = \hat{b}_n$. By taking

$$L_1 \ge \max\left\{\frac{2}{m_1}r_1, \frac{(b_{n+1} - a_{n+1})}{(b_n - a_n)^2(b_{n-1} - a_{n-1})^2}\right\} \qquad n = 2, 3, \ldots, N_1$$

and using (21) we obtain (17). □

COROLLARY 5.4. *Under the assumptions of Theorem* 5.3, $\{\epsilon_n\}_{n=1}^{\infty} = \{(b_n - a_n)\}_{n=1}^{\infty}$ *converges to zero with R-order at least* $1 + 3^{1/2} = 2.732\ldots$. *Since asymptotically Algorithm* 4.1 *requires only two function evaluations per iteration, its efficiency index is* $(1 + (3^{1/2}))^{1/2} = 1.6529\ldots$.

PROOF. By Theorem 5.3, $\{\epsilon_n\}_{n=1}^{\infty}$ converges to zero, and $\epsilon_{n+1} \le L_1 \epsilon_n^2 \epsilon_{n-1}^2$, for $n = 2, 3, \ldots$; and the result follows by invoking Theorem 2.1 of Potra [1989]. □

THEOREM 5.5. *Under the assumptions of Lemma* 4.2, *the sequence of diameters* $\{(b_n - a_n)\}_{n=1}^{\infty}$ *produced by Algorithm* 4.2 *converges to zero, and there is an* $L_2 > 0$ *such that*

$$b_{n+1} - a_{n+1} \le L_2(b_n - a_n)^4(b_{n-1} - a_{n-1})^3, \qquad \forall n = 2, 3, \ldots. \tag{22}$$

Moreover, there is an N_2 *such that for all* $n > N_2$ *we have*

$$a_{n+1} = \hat{a}_n \quad \text{and} \quad b_{n+1} = \hat{b}_n.$$

Hence when $n > N_2$, Algorithm 4.2 requires only three function evaluations per iteration.

PROOF. The proof is about the same as that of Theorem 5.3. We assume that $f'(x) \neq 0$ for all $x \in [a, b]$. Take N_2 such that $N_2 > \max\{n_2, n_3\}$. When $n > N_2$ then, as in the proof of Theorem 5.3, we have

$$\hat{b}_n - \hat{a}_n \leq \frac{2}{m_1} |f(\tilde{c}_n)|. \tag{23}$$

Now by Lemma 5.2, $|f(\tilde{c}_n)| \leq r_2(b_n - a_n)^4(b_{n-1} - a_{n-1})^3$. Therefore

$$\hat{b}_n - \hat{a}_n \leq \frac{2}{m_1} r_2(b_n - a_n)^4(b_{n-1} - a_{n-1})^3, \qquad \forall n > N_2. \tag{24}$$

The rest of the proof is similar to the corresponding part of the proof of Theorem 5.3 and is omitted. □

COROLLARY 5.6. *Under the assumptions of Theorem 5.5, $\{\epsilon_n\}_{n=1}^{\infty} = \{(b_n - a_n)\}_{n=1}^{\infty}$ converges to zero with R-order at least $2 + 7^{1/2} = 4.646\ldots$. Since asymptotically Algorithm 4.2 requires only three function evaluations per iteration, its efficiency index is $(2 + (7^{1/2}))^{1/3} = 1.6686\ldots$.* □

Next, we notice that Algorithm 4.2 is an optimal procedure in the following sense. It is clear that Algorithm 4.2 improves Algorithm 4.1 by repeating 4.2.3–4.2.4 in 4.2.5–4.2.6. If we repeat this k times, we will get an algorithm of the form:

Algorithm 5.7

5.1.1–5.1.2: same as 4.2.1–4.2.2;
for $n = 2, 3, \ldots,$ do
5.1.3: same as 4.2.3;
5.1.4: set $e_n^{(1)} = d_n$, call *bracket*$(a_n, b_n, c_n, a_n^{(1)}, b_n^{(1)}, d_n^{(1)})$;
⋮
5.1.2k: set $e_n^{(k-1)} = d_n^{(k-2)}$, call *bracket*$(a_n^{(k-2)}, b_n^{(k-2)}, c_n^{(k-2)}, a_n^{(k-1)}, b_n^{(k-1)}, d_n^{(k-1)})$;
5.1.2k + 1:
  if $\prod_{i \neq j}(\tilde{f}_i - \tilde{f}_j) = 0$ where $\tilde{f}_1 = f(a_n^{(k-1)})$, $\tilde{f}_2 = f(b_n^{(k-1)})$, $\tilde{f}_3 = f(d_n^{(k-1)})$, $\tilde{f}_4 = f(e_n^{(k-1)})$,
    then call *Newton-Quadratic*$(a_n^{(k-1)}, b_n^{(k-1)}, d_n^{(k-1)}, \tilde{c}_n, k + 1)$,
  else
    call *ipzero*$(a_n^{(k-1)}, b_n^{(k-1)}, d_n^{(k-1)}, e_n^{(k-1)}, \tilde{c}_n)$,
    if $(\tilde{c}_n - a_n^{(k-1)})(\tilde{c}_n - b_n^{(k-1)}) \geq 0$
    then call *Newton-Quadratic*$(a_n^{(k-1)}, b_n^{(k-1)}, d_n^{(k-1)}, \tilde{c}_n, k + 1)$,
  endif;
5.1.2k + 2: call *bracket*$(a_n^{(k-1)}, b_n^{(k-1)}, \tilde{c}_n, \bar{a}_n, \bar{b}_n, \bar{d}_n)$;
5.1.2k + 3–5.1.2k + 7: same as 4.2.7–4.2.11.

Algorithms 4.1 and 4.2 are special cases of Algorithm 5.7. Furthermore, when $k \geq 2$, similar to Lemma 5.2, Theorem 5.3, and Theorem 5.5 we see that for Algorithm 5.7,

$$(b_{n+1} - a_{n+1}) \leq L_k (b_n - a_n)^{3k-2} (b_{n-1} - a_{n-1})^3, \qquad n = 2, 3, \ldots$$

for some $L_k > 0$. Hence when $k \geq 2$ Algorithm 5.7 has the R-order at least

$$\tau = \frac{3k-2}{2} + \sqrt{3 + \left(\frac{3k-2}{2}\right)^2},$$

which is the positive root of the equation $t^2 - (3k-2)t - 3 = 0$. Since asymptotically Algorithm 5.7 requires $k + 1$ function evaluations per iteration, the efficiency index is

$$I_k = \left(\frac{3k-2}{2} + \sqrt{3 + \left(\frac{3k-2}{2}\right)^2}\right)^{1/(k+1)}$$

when $k \geq 2$. In a straightforward manner it can be proved that $I_k < I_2$ for all $k > 2$. Therefore, Algorithm 4.2 is optimal.

6. NUMERICAL EXPERIMENTS

In this section we present our numerical experiments comparing Algorithms 4.1 and 4.2 with Algorithms 2.1–2.5, with the methods of Dekker [1969] and Brent [1972], with the Algorithms M and R of Bus and Dekker [1975], and with the Algorithm LZ4 of Le [1985]. In our experiments, the parameter μ in Algorithms 2.1–2.5 and 4.1–4.2 was chosen as 0.5. For Dekker's method we translated the ALGOL 60 routine Zeroin, presented by Dekker, into Fortran; for Algorithms M and R of Bus and Dekker we did the same (i.e., we translated into Fortran the ALGOL 60 routines Zeroin and Zeroinrat presented in Bus and Dekker); for Brent's method we simply used the Fortran routine Zero presented in the Appendix of Brent, while for the Algorithm LZ4 of Le we used his Fortran code. The machine used was an AT&T 3B2-1000 Model 80, in double precision. The test problems are listed in Table I. The termination criterion was the one used by Brent, i.e.,

$$b - a \leq 2 \cdot tole(a, b), \tag{25}$$

where $[a, b]$ is the current enclosing interval, and

$$tole(a, b) = 2 \cdot |u| \cdot macheps + tol.$$

Here $u \in \{a, b\}$ such that $|f(u)| = \min\{|f(a)|, |f(b)|\}$; *macheps* is the relative machine precision which in our case is $1.9073486328 \times 10^{-16}$, and *tol* is a user-given nonnegative number.

Table I. Test Problems

#	function $f(x)$	$[a,b]$	parameter
1	$\sin x - x/2$	$[\pi/2, \pi]$	
2	$-2\sum_{i=1}^{20}(2i-5)^2/(x-i^2)^3$	$[a_n, b_n]$ $a_n = n^2 + 10^{-9}$ $b_n = (n+1)^2 - 10^{-9}$	$n = 1(1)10$
3	axe^{bx}	$[-9, 31]$	$a = -40,\ b = -1$ $a = -100,\ b = -2$ $a = -200,\ b = -3$
4	$x^n - a$	$[0, 5]$ $[-0.95, 4.05]$	$a = 0.2, 1,\ n = 4(2)12$ $a = 1,\ n = 8(2)14$
5	$\sin x - 0.5$	$[0, 1.5]$	
6	$2xe^{-n} - 2e^{-nx} + 1$	$[0, 1]$	$n = 1(1)5, 20(20)100$
7	$[1 + (1-n)^2]x - (1-nx)^2$	$[0, 1]$	$n = 5, 10, 20$
8	$x^2 - (1-x)^n$	$[0, 1]$	$n = 2, 5, 10, 15, 20$
9	$[1 + (1-n)^4]x - (1-nx)^4$	$[0, 1]$	$n = 1, 2, 4, 5, 8, 15, 20$
10	$e^{-nx}(x-1) + x^n$	$[0, 1]$	$n = 1, 5, 10, 15, 20$
11	$(nx-1)/((n-1)x)$	$[0.01, 1]$	$n = 2, 5, 15, 20$
12	$x^{\frac{1}{n}} - n^{\frac{1}{n}}$	$[1, 100]$	$n = 2(1)6, 7(2)33$
13	$\begin{cases} 0 & \text{if } x = 0 \\ xe^{-x^{-2}} & \text{otherwise} \end{cases}$	$[-1, 4]$	
14	$\begin{cases} \frac{n}{20}(\frac{x}{1.5} + \sin x - 1) & \text{if } x \geq 0 \\ \frac{-n}{20} & \text{otherwise} \end{cases}$	$[-10^4, \pi/2]$	$n = 1(1)40$
15	$\begin{cases} e - 1.859 & \text{if } x > \frac{2\times10^{-3}}{1+n} \\ e^{\frac{(n+1)x}{2}\times 10^3} - 1.859 & \text{if } x \in [0, \frac{2\times10^{-3}}{1+n}] \\ -0.859 & \text{if } x < 0 \end{cases}$	$[-10^4, 10^{-4}]$	$n = 20(1)40$ $n = 100(100)1000$

Due to the above termination criterion, a natural modification of the subroutine *bracket* was employed in our implementations of Algorithms 2.1–2.5 and 4.1–4.2. The modified subroutine is the following:

Subroutine *bracket*$(a, b, c, \bar{a}, \bar{b})$ (or *bracket*$(a, b, c, \bar{a}, \bar{b}, d)$)

set $\delta = \lambda \cdot tole(a, b)$ for some user-given fixed $\lambda \in (0, 1)$ (in our experiments we took $\lambda = 0.7$).
  if $b - a \leq 4\delta$, then set $c = (a + b)/2$, goto 10;
  if $c \leq a + 2\delta$, then set $c = a + 2\delta$, goto 10;
  if $c \geq b - 2\delta$, then set $c = b - 2\delta$, goto 10;
10 if $f(c) = 0$, then print c and terminate;
  if $f(a)f(c) < 0$, then $\bar{a} = a$, $\bar{b} = c$, $(d = b)$;
  if $f(b)f(c) < 0$, then $\bar{a} = c$, $\bar{b} = b$, $(d = a)$;
  calculate $tole(\bar{a}, \bar{b})$;
  if $\bar{b} - \bar{a} \leq 2 \cdot tole(\bar{a}, \bar{b})$, then terminate.

In our experiments we tested all the problems listed in Table I with different user-given *tol* ($tol = 10^{-7}$, 10^{-10}, 10^{-15}, and 0). The total number of function evaluations in solving all the problems (154 cases) are listed in Table II, where BR, DE, M, R, and LE stand for Brent's method, Dekker's method, Algorithms M and R of Bus and Dekker, and Le's method, respectively, and

Table II. Total Number of Function Evaluations in Solving All the Problems Listed in Table I

tol	BR	DE	M	R	LE	2.1	2.2	2.3	2.4	2.5	4.1	4.2
10^{-7}	2804	2808 1 un	2839	7630	2694	3154	2950	2645	2791	2687	2696	2650
10^{-10}	2905	2963 1 un	2992	7768	2821	3338	3060	2789	2922	2819	2835	2786
10^{-15}	2975	3196 1 un	3261	8014	3061	3448	3151	2948	3015	2914	2908	2859
0	3008	2998 15un	3146 11un	8230	3165	3509	3219	3029	3060	2954	2950	2884

Table III. Total Number of Function Evaluations in Solving the 139 Cases that are Solvable by All Methods

tol	BR	DE	M	R	LE	2.1	2.2	2.3	2.4	2.5	4.1	4.2
10^{-7}	2501	2528	2527	6830	2412	2796	2588	2341	2464	2382	2377	2347
10^{-10}	2589	2666	2663	6952	2529	2957	2682	2464	2576	2501	2499	2469
10^{-15}	2651	2874	2903	7184	2756	3052	2762	2615	2664	2577	2570	2535
0	2674	2998	3035	7349	2835	3094	2820	2690	2696	2598	2600	2554

"un" stands for "unsolved" meaning that a problem is not solved within 1000 iterations. From there we see that Algorithms 4.1 and 4.2 compare well with the other 10 methods. The Algorithm 4.2 in this article has the best behavior, especially when the termination tolerance is small. This reconfirms the fact that the efficiency index is an asymptotic notion.

In our experiments we noticed that problem (13) was not solved by Dekker's method within 1000 iterations. Furthermore, when $tol = 0$, there were 15 cases unsolved by Dekker's method and 11 cases (among those 15) unsolved by the Algorithm M of Bus and Dekker. To make the comparison more informative we tested the 139 cases that were solvable (within 1000 iterations) by all the 12 methods. The results are listed in Table III.

We also mention that the functions behave quite differently around the calculated zeros. In fact, problems (3), (13), (14), and (15) require many more function evaluations than others. In particular, the Algorithm R of Bus and Dekker behaves very badly on problems (14) and (15), while Dekker's method did not solve (13) (within 1000 iterations) at all. To clarify these situations, we tested three groups, each representing a subset of the problem set listed in Table I. The first group contains only problem (13). The second group represents (3), (14), and (15). The third group represents the rest of the problems. The number of function evaluations for each case with $tol = 10^{-15}$, as well as the total number of function evaluations for each group, is listed in Tables IV–VI, respectively.

Finally, it is interesting to mention that with problem (13) care is needed when coding the function. In this case,

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

Table IV. Number of Function Evaluations in Solving Problem (13) when $tol = 10^{-15}$

BR	DE	M	R	LE	2.1	2.2	2.3	2.4	2.5	4.1	4.2
23	un	32	28	16	24	31	19	27	23	28	29

Table V. Number of Function Evaluations in Solving the Second Group of Representative Cases when $tol = 10^{-15}$

Prob.	Para.	BR	DE	M	R	LE	2.1	2.2	2.3	2.4	2.5	4.1	4.2
#3	a=-100 b=-2	19	20	20	18	16	29	34	25	26	27	25	24
#14	n=10	21	23	23	67	21	23	20	18	20	19	20	19
#14	n=30	21	23	23	67	21	23	19	18	20	19	20	19
#15	n=30	36	36	36	136	35	38	33	29	29	32	29	31
#15	n=500	39	39	39	139	40	41	37	34	33	34	35	35
Total		136	141	141	427	133	154	143	124	128	131	129	128

'Para.' stands for 'parameter'.

Table VI. Number of Function Evaluations in Solving the Third Group of Representative Cases when $tol = 10^{-15}$

Prob.	Para.	BR	DE	M	R	LE	2.1	2.2	2.3	2.4	2.5	4.1	4.2
#1		9	10	10	9	9	11	9	11	10	9	10	10
#2	n=2	10	10	10	9	11	18	18	17	17	12	15	11
#4	a=1 n=4 on [0,5]	15	16	16	14	12	18	20	16	12	13	12	13
#5		10	10	10	9	9	11	9	10	10	8	11	10
#6	n=20	13	13	13	15	12	15	13	15	12	11	12	11
#7	n=10	9	9	9	9	7	11	5	5	6	5	7	7
#8	n=10	11	11	11	11	11	15	15	17	14	15	12	11
#9	n=1	10	10	10	9	10	12	11	11	11	11	11	9
#10	n=5	9	9	9	9	9	15	14	14	14	11	12	9
#11	n=20	14	15	15	9	14	21	21	20	18	21	17	18
#12	n=3	10	13	13	13	11	13	10	13	12	11	6	5
Total		120	126	126	116	115	160	145	149	136	127	125	114

'Para.' stands for 'parameter'.

and the initial interval is $[-1, 4]$. If we code $xe^{-x^{-2}}$ in Fortran 77 as $x \cdot (e^{-1/x^2})$ then all 11 algorithms that solve this problem within 1000 iterations deliver values around 0.02 as the exact solution, because the result of the computation of $0.02 \cdot (e^{-1/(0.02)^2})$ on our machine is equal to 0. However, when we code $xe^{-x^{-2}}$ as $x/e^{1/x^2}$, all algorithms give correct solutions. The same is true when we tried to use Dekker's method to solve this problem with a larger tolerance such as $tol = 10^{-3}$.

ACKNOWLEDGMENTS

We thank D. Le for kindly sending us the Fortran code of his Algorithm LZ4. We are also grateful to the referees for valuable comments on an earlier version of this article.

REFERENCES

ALEFELD, G. AND POTRA, F. A. 1988. On two higher order enclosing methods of J. W. Schmidt. Z. angew. *Math. Mech. 68*, 8, 331–337.

ALEFELD, G. AND POTRA, F. A. 1992. Some efficient methods for enclosing simple zeros of nonlinear equations. *BIT 32*, 334–344.

ALEFELD, G., POTRA, F. A., AND SHI, Y. 1993. On enclosing simple roots of nonlinear equations. *Math. Comput. 61*, 733–744

ANDERSON, N. AND BJÖRCK, A. 1973. A new high order method of regula falsi type for computing a root of an equation. *BIT 13*, 253–264.

ATKINSON, K. 1989. *An Introduction to Numerical Analysis*. John Wiley & Sons, New York.

BRENT, R. P. 1972. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, Englewood Cliffs, N.J.

BUS, J. C. P. AND DEKKER, T. J. 1975. Two efficient algorithms with guaranteed convergence for finding a zero of a function. *ACM Trans. Math. Softw. 1*, 330–345.

DEKKER, T. J. 1969. 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.

KING, R. F. 1976. Methods without secant steps for finding a bracketed root. *Computing 17*, 49–57.

LE, D. 1985. An efficient derivative-free method for solving nonlinear equations. *ACM Trans. Math. Softw. 11*, 250–262.

OSTROWSKI, A. M. 1973. *Solution of Equations in Banach Spaces*. Academic Press, New York.

POTRA, F. A. 1989. On Q-order and R-order of convergence *J. Optim. Theor. Appl. 63*, 415–431.

SCHMIDT, J. W. 1971. Eingrenzung der Lösungen nichtlinearer Gleichungen mit höherer Konvergenzgeschwindigkeit. *Computing 8*, 208–215.

STOER, J. AND BULIRSCH, R. 1980. *Introduction to Numerical Analysis*. Springer-Verlag, New York

Received May 1993; revised August 1994; accepted August 1994