Reliable Computing 2 (4) (1996), pp. 357–372

Improving the efficiency index in enclosing a root of an equation

YIXUN SHI

Recently several algorithms have been developed which achieve high efficiency index in enclosing a root of the equation $f(x) = 0$ in an interval $[a, b]$ over which $f(x)$ is continuous and $f(a)f(b) < 0$. The highest efficiency index, $1.6686\ldots$, was achieved in [4] using the inverse cubic interpolation. This paper studies the possibility of improving efficiency index by using high order inverse interpolations. A class of algorithms are presented and the optimal one of the class has achieved the efficiency index $1.7282\ldots$ With a user-given accurary ε and starting with the initial interval $[a_1, b_1] = [a, b]$, these algorithms guarantee to find in finitely many iterations an enclosing interval $[a_n, b_n]$ that contains a root of the equation and whose length $b_n - a_n$ is smaller than ε. Numerical experiments indicate that the new algorithm performs very well in practice.

Повышение индекса эффективности нахождения оценки корня уравнения

Йиксун Ши

В последнее время было разработано несколько алгоритмов, позволяющих достичь высокого индекса эффективности при нахождении оценки корня уравнения $f(x) = 0$ в интервале $[a, b]$, на котором функция $f(x)$ непрерывна и $f(a)f(b) < 0$. Наилучший индекс эффективности, равный $1.6686\ldots$, был достигнут в работе [4] с использованием обратной кубической интерполяции. В настоящей работе исследуется возможность дальнейшего улучшения индекса эффективности с использованием обратной интерполяции более высокого порядка. Представлен класс алгоритмов, наилучший представитель которого достигает индекса эффективности $1.7282\ldots$ Начав с исходного интервала $[a_1, b_1] = [a, b]$, эти алгоритмы гарантированно находят за конечное число итераций включающий интервал $[a_n, b_n]$, содержащий корень уравнения, чья ширина $b_n - a_n$ не превышает заданной пользователем величины погрешности ε. Численные эксперименты показывают, что производительность данного алгоритма на практических задачах достаточно велика.

1. Introduction

Recently several algorithms have been developed in [2–4] which achieve high efficiency index, in the sense of Ostrowski [9], in enclosing a root x_* of the equation

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

in an interval $[a, b]$, where $f(x)$ is continuous over $[a, b]$ and $f(a)f(b) < 0$. Starting with the initial enclosing interval $[a_1, b_1] = [a, b]$, these algorithms produce a sequence of intervals $\{[a_n, b_n]\}_{n=1}^{\infty}$ such that

$$x_* \in [a_{n+1}, b_{n+1}] \subseteq [a_n, b_n] \subseteq \cdots \subseteq [a_1, b_1] = [a, b],$$
$$\lim_{n\to\infty}(b_n - a_n) = 0.$$

Let us first give the definition of efficiency index referred throughout this paper. The following definitions are also given in [10], where $\{\varepsilon_n\}$ is a sequence of positive numbers such that $\lim_{n\to\infty} \varepsilon_n = 0$.

Definition 1.

1. *$\{\varepsilon_n\}$ converges with Q-order $\tau > 1$ if there are two positive constants m and M such that $m\varepsilon_n^\tau \le \varepsilon_{n+1} \le M\varepsilon_n^\tau$ for all n;*
2. *$\{\varepsilon_n\}$ converges with R-order $\tau > 1$ if there are two positive constants m and M and two sequences $\{\xi_n\}$ and $\{\eta_n\}$ that converge to zero with Q-order τ such that $m\xi_n \le \varepsilon_n \le M\eta_n$ for all n;*
3. *If an algorithm produces a sequence of enclosing intervals $\{[a_n, b_n]\}_{n=1}^{\infty}$ such that $(b_n - a_n)$ converges to zero with R-order or Q-order $\tau > 1$, and if asymptotically k function evaluations are required in each iteration, then the efficiency index of the algorithm equals $\tau^{1/k}$.*

Obviously, if a sequence converges to zero with Q-order τ then it also has the R-order τ. Combining this fact with the above definition, one sees that roughly speaking Q-order and R-order "equally well" describe the convergence speed of an algorithm. This is why the efficiency index is universally defined for both Q-order and R-order. The significance of the efficiency index is that it describes the asymptotic average improvement obtained from each function evaluation. In other words, this is a measure of "gain versus cost". The purpose of this paper is to propose new algorithms that achieve higher efficiency index while guarantee to approximate the root to any given accuracy in finitely many iterations.

Among the algorithms developed in [2–4], the Algorithm 4.2 of [4] has achieved the highest efficiency index $1.6686\ldots$ by using the inverse cubic interpolation. Numerical experiments show that these algorithms compare well with the efficient solvers of Dekker [7], Bus and Dekker [6], Brent [5], and Le [8]. The Algorithm 4.2 of [4] has the best behavior in the experiments. The basic idea of this algorithm, which is described as the Algorithm 1 in this section, is to repeatedly use the inverse cubic interpolation in Steps 1.3 and 1.5. In these two steps, either an inverse cubic interpolation is applied or an approximate quadratic interpolation in employed. It is proved in [4] that asymptotically the inverse cubic interpolation will always be applied and thus higher efficiency index is achieved. Steps 1.7 and 1.8 form a double-size secant step. Together with Steps 1.9–1.11 they guarantee the convergence of the algorithm as well as a high efficiency index. Please see [4] for details.

Before giving Algorithm 1, let us first list out two subroutines *bracket* and *Newton-Quadratic* that are being called by the algorithm. The inputs a, b, c for the subroutine *bracket* are such that $c \in (a, b)$, $f(x)$ is continuous on $[a, b]$, and $f(a)f(b) < 0$.

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

compute $f(c)$;

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 $\bar{a} = c,\ \bar{b} = b,\ d = a$. #

Newton-Quadratic has a, b, d, and k as inputs and r as output. $f(x)$ is continuous on $[a, b]$ and $f(a)f(b) < 0$. It is also 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

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

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 $P(a)P(b) < 0$.

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:

$$r_i = r_{i-1} - \frac{P(r_{i-1})}{P'(r_{i-1})} = r_{i-1} - \frac{P(r_{i-1})}{B + A(2r_{i-1} - a - b)}$$

$r = r_k$. #

We are now in the position to describe the following Algorithm 1.

Algorithm 1 (Algorithm 4.2 of [4]).

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

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

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

1.3 if $n = 2$ or if $n > 2$ but $\prod_{i \neq j}(f_i - f_j) = 0$ (where $f_1 = f(a_n)$, $f_2 = f(b_n)$, $f_3 = f(d_n)$, $f_4 = f(e_n)$) then call *Newton-Quadratic*$(a_n, b_n, d_n, c_n, 2)$ and goto Step 1.4.

Otherwise compute $c_n = IP_1(0)$ where $IP_1(y)$ is the polynomial obtained by the inverse cubic interpolation at the points $(a_n, f(a_n))$, $(b_n, f(b_n))$, $(d_n, f(d_n))$, and $(e_n, f(e_n))$. If $(c_n - a_n)(c_n - b_n) \geq 0$, then call *Newton-Quadratic*$(a_n, b_n, d_n, c_n, 2)$. Goto Step 1.4.

1.4 set $\bar{e}_n = d_n$, call *bracket*$(a_n, b_n, c_n, \bar{a}_n, \bar{b}_n, \bar{d}_n)$;

1.5 if $\prod_{i \neq j}(\bar{f}_i - \bar{f}_j) = 0$ (where $\bar{f}_1 = f(\bar{a}_n)$, $\bar{f}_2 = f(\bar{b}_n)$, $\bar{f}_3 = f(\bar{d}_n)$, and $\bar{f}_4 = f(\bar{e}_n)$) then call *Newton-Quadratic*$(\bar{a}_n, \bar{b}_n, \bar{d}_n, \bar{c}_n, 3)$ and goto Step 1.6.

Otherwise compute $\bar{c}_n = IP_2(0)$ where $IP_2(y)$ is the polynomial obtained by the inverse cubic interpolation at the points $(\bar{a}_n, f(\bar{a}_n))$, $(\bar{b}_n, f(\bar{b}_n))$, $(\bar{d}_n, f(\bar{d}_n))$, and $(\bar{e}_n, f(\bar{e}_n))$. If $(\bar{c}_n - \bar{a}_n)(\bar{c}_n - \bar{b}_n) \geq 0$, then call *Newton-Quadratic*$(\bar{a}_n, \bar{b}_n, \bar{d}_n, \bar{c}_n, 3)$. Goto Step 1.6.

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

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

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

1.9 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 $\hat{c}_n = \bar{c}_n$;

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

1.11 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. #

The idea used in Algorithm 1 to achieve the higher efficiency index is to employ the inverse cubic interpolation instead of classical linear or quadratic interpolations whenever possible. Thus it becomes interesting to study the possibility of improving the efficiency index by applying higher order inverse interpolations. In this paper, we propose a class of enclosing algorithms which, in the n-th iteration, uses all the function values computed in the previous iteration as well as those already computed in the current iteration to form an inverse interpolation with the highest possible order. With a user-given accurary ε and starting with the initial interval $[a_1, b_1] = [a, b]$, these algorithms guarantee to find in finitely many iterations an enclosing interval $[a_n, b_n]$ that contains a root of the equation and whose length $b_n - a_n$ is smaller than ε. The optimal algorithm of this class has achieved the efficiency index $1.7282\ldots$ The algorithms are presented in the next section. In Section 3 the results on efficiency index are derived. Numerical experiments are reported in Section 4.

2. Algorithm

In this section we present a class of algorithms, universally described as the following Algorithm 2, for enclosing a root x_* of (1) in an interval $[a, b]$, where $f(x)$ is continuous over $[a, b]$ and $f(a)f(b) < 0$.

The basic idea used in Algorithm 2 is that in the n-th iteration, the algorithm uses all the function values computed in the $(n-1)$-th iteration as well as those already computed in the current iteration to form and apply, whenever possible, the corresponding high order inverse interpolation. When that is not possible, an approximate quadratic interpolation is used by calling the subroutine *Newton-Quadratic* described in Section 1. It is proved in Section 3 that asymptotically the inverse interpolation will always be applied and thus a high efficiency index may be achieved. This idea is implemented in Step 2.3. Each algorithm of this class is associated with an integer parameter k such that $k \geq 4$. At the n-th iteration when $n \geq k$, the inverse interpolation (or an approximate quadratic interpolation, but asymptotically always the inverse interpolation) is repeated for $k-3$ times. A more detailed discussion is provided after the presentation of Algorithm 2. The algorithm also needs to call the subroutine *bracket*. There is another parameter μ such that $\mu \in (0,1)$, usually chosen as $\mu = 0.5$. For convenience, let us give the following definition.

Definition 2. *Suppose $x_1, x_2, \ldots, x_j$ are j distinct values and so are the function values $f(x_1), f(x_2), \ldots, f(x_j)$. Suppose $IP(y)$ is the polynomial of degree $j-1$ obtained by the inverse interpolation at the points $(x_1, f(x_1)), (x_2, f(x_2)), \ldots, (x_j, f(x_j))$. We say that $\bar{x}$ is obtained by the inverse interpolation at $x_1, x_2, \ldots, x_j$ if*

$$\bar{x} = IP(0). \tag{2}$$

We also need to introduce some notations used in Algorithm 2. In the following Algorithm 2, the current enclosing interval at the begining of a general iteration, say n-th iteration with $n \geq k$, is denoted by $[a_n, b_n]$. After Step 2.3 an intermediate interval $[\bar{a}_n, \bar{b}_n]$ is obtained. Then at Step 2.7 we get $[\hat{a}_n, \hat{b}_n]$. From that we obtain $[a_{n+1}, b_{n+1}]$ at Step 2.8. They satisfy that

$$[a_{n+1}, b_{n+1}] \subseteq [\hat{a}_n, \hat{b}_n] \subseteq [\bar{a}_n, \bar{b}_n] \subseteq [a_n, b_n].$$

More notations such as $a_n^{(i)}$, $b_n^{(i)}$, $d_n^{(i)}$ are used in Steps 2.3 and 2.9. Here $a_n^{(i)}$ and $b_n^{(i)}$ satisfy that

$$[\bar{a}_n, \bar{b}_n] = [a_n^{(k-2)}, b_n^{(k-2)}] \subseteq \cdots \subseteq [a_n^{(1)}, b_n^{(1)}] = [a_n, b_n]$$

while $d_n^{(i)}$ are generated in the procedure for use in the next iteration as explaned after the presentation of the algorithm.

Algorithm 2.

2.1 set $a_1 = a$, $b_1 = b$, $c_1 = a_1 - f(a_1)/f[a_1, b_1]$;

2.2 call *bracket*$(a_1, b_1, c_1, a_2, b_2, d_1^{(1)})$;

For $n = 2, 3, \ldots$ execute Step 2.3 through to Step 2.9:

2.3 execute the computations below:

2.3.1 if $n = 2$ then

call *Newton-Quadratic*$(a_2, b_2, d_1^{(1)}, temp, 2)$;

call *bracket*$(a_2, b_2, temp, \bar{a}_2, \bar{b}_2, d_2^{(1)})$;

goto Step 2.4;

2.3.2 if $n = 3$ then

if $f(a_3)$, $f(b_3)$, $f(d_2^{(1)})$, $f(d_2^{(2)})$ are distinct and if $\bar{x}$ obtained by the inverse interpolation at a_3, b_3, $d_2^{(1)}$, $d_2^{(2)}$ satisfies $\bar{x} \in (a_3, b_3)$, then $temp = \bar{x}$. Otherwise call *Newton-Quadratic*$(a_3, b_3, d_2^{(2)}, temp, 2)$;

call *bracket*$(a_3, b_3, temp, \bar{a}_3, \bar{b}_3, d_3^{(1)})$;

goto Step 2.4;

2.3.3 if $3 < n \leq k-1$ then set $a_n^{(1)} = a_n$, $b_n^{(1)} = b_n$, and $d_n^{(0)} = d_{n-1}^{(n-2)}$.

For $i = 1, 2, \ldots, n-2$ do:

if $f(a_n^{(i)})$, $f(b_n^{(i)})$, $f(d_n^{(0)}) (= f(d_{n-1}^{(n-2)}))$, $f(d_n^{(1)}), \ldots, f(d_n^{(i-1)})$, $f(d_{n-1}^{(1)}), \ldots, f(d_{n-1}^{(n-3)})$ are distinct and if $\bar{x}$ obtained by the inverse interpolation at $a_n^{(i)}$, $b_n^{(i)}$, $d_n^{(0)} (= d_{n-1}^{(n-2)})$, $d_n^{(1)}, \ldots, d_n^{(i-1)}$, $d_{n-1}^{(1)}, \ldots, d_{n-1}^{(n-3)}$ satisfies that $\bar{x} \in (a_n^{(i)}, b_n^{(i)})$, then $temp = \bar{x}$. Otherwise call *Newton-Quadratic*$(a_n^{(i)}, b_n^{(i)}, d_n^{(i-1)}, temp, 2)$;

call *bracket*$(a_n^{(i)}, b_n^{(i)}, temp, a_n^{(i+1)}, b_n^{(i+1)}, d_n^{(i)})$;

end do;

$\bar{a}_n = a_n^{(n-1)}$, $\bar{b}_n = b_n^{(n-1)}$, goto Step 2.4;

2.3.4 if $n \geq k$ then set $a_n^{(1)} = a_n$, $b_n^{(1)} = b_n$, and $d_n^{(0)} = d_{n-1}^{(k-2)}$.

For $i = 1, 2, \ldots k-3$ do:

if $f(a_n^{(i)})$, $f(b_n^{(i)})$, $f(d_n^{(0)}) (= f(d_{n-1}^{(k-2)}))$, $f(d_n^{(1)}), \ldots, f(d_n^{(i-1)})$, $f(d_{n-1}^{(1)}), \ldots, f(d_{n-1}^{(k-3)})$

are distinct and if $\bar{x}$ obtained by the inverse interpolation at $a_n^{(i)}$, $b_n^{(i)}$, $d_n^{(0)}$ $(= d_{n-1}^{(k-2)})$, $d_n^{(1)}, \ldots, d_n^{(i-1)}$, $d_{n-1}^{(1)}, \ldots, d_{n-1}^{(k-3)}$ satisfies that $\bar{x} \in (a_n^{(i)}, b_n^{(i)})$, then *temp* $= \bar{x}$. Otherwise call *Newton-Quadratic*$(a_n^{(i)}, b_n^{(i)}, d_n^{(i-1)}, temp, 2)$;

call *bracket* $(a_n^{(i)}, b_n^{(i)}, temp, a_n^{(i+1)}, b_n^{(i+1)}, d_n^{(i)})$;

end do;

$\bar{a}_n = a_n^{(k-2)}$, $\bar{b}_n = b_n^{(k-2)}$, goto Step 2.4;

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

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

2.6 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 $\hat{c}_n = \bar{c}_n$;

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

2.8 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*$(\hat{a}_n, \hat{b}_n, 0.5(\hat{a}_n + \hat{b}_n), a_{n+1}, b_{n+1}, \hat{d}_n)$,
endif;

2.9 if $n = 2$, set $d_2^{(2)} = \hat{d}_2$,

if $3 \le n \le k-1$, set $d_n^{(n-1)} = \hat{d}_n$,

if $n \ge k$, set $d_n^{(k-2)} = \hat{d}_n$. #

We see that Step 2.8 guarantees that

$$b_{n+1} - a_{n+1} \le \mu_1(b_n - a_n) \tag{3}$$

with $\mu_1 = \max\{\mu, 0.5\} < 1$. Hence, either a root of (1) is found in a finite number of iterations, or there is a root x_* of (1) in $[a, b]$ such that

$$x_* \in [a_{n+1}, b_{n+1}] \subseteq [a_n, b_n] \subseteq \cdots \subseteq [a_1, b_1] = [a, b] \tag{4}$$

and

$$b_n - a_n \longrightarrow 0 \tag{5}$$

with at least linear convergence. In this case, for any user-given accuracy ε, the algorithm obtains in finitely many iterations an enclosing interval $[a_n, b_n]$ such that $b_n - a_n < \varepsilon$.

At the end of the n-th iteration when $3 < n \le k-1$, $n+1$ points $(a_{n+1}, f(a_{n+1}))$, $(b_{n+1}, f(b_{n+1}))$, $(d_n^{(1)}, f(d_n^{(1)})), \ldots, (d_n^{(n-1)}, f(d_n^{(n-1)}))$ are available for the use in the next iteration. They satisfy that

$$\{a_n, b_n\} \subseteq \{a_{n+1}, b_{n+1}, d_n^{(1)}, \ldots, d_n^{(n-1)}\}, \tag{6}$$

$$\{a_{n+1}, b_{n+1}, d_n^{(1)}, \ldots, d_n^{(n-1)}\} \subseteq [a_n, b_n] \tag{7}$$

and

$$d_n^{(i)} \notin [a_{n+1}, b_{n+1}], \quad \forall i = 1, 2, \ldots, n-1. \tag{8}$$

Therefore at the begining of k-th iteration, k points $(a_k, f(a_k))$, $(b_k, f(b_k))$, $(d_{k-1}^{(1)}, f(d_{k-1}^{(1)})), \ldots,$ $(d_{k-1}^{(k-2)}, f(d_{k-1}^{(k-2)}))$ are available, for which (6)–(8) hold with $n = k - 1$.

Starting with k-th iteration, Algorithm 2 computes only $k - 3$ points at Step 2.3 in each iteration. Therefore at the end of the n-th iteration when $n \geq k$, k points $(a_{n+1}, f(a_{n+1}))$, $(b_{n+1}, f(b_{n+1}))$, $(d_n^{(1)}, f(d_n^{(1)})), \ldots, (d_n^{(k-2)}, f(d_n^{(k-2)}))$ are ready to be used in the next iteration. (6)–(8) remain true if we replace $d_n^{(n-1)}$ by $d_n^{(k-2)}$. To sum it up, in the n-th iteration when $n \geq k$,

(I) k points $(a_n, f(a_n))$, $(b_n, f(b_n))$, $(d_{n-1}^{(1)}, f(d_{n-1}^{(1)})), \ldots, (d_{n-1}^{(k-2)}, f(d_{n-1}^{(k-2)}))$ are carried over from the previous iteration;

(II) $k - 3$ new points are computed in Step 2.3, each is obtained, whenever possible, by using the inverse interpolation at the k carried-over points as well as the points already computed in Step 2.3 of the current iteration;

(III) One new point is computed in Steps 2.7–2.9. This point may cost an additional function evaluation at Step 2.8. However, in next section we will show that when n is big enough,

$$\hat{b}_n - \hat{a}_n < \mu(b_n - a_n)$$

always holds. Therefore, asymptotically Algorithm 2 requires only $k-2$ function evaluations per iteration;

(IV) The points $(a_n, f(a_n))$, $(b_n, f(b_n))$, plus the $k - 2$ points computed in the n-th iteration, form the group of k points:

$$(a_{n+1}, f(a_{n+1})),\ (b_{n+1}, f(b_{n+1})),\ (d_n^{(1)}, f(d_n^{(1)})), \ldots, (d_n^{(k-2)}, f(d_n^{(k-2)}))$$

for the use in $(n + 1)$-th iteration.

3. Efficiency index of Algorithm 2

In this section we show that under certain smoothness assumptions the asymtotic efficiency index of Algorithm 2 is

$$I_k = \left[(k-3)(k-2)/4 + 1/2 + \sqrt{(k-3)(k-2) + ((k-3)(k-2)/4 + 1/2)^2}\right]^{\frac{1}{k-2}} \tag{9}$$

for each integer $k \geq 4$. We will also show that $I_k \leq I_5$ for all $k \geq 4$. Hence $k = 5$ yields the optimal procedure of this class, achieving the efficiency index $I_5 = 1.7282\ldots$ In this case at most four and asymptotically only three function evaluations are needed in each iteration. The total number of function evaluations thus will be bounded by four times of that needed by the bisection method.

In the rest of this section, the following assumptions (A), (B), and (C) are assumed to be true.

(A) $f(x)$ is continuously differentiable in $[a, b]$ and $f(a)f(b) < 0$.

(B) x_* is a simple zero of $f(x)$ in $[a, b]$.

(C) *Algorithm 2 does not terminate after a finite number of iterations. (4) and (5), plus assumptions (A) and (B), then imply that $f'(x) \neq 0$ in $[a_n, b_n]$ when n is big enough. Therefore without loss of generality we assume that $f'(x) \neq 0$ in $[a, b]$.*

We first prove the following Lemma 1.

Lemma 1. *Under assumptions (A), (B), (C), also assume that $f(x)$ is j times continuously differentiable in $[a, b]$. Suppose $\{x_1, \ldots, x_j\} \subseteq [a, b]$ and also suppose that $\bar{x}$ is obtained by the inverse interpolation at $x_1, \ldots, x_j$, then there is a constant number M_j, independent of $x_1, \ldots, x_j$, such that*

$$|\bar{x} - x_*| \leq M_j |f(x_1)| \ldots |f(x_j)|. \tag{10}$$

Proof. Since we assume that $f'(x) \neq 0$ in $[a, b]$, the inverse function $f^{-1}(y)$ exists for $y \in f([a, b])$ where $f([a, b])$ stands for the image of $[a, b]$ under the function $f(x)$. It is clear that for all $y = f(x) \in f([a, b])$,

$$[f^{-1}(y)]' = \frac{1}{f'(x)}$$

and

$$[f^{-1}(y)]'' = \frac{-f''(x)}{(f'(x))^3}.$$

For $l < j$, suppose

$$[f^{-1}(y)]^{(l)} = \frac{P_l(x)}{(f'(x))^{2l-1}}$$

where $P_l(x)$ is a polynomial of $f'(x), f''(x), \ldots, f^{(l)}(x)$. Then

$$[f^{-1}(y)]^{(l+1)} = \frac{f'(x)P_l'(x) - (2l-1)f''(x)P_l(x)}{(f'(x))^{2l+1}} = \frac{P_{l+1}(x)}{(f'(x))^{2(l+1)-1}}$$

with $P_{l+1}(x)$ being a polynomial of $f'(x), f''(x), \ldots, f^{(l+1)}(x)$. Hence by induction we see that for any $y = f(x) \in f([a, b])$, $[f^{-1}(y)]^{(j)}$ exists and

$$[f^{-1}(y)]^{(j)} = \frac{P_j(x)}{(f'(x))^{2j-1}} \tag{11}$$

where $P_j(x)$ is a polynomial of $f'(x), f''(x), \ldots, f^{(j)}(x)$. Since $f(x)$ is j times continuously differentiable, above arguments indicate that $f^{-1}(y)$ is also j times continuous in $f([a, b])$. The facts that $x_* = f^{-1}(0)$ and $\bar{x} = IP(0)$ (where $IP(y)$ is the inverse interpolation polynomial at $x_1, \ldots, x_j$) imply that

$$|\bar{x} - x_*| \leq M_j |f(x_1)| \ldots |f(x_j)|$$

with

$$M_j = \left(\max_{y \in f([a,b])} \left| [f^{-1}(y)]^{(j)} \right| \right) / (j!).$$

(10) is therefore proved. □

Lemma 2. *Under assumptions (A), (B), (C), also assume that $f(x)$ is $2k - 4$ times continuously differentiable in $[a, b]$. Then there is an $r > 0$ and an integer $N \geq k$ such that the high order* *inverse interpolations are always used at Step 2.3 of n-th iteration when $n \geq N$, and that the u_n obtained at Step 2.4 satisfies that*

$$|f(u_n)| \leq r(b_n - a_n)^{\alpha}(b_{n-1} - a_{n-1})^{\beta}, \quad \forall n \geq N \tag{12}$$

with $\alpha = \frac{(k-3)(k-2)}{2} + 1$ *and* $\beta = (k-3)(k-2)$.

Proof. Consider $n \geq k$ and $i \in \{1, 2, \ldots, k-3\}$. Since we assume that $f'(x) \neq 0$ in $[a, b]$, $f(x)$ is monotone and thus all the function values involved in Step 2.3 are distinct. Therefore we only need to prove that when n is big enough

$$\bar{x}_i \in (a_n^{(i)}, b_n^{(i)}), \quad \forall i = 1, 2, \ldots, k-3 \tag{13}$$

where $\bar{x}_i$ is obtained by the inverse interpolation at $a_n^{(i)}$, $b_n^{(i)}$, $d_n^{(0)}(= d_{n-1}^{(k-2)})$, $d_n^{(1)}, \ldots, d_n^{(i-1)}$, $d_{n-1}^{(1)}, \ldots, d_{n-1}^{(k-3)}$.

By Lemma 1 we see that

$$\begin{aligned}
|\bar{x}_i - x_*| &\leq M_{k+i-1}|f(a_n^{(i)})|\,|f(b_n^{(i)})|\,|f(d_n^{(0)})|\,|f(d_n^{(1)})| \ldots |f(d_n^{(i-1)})| \\
&\quad \times |f(d_{n-1}^{(1)})| \ldots |f(d_{n-1}^{(k-3)})| \\
&= M_{k+i-1}|f(a_n^{(i)})|\,|f(b_n^{(i)})|\,|f(d_n^{(1)})| \ldots |f(d_n^{(i-1)})| \\
&\quad \times |f(d_{n-1}^{(1)})| \ldots |f(d_{n-1}^{(k-3)})|\,|f(d_{n-1}^{(k-2)})| \\
&\leq M_{k+i-1}m^{k+i-1}(b_n - a_n)^{i+1}(b_{n-1} - a_{n-1})^{k-2}
\end{aligned} \tag{14}$$

where $m = \max_{a \leq x \leq b} |f'(x)|$. Since $x_* \in (a, b)$ and $b_n - a_n$ converges to zero, (14) implies that there is an $N_1 \geq k$ such that when $n \geq N_1$, $\bar{x}_i \in (a, b)$ for all $i = 1, 2, \ldots, k-3$. It then follows that

$$|f(\bar{x}_i)| = |f(\bar{x}_i) - f(x_*)| \leq m|\bar{x}_i - x_*|.$$

Therefore, when $n \geq N_1$, for all $i = 1, 2, \ldots, k-3$ we have

$$|f(\bar{x}_i)| \leq M_{k+i-1}m^{k+i-1}(b_n - a_n)^{i}(b_{n-1} - a_{n-1})^{k-2}|f(a_n^{(i)})| \tag{15}$$

and

$$|f(\bar{x}_i)| \leq M_{k+i-1}m^{k+i-1}(b_n - a_n)^{i}(b_{n-1} - a_{n-1})^{k-2}|f(b_n^{(i)})|. \tag{16}$$

From (15) and (16) we see that there is an $N \geq N_1$ such that when $n \geq N$

$$|f(\bar{x}_i)| < \min\{|f(a_n^{(i)})|, |f(b_n^{(i)})|\}, \quad \forall i = 1, 2, \ldots, k-3. \tag{17}$$

(13) follows immediately because $f(x)$ is monotone on $[a, b]$.

We now show that (12) holds when $n \geq N$. Let us consider Step 2.3 of the n-th iteration for $n \geq N$ and apply induction on i.

For $i = 1$, (14) indicates that

$$|\bar{x}_1 - x_*| \leq M_k m^k (b_n - a_n)^2 (b_{n-1} - a_{n-1})^{k-2}.$$

Therefore

$$\begin{aligned}
|f(\bar{x}_1)| &\leq M_k m^{k+1}(b_n - a_n)^2(b_{n-1} - a_{n-1})^{k-2} \\
&= r_1(b_n - a_n)^2(b_{n-1} - a_{n-1})^{k-2}
\end{aligned} \tag{18}$$

where $r_1 = M_k m^{k+1} > 0$.

Similarly,

$$\begin{aligned} |f(\bar{x}_2)| &\leq m|\bar{x}_2 - x_*| \\ &\leq M_{k+1}m^{k+1}(b_n - a_n)^2(b_{n-1} - a_{n-1})^{k-2}|f(\bar{x}_1)| \\ &\leq r_2(b_n - a_n)^4(b_{n-1} - a_{n-1})^{2(k-2)} \end{aligned} \tag{19}$$

where $r_2 = r_1 M_{k+1} m^{k+1} > 0$.

Suppose for $2 \leq l < k-3$ we have that

$$|f(\bar{x}_l)| \leq r_l(b_n - a_n)^{(2+2+3+\cdots+l)}(b_{n-1} - a_{n-1})^{l(k-2)} \tag{20}$$

for some $r_l > 0$, then

$$\begin{aligned} |f(\bar{x}_{l+1})| &\leq m|\bar{x}_{l+1} - x_*| \\ &\leq mM_{k+l}|f(a_n^{(l+1)})|\,|f(b_n^{(l+1)})|\,|f(d_n^{(1)})|\ldots|f(d_n^{(l)})|\,|f(d_{n-1}^{(1)})|\ldots|f(d_{n-1}^{(k-2)})| \\ &\leq M_{k+l}m^{k+l}(b_n - a_n)^{l+1}(b_{n-1} - a_{n-1})^{k-2}|f(\bar{x}_l)| \\ &\leq r_{l+1}(b_n - a_n)^{(2+2+3+\cdots+l+(l+1))}(b_{n-1} - a_{n-1})^{(l+1)(k-2)} \end{aligned} \tag{21}$$

with $r_{l+1} = r_l M_{k+l} m^{k+l} > 0$. Here we notice that $\bar{x}_l \in \{a_n^{(l+1)}, b_n^{(l+1)}, d_n^{(1)}, \ldots, d_n^{(l)}\}$.

Therefore, by induction we see that there is an $r > 0$ such that when $n \geq N$

$$\begin{aligned} |f(\bar{x}_{k-3})| &\leq r(b_n - a_n)^{(2+2+3+\cdots+(k-3))}(b_{n-1} - a_{n-1})^{(k-3)(k-2)} \\ &= r(b_n - a_n)^{\alpha}(b_{n-1} - a_{n-1})^{\beta} \end{aligned} \tag{22}$$

where $\alpha = [(k-3)(k-2)]/2 + 1$ and $\beta = (k-3)(k-2)$.

From Step 2.3 of Algorithm 2 we see that $\bar{x}_{k-3} \in \{\bar{a}_n, \bar{b}_n\}$ when $n \geq N$. From Step 2.4 we see that $|f(u_n)| = \min\{|f(\bar{a}_n)|, |f(\bar{b}_n)|\}$ for all n. Therefore $|f(u_n)| \leq |f(\bar{x}_{k-3})|$ for $n \geq N$ and (22) thus implies (12). □

The following Lemma 3 is adopted from Alefeld and Potra [2], and the same proof in [2] applies.

Lemma 3 (see Alefeld and Potra [2]). *Under assumptions (A), (B), (C), there is an n_1 such that for all $n > n_1$, $\bar{c}_n$ and u_n in Step 2.5 satisfy that*

$$f(\bar{c}_n)f(u_n) < 0. \tag{23}$$

We are now ready to prove the assymptotic convergence property of Algorithm 2.

Theorem 1. *Under the assumptions of Lemma 2, the sequence of diameters $\{(b_n - a_n)\}_{n=1}^{\infty}$ of the enclosing intervals produced by Algorithm 2 converges to zero, and there is an $L > 0$ such that*

$$b_{n+1} - a_{n+1} \leq L(b_n - a_n)^{\alpha}(b_{n-1} - a_{n-1})^{\beta}, \quad \forall n = 2, 3, \ldots \tag{24}$$

where $\alpha = [(k-3)(k-2)]/2 + 1$ and $\beta = (k-3)(k-2)$. Moreover, there is an n_2 such that for all $n > n_2$

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

Hence when $n > n_2$, Algorithm 2 requires only $k-2$ function evaluations per iteration.

Proof. Let us recall that in this section we assume without loss of generality that $f'(x) \neq 0$ for all $x \in [a, b]$. Thus we may assume that

$$m_1 = \min_{a \le x \le b} |f'(x)| > 0.$$

Consider the integers N of Lemma 2 and n_1 of Lemma 3. Let $n_2 > \max\{N, n_1\}$. Then by Lemma 3, (23) holds when $n > n_2$. From Steps 2.5–2.7 of Algorithm 2 and the fact that $u_n, \bar{c}_n \in [\bar{a}_n, \bar{b}_n]$ we see that

$$\hat{b}_n - \hat{a}_n \le |\bar{c}_n - u_n|, \quad \forall n > n_2. \tag{25}$$

From Step 2.5 we also see that

$$|\bar{c}_n - u_n| = |2f[\bar{a}_n, \bar{b}_n]^{-1} f(u_n)| \le \tfrac{2}{m_1}|f(u_n)|. \tag{26}$$

(25), (26), and (12) now imply that

$$\hat{b}_n - \hat{a}_n \le \frac{2r}{m_1}(b_n - a_n)^\alpha (b_{n-1} - a_{n-1})^\beta, \quad \forall n > n_2. \tag{27}$$

Since $\{(b_n - a_n)\}_{n=1}^\infty$ converges to zero, if n_2 is large enough then

$$\hat{b}_n - \hat{a}_n < \mu(b_n - a_n), \quad \forall n > n_2.$$

This shows that for all $n > n_2$

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

Finally (24) follows by using (27) and taking

$$L \ge \max\left\{\frac{2r}{m_1}, \ \frac{(b_{n+1} - a_{n+1})}{(b_n - a_n)^\alpha (b_{n-1} - a_{n-1})^\beta}; \ n = 2, 3, \ldots, n_2\right\}.$$

The proof is therefore completed. □

Corollary. *Under the assumptions of Theorem 1,* $\{(b_n - a_n)\}_{n=1}^\infty$ *converges to zero with an R-order at least* $\alpha/2 + \sqrt{\beta + \alpha^2/4}$ *where*

$$\alpha = [(k-3)(k-2)]/2 + 1$$

and

$$\beta = (k-3)(k-2).$$

Since asymptotically Algorithm 2 requires $k - 2$ *function evaluations per iteration, its efficiency index is*

$$\begin{aligned} I_k &= \left(\alpha/2 + \sqrt{\beta + \alpha^2/4}\right)^{\frac{1}{k-2}} \\ &= \left[(k-3)(k-2)/4 + 1/2 + \sqrt{(k-3)(k-2) + ((k-3)(k-2)/4 + 1/2)^2}\right]^{\frac{1}{k-2}}. \end{aligned} \tag{28}$$

Proof. By Theorem 1, $\{\varepsilon_n\}_{n=1}^\infty$ converges to zero and for all $n = 2, 3, \ldots$

$$\varepsilon_{n+1} \le L\varepsilon_n^\alpha \varepsilon_{n-1}^\beta.$$

The result follows by invoking Theorem 2.1 of [10]. □

The next theorem indicates that the optimal procedure of this class of algorithms represented by Algorithm 2 is obtained when $k = 5$. In this case, at most four and asymptotically only three function evaluations are needed in each iteration, and the efficiency index is $I_5 = 1.7282\ldots$

Theorem 2. *Let I_k be as given in (28). Then $I_k \leq I_5$ for all $k \geq 4$.*

Proof. For $x \geq 4$, consider

$$\begin{aligned} l_1(x) &= (x-2)(x-3)/4 + 1/2, \\ l_2(x) &= \sqrt{(x-2)(x-3) + (l_1(x))^2}, \\ h(x) &= l_1(x) + l_2(x) \end{aligned}$$

and

$$g(x) = \frac{1}{x-2} \ln(h(x)).$$

Then $I_k = \exp(g(k))$ for all $k \geq 4$. It is easy to see that $h(x) > 1$ for all $x \geq 4$ and $\ln(h(x)) > 5/2$ for all $x \geq 8$. Hence when $x \geq 8$,

$$\begin{aligned} (x-2)h'(x) &= \frac{(x-2)(x-3)}{4} + \frac{(x-2)^2}{4} + \frac{(x-2)(2x-5)}{2l_2(x)} + \frac{(x-2)(2x-5)l_1(x)}{4l_2(x)} \\ &\leq \frac{(x-2)(x-3)}{4} + \frac{(x-2)^2}{4} + \frac{2(x-2)(2x-5)}{(x-2)(x-3)} + \frac{(x-2)(2x-5)}{4} \\ &= (x-2)(x-3) + \frac{x-2}{2} + 4 + \frac{2}{x-3} \\ &\leq (x-2)(x-3)\left[1 + \frac{1}{2(x-3)} + \frac{9}{2(x-2)(x-3)}\right] \\ &\leq \frac{5}{4}(x-2)(x-3) \\ &< 5l_1(x) \\ &< \frac{5}{2}h(x) \\ &< h(x)\ln(h(x)). \end{aligned}$$

Therefore

$$g'(x) = \frac{(x-2)h'(x) - h(x)\ln(h(x))}{(x-2)^2 h(x)} < 0, \quad \forall x \geq 8. \tag{29}$$

(29) implies that $I_k \leq I_8$ for all $k \geq 8$. Direct calculation shows that

$$I_5 = \max\{I_k;\ k = 4, 5, 6, 7, 8\}.$$

The theorem is thus proved. $\square$

4. Preliminary numerical experiments

The numerical results reported in [4] show that Algorithm 1 has the best behavior in comparison with several widely used equation solvers such as the algorithms of Dekker [7], Brent [5], Bus and Dekker [6], and Le [8]. In this section we present some preliminary numerical experiments comparing Algorithms 1 and 2 with $k = 5$. The parameter μ was chosen as 0.5. The machine

#	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	$\begin{cases} \frac{n}{20}(\frac{x}{1.5} + \sin x - 1) & \text{if } x \geq 0 \\ \frac{-n}{20} & \text{otherwise} \end{cases}$	$[-10^4, \pi/2]$	$n = 10, 20, 30, 40$
15	$\begin{cases} e - 1.859 & \text{if } x > \frac{2\times 10^{-3}}{1+n} \\ e^{\frac{(n+1)x}{2}\times 10^3} - 1.859 & \text{if } x \in [0, \frac{2\times 10^{-3}}{1+n}] \\ -0.859 & \text{if } x < 0 \end{cases}$	$[-10^4, 10^{-4}]$	$n = 20, 30, 40$ $n = 100(100)1000$

Table 1. Test problems

used was AT&T 3B2–1000 Model 80, and double precision was used. The test problems are listed in Table 1. The termination criterion was the one suggested by Brent [5], i.e.

$$b - a \leq 2 \cdot \mathit{tole}(a, b) \tag{30}$$

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

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

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.

Due to the above termination criterion, a natural modification of the subroutine ***bracket*** was employed in our implementation of the two algorithms. The modified subroutine is as follows.

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

set $\delta = \lambda \cdot \mathit{tole}(a, b)$ 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 $\mathit{tole}(\bar{a}, \bar{b})$;

if $\bar{b} - \bar{a} \leq 2 \cdot \mathit{tole}(\bar{a}, \bar{b})$, then terminate. #

We tested all the problems listed in Table 1 with different user-given $\underline{tol}$ ($\underline{tol} = 10^{-7}, 10^{-10}, 10^{-15}$, and 0). The total number of function evaluations in solving all the problems (100 cases) are listed in Table 2. From there we see that the performance of these two algorithms are well comparable, and the behavior of Algorithm 2 is slightly better than that of Algorithm 1.

We also tested two special problems. In one problem,

$$f(x) = x^n \quad \text{and} \quad [a, b] = [-1, 10] \tag{31}$$

with n being 5, 7, 9, 11, 13, and 15. In this case, the root $x_* = 0$ is not a simple root. Hence the assumptions in Section 3 are not satisfied. Another problem is that

$$f(x) = x^{1/n} - 1 \quad \text{and} \quad [a, b] = [0, 10] \tag{32}$$

with the same values of n. Now $x_* = 1$ is a simple root and $f^{-1}(y) = (y + 1)^n$ is a polynomial. All the assumptions of Section 3 are satisfied in this case. In both of those two cases, Algorithm 2 works much better than Algorithm 1. The corresponding numerical results are listed in Table 3 and Table 4.

In order to show the effectiveness of "improving efficiency index", we list in Table 5 the following numerical results: for Problem 15 with $n = 40$ (listed in Table 1) and $tol = 10^{-15}$, Algorithm 2 uses 31 function evaluations to obtain an enclosing interval that meets the termination criterion (30) while Algorithm 1 uses 32 function evaluations. Both algorithms start with the same initial interval whose length is about 10000. After using 21 function evaluations, Algorithm 1 obtains an enclosing interval with length 0.1104E−2 (here 0.1104E−2 stands for 0.1104×10^{-2}, and similar notations are also used below), and Algorithm 2 gets one whose length is 0.1021E−2. Table 5 lists the length of enclosing intervals obtained after each function evaluation, starting with the 21st function evaluation, upto the termination under the criterion (30). The results reconfirms the fact that "improving efficiency index" increases the ASYMPTOTIC AVERAGE improvement obtained from each function evaluation.

As a conclusion from our preliminary numerical experiments, we see that in general Algorithm 2 is very well comparable to Algorithm 1. We have also considered two special cases: Problem (31) whose solution is not a simple root and thus the assumptions in Lemma 1—Corollary are not satisfied, and Problem (32) where all assumptions in Section 3 are satisfied. In both cases Algorithm 2 works much better than Algorithm 1. From Table 5 we also see that with a higher efficiency index, the Algorithm 2 does have a higher asymptotic

tol	10^{-7}	10^{-10}	10^{-15}	0
Alg. 1	1480	1555	1609	1631
Alg. 2	1462	1529	1597	1627

Table 2. Total number of function evaluations in solving all the problems listed in Table 1

tol	10^{-7}	10^{-10}	10^{-15}	0
Alg. 1	470	656	895	2143
Alg. 2	385	482	735	1715

Table 3. Total number of function evaluations in solving problem (31) with $n = 5, 7, 9, 11, 13, 15$

tol	10^{-7}	10^{-10}	10^{-15}	0
Alg. 1	78	82	87	87
Alg. 2	72	73	74	75

Table 4. Total number of function evaluations in solving problem (32) with $n = 5, 7, 9, 11, 13, 15$

function evaluation	Algorithm 1	Algorithm 2
21st	0.1104E−2	0.1021E−2
22nd	0.4205E−3	0.5107E−3
23rd	0.1716E−3	0.1946E−3
24th	0.8580E−4	0.7926E−4
25th	0.4064E−4	0.3963E−4
26th	0.2919E−4	0.3180E−4
27th	0.9509E−5	0.1914E−5
28th	0.5009E−5	0.4862E−6
29th	0.4914E−5	0.2390E−6
30th	0.2234E−8	0.2389E−6
31st	0.1175E−8	0.1400E−14
32nd	0.1400E−14	

Table 5. Length of enclosing intervals obtained after each function evaluation in solving Problem 15 with $n = 40$ and $tol = 10^{-15}$, starting with the 21st function evaluation, upto the termination

AVERAGE convergence speed. We wish to mention that Algorithm 2 uses the fifth order inverse interpolation to achieve a higher efficiency index than Algorithm 1 which uses the third order inverse interpolation. Both algorithms require at most four function evaluations per iteration and asymptotically only three. Since a function evaluation usually costs much more than the computation of $\bar{x} = IP(0)$ defined in (2), Algorithm 2 in general will not have a higher computational complexity than Algorithm 1 does.

Finally, we notice that our analyses in Section 3 indicate that Algorithm 2 achieves the efficiency index 1.7282... when the function $f(x)$ is six times continuously differentiable. In [4], it is proved that in order for Algorithm 1 to achieve the efficiency index 1.6686... the function $f(x)$ only needs to be four times continuously differentiable. This makes Algorithm 2 seem more restrictive than Algorithm 1. Fortunately, both algorithms guarantee the linear convergence shown in (3) as long as $f(x)$ itself is continuous. Our experiment with problem (31) also show that Algorithm 2 may perform better than Algorithm 1 even if the assumptions in Section 3 are not satisfied.

Acknowledgement

The author would like to thank the referees for their valuable comments and suggestions.

References

[1] Alefeld, G. and Potra, F. A. *On two higher order enclosing methods of J. W. Schmidt*. Z. angew. Math. Mech **68** (8) (1988), pp. 331–337.

[2] Alefeld, G. and Potra, F. A. *Some efficient methods for enclosing simple zeros of nonlinear equations*. BIT **32** (1992), pp. 334–344.

[3] Alefeld, G., Potra, F. A., and Shi, Yi. *On enclosing simple roots of nonlinear equations*. Mathematics of Computation **61** (1993), pp. 733–744.

[4] Alefeld, G., Potra, F. A., and Shi, Yi. *Enclosing zeros of continuous functions*. TOMS **21** (1995), pp. 327–344.

[5] Brent, R. P. *Algorithms for minimization without derivatives*. Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[6] Bus, J. C. P. and Dekker, T. J. *Two efficient algorithms with guaranteed convergence for finding a zero of a function*. TOMS **1** (1975), pp. 330–345.

[7] Dekker, T. J. *Finding a zero by means of successive linear interpolation*. In: Dejon, B. and Henrici, P. "Constructive Aspects of the Fundamental Theorem of Algebra", Wiley Interscience, 1969.

[8] Le, D. *An efficient derivative—free method for solving nonlinear equations*. TOMS **11** (1985), pp. 250–262.

[9] Ostrowski, A. M. *Solution of equations in Banach spaces*. Acedemic Press, New York, 1973.

[10] Potra, F. A. *On Q-order and R-order of convergence*. J. Optim. Theory Appl. **63** (1989), pp. 415–431.

Received: December 1, 1995
Revised version: June 27, 1996

Department of Mathematics and Computer Science
Bloomsburg University of Pennsylvania
Bloomsburg
PA 17815
USA