Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Superlinear bracketing method for solving nonlinear equations

Alojz Suhadolnik

University of Ljubljana, Faculty of Mechanical Engineering, Aškerčeva 6, 1000 Ljubljana, EU, Slovenia

A R T I C L E I N F O

Keywords:
Nonlinear equations
Muller's method
Root finding

A B S T R A C T

A new method based on Muller's algorithm for solving nonlinear equations has been developed. The classic Muller's method is based on an interpolating polynomial built on the last three points of an iterative sequence. Unfortunately, Muller's method is not globally convergent. In order to ensure the global convergence a bracketing is introduced. The proposed method does not require the use of a derivative of the function and is more rapidly convergent than a classical regula falsi method. The method is good alternative to other bracketing methods.

1. Introduction

Many algorithms have been proposed for finding a single root of a nonlinear equation

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

The most basic method among so-called root bracketing methods is a dichotomy method also known as a bisection method [1]. This classic example of an enclosure method has unfortunately a rather slow convergence with an order of convergence equal to one. If $f(x)$ is a continuous function on the interval $[x_a, x_b]$ and $f(x_a)f(x_b) < 0$, then the method is guaranteed to converge. Another enclosure root finding method based on a linear interpolation is a regula falsi technique (false position) [1]. Its convergence order is also linear, but it is faster in some cases than the bisection method. In general, bracketing versions of the open methods usually sacrifice some speed and computational cost in order to ensure convergence. This feature can be clearly observed on the secant method vs. regula falsi. A superlinear variant of the regula falsi method called an Illinois method is also known [2]. In order to increase reliability, combinations of several methods were introduced. Dekker [3] combined the bisection method and secant method. This method was upgraded by Brent [4], where an inverse quadratic interpolation is involved. Further improvement of Brent's method was performed by Alefeld and Potra [5] with several new algorithms. The use of an exponential function in the root finding algorithm on the closed interval was introduced by Ridders [6].

Instead of a linear interpolation in the case of the regula falsi, a quadratic polynomial interpolation known as Muller's method [7] can be used in combination with the bracketing algorithm. This approach may present advantages with respect to some other bracketing methods. The convergence of Muller's method can also be ensured by combining the algorithm with the bisection [8–10] or regula falsi [11]. Similar to the case of Brent's method, the inverse quadratic interpolation is also combined with Muller's method [12]. Recently, more papers were published with some new versions of enclosing methods [13–18].

In this paper a bracketing version of Muller's method is introduced. The method is suitable for finding roots of the nonlinear equations on the predefined interval. A bracketing technique ensures the convergence of the presented method while the order of convergence is at least 1.618, so that the method is more rapidly convergent than the regula falsi method. This method does not require the use of the derivative of the function. The derivative is not easy to calculate or is not available in

E-mail address: alojz.suhadolnik@guest.arnes.si

0096-3003/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2012.12.064

a number of applications. This technique requires only one function evaluation per iteration so that an asymptotic efficiency index in the sense of Ostrowski [19] is at least 1.618 if one endpoint of the closed interval converges to the zero of the function or even 1.839 in case where both endpoints of the closed interval converge alternately to the zero. The method consists of only one interpolating algorithm and is not the combination of several methods as for example is Brent's method. This property contributes to the simplicity of the algorithm.

2. Root finding method

It will be assumed in this paper that the function $f(x)$ is continuous and has a zero on a closed interval $[x_a, x_b]$ where

$$f(x_a)f(x_b) < 0. \tag{2}$$

Without loss of generality we suppose that $f(x_a) < 0$ and $f(x_b) > 0$. A new iterative value on the closed interval is calculated by Muller's method where a parabola is fitted to three points of the function $f(x)$. The first $(x_a, f(x_a))$ and second $(x_b, f(x_b))$ points are the interval border points. The third point $(x_c = .5(x_a + x_b), f(x_c))$; $x_c \in [x_a, x_b]$ is calculated as a mid-point before an iterative procedure starts. After calculating x_c, three points $(x_a, f(x_a))$, $(x_c, f(x_c))$ and $(x_b, f(x_b))$ are available for the first iteration and through these points, an interpolating polynomial can be constructed

$$p(x) = A(x - x_c)^2 + B(x - x_c) + C. \tag{3}$$

The polynomial equation can be rewritten for all three points

$$\begin{cases} f(x_a) = A(x_a - x_c)^2 + B(x_a - x_c) + C, \\ f(x_b) = A(x_b - x_c)^2 + B(x_b - x_c) + C, \\ f(x_c) = C, \end{cases} \tag{4}$$

and the parameters A, B and C can be easily determined

$$\begin{cases} A = \frac{f(x_a)-f(x_c)}{(x_a-x_c)(x_a-x_b)} + \frac{f(x_c)-f(x_b)}{(x_b-x_c)(x_a-x_b)}, \\ B = \frac{[f(x_c)-f(x_a)](x_b-x_c)}{(x_a-x_c)(x_a-x_b)} - \frac{[f(x_c)-f(x_b)](x_a-x_c)}{(x_b-x_c)(x_a-x_b)}, \\ C = f(x_c). \end{cases} \tag{5}$$

Finally, both roots $x^{*p}_{1,2}$ of the interpolating polynomial defined by Eq. (3) can be calculated

$$x^{*p}_{1,2} = x_c - \frac{2C}{B \pm \sqrt{B^2 - 4AC}}. \tag{6}$$

In this equation, both roots $x^{*p}_{1,2}$ are calculated and then the root which is closer to x_c is selected. In further calculations only the root with $x_p = \min_{j=1,2} |x_c - x^{*p}_j|$ is carried and the initial interval can be narrowed. The first endpoint of a new interval becomes x_c. The second endpoint remains x_a if the zero of the function f lies between x_a and x_c. In this case, the following inequality holds $f(x_a)f(x_c) < 0$. If $f(x_a)f(x_c) \geqslant 0$ then the second endpoint remains x_b. In the next iteration x_c has a new value $x_c = x_p$. For further details see Section 5.

Now the Eq. (6) can be rewritten in an iterative form by renaming $x_i = x_c$, $i \in \mathbb{N}$ as an approximate value of the root in the current iteration and $x_{i+1} = x_p$ as a new calculated value. In order to keep $|x_{i+1} - x_i|$ minimal as stated before, a sign before the square root in Eq. (6) and the sign of B must be equal

$$x_{i+1} = x_i - \frac{2f(x_i)}{B_i + \operatorname{sgn}(B_i)\sqrt{B_i^2 - 4A_i f(x_i)}}, \tag{7}$$

where A, B and C from (5) are indexed by i in each iteration on the interval $[x_a, x_b]$ as follows

$$\begin{cases} A_i = \frac{f(x_a)-f(x_i)}{(x_a-x_i)(x_a-x_b)} + \frac{f(x_i)-f(x_b)}{(x_b-x_i)(x_a-x_b)}, \\ B_i = \frac{[f(x_i)-f(x_a)](x_b-x_i)}{(x_a-x_i)(x_a-x_b)} - \frac{[f(x_i)-f(x_b)](x_a-x_i)}{(x_b-x_i)(x_a-x_b)}, \\ C_i = f(x_i). \end{cases} \tag{8}$$

The new interval endpoints x_a, x_b and value of x_c are defined at the end of each iteration in the following way: if $f(x_a)f(x_i) < 0$ then $x_b = x_i$, else $x_a = x_i$ and finally $x_c = x_{i+1}$.

In some cases, a minimal value of $|x_{i+1} - x_i|$ corresponds to the root which is outside the interval $[x_a, x_b]$. In order to avoid this situation, the algorithm checks the presence of the selected root on the interval. If $x_{i+1} > x_b$ or $x_{i+1} < x_a$ then Eq. (7) must change the sign before the square root and another root inside the interval is selected

$$x_{i+1} = x_i - \frac{B_i + \operatorname{sgn}(B_i)\sqrt{B_i^2 - 4A_i f(x_i)}}{2A_i}. \tag{9}$$

3. Convergence theorem

If a parabola intersects the closed interval $[x_a, x_b]$ the following lemma holds.

Lemma 1. *Let $p(x)$ represent a parabola defined in* Eq. (3). *If $[x_a, x_b]$ is closed interval and $p(x_a)p(x_b) < 0$ then the interval contains only one zero of the parabola x_p which divides the interval $[x_a, x_b]$ into two nonzero subintervals $[x_a, x_p]$ and $[x_p, x_b]$.*

Proof. The parabola is a single-valued continuous function. If an inequality $p(x_a)p(x_b) < 0$ holds, then $p(x_a)$ and $p(x_b)$ have the opposite sign. In this case the parabola has two real zeros but the closed interval $[x_a, x_b]$ contains an odd number of zeros that is one zero x_p of the parabola which divides the interval $[x_a, x_b]$. □

Although the classic Muller's method has a convergence order of approximately 1.839, the method is only locally convergent. In the following theorems the convergence order of the proposed method on the closed interval is presented.

Theorem 1. *Let $f(x)$ contain only one zero x^* in the closed interval $[x_a, x_b]$ with $f(x_a)f(x_b) < 0$ and $f'(x^*) \neq 0$. Suppose $f \in C^3$, at least in a neighbourhood of x^*. If the second order polynomial $p_i(x) = A_i(x - x_c)^2 + B_i(x - x_c) + C_i$ is defined by* parameters (8), *then the sequence of polynomial roots $\{x_i\}$ defined in* Eq. (7) *converges to x^* with the convergence order of 1.839 in case where both endpoints of the closed interval alternately converge to x^* and the polynomial roots $\{x_i\}$ are in the neighbourhood of x^*.*

Proof. In the beginning of each iteration, x_i has the value of x_c. If the polynomial roots $\{x_i\}$ converge to x^* in an alternated manner around x^*, then the interval endpoints x_a and x_b have either the value of x_{i-1} or x_{i-2}. Before the end of the iteration x_c accepts the value of x_p=x_{i+1} while x_a and x_b receive the following values: if $f(x_a)f(x_i) < 0$ then $x_b = x_i$, $x_a = x_{i-1}$, and if $f(x_a)f(x_i) \geqslant 0$ then $x_a = x_i$, $x_b = x_{i-1}$. During each iteration the three points $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$ and $(x_{i-2}, f(x_{i-2}))$ are available. Through these points the interpolating polynomial can be constructed and the root of the interpolating polynomial on the closed interval $[x_a, x_b]$ is x_{i+1}. Let the error equations be defined in the following way

$$\begin{cases} e_{i+1} = x_{i+1} - x^*, \\ e_i = x_i - x^*, \\ e_{i-1} = x_{i-1} - x^*, \\ e_{i-2} = x_{i-2} - x^*. \end{cases} \tag{10}$$

The polynomial equations from (4) with parameters A_i, B_i and C_i from (8) can be expanded into a truncated Taylor series near the function zero x^* in each iterative step i where x_a=x_{i-2}, x_b=x_{i-1} and x_c=x_i

$$\begin{cases} A_i(e_{i-2} - e_i)^2 + B_i(e_{i-2} - e_i) + C_i \approx e_{i-2}f'(x^*) + \frac{1}{2}e_{i-2}^2 f''(x^*) + \frac{1}{6}e_{i-2}^3 f'''(x^*), \\ A_i(e_{i-1} - e_i)^2 + B_i(e_{i-1} - e_i) + C_i \approx e_{i-1}f'(x^*) + \frac{1}{2}e_{i-1}^2 f''(x^*) + \frac{1}{6}e_{i-1}^3 f'''(x^*), \\ C_i \approx e_i f'(x^*) + \frac{1}{2}e_i^2 f''(x^*) + \frac{1}{6}e_i^3 f'''(x^*). \end{cases} \tag{11}$$

By solving this system of equations, the parameters A_i, B_i and C_i can be obtained

$$\begin{cases} A_i \approx \frac{1}{6}(3f''(x^*) + e_{i-2}f'''(x^*) + e_{i-1}f'''(x^*) + e_i f'''(x^*)), \\ B_i \approx \frac{1}{6}(6f'(x^*) + 6e_i f''(x^*) - e_{i-1}e_{i-2}f'''(x^*) + e_{i-1}e_i f'''(x^*) + e_{i-2}e_i f'''(x^*) + 2e_i^2 f'''(x^*)), \\ C_i \approx \frac{1}{6}(6e_i f'(x^*) + 3e_i^2 f''(x^*) + e_i^3 f'''(x^*)). \end{cases} \tag{12}$$

Consequently, the root of the polynomial equation $A_i(x_{i+1} - x_i)^2 + B_i(x_{i+1} - x_i) + C_i = 0 \leftrightarrow A_i(e_{i+1} - e_i)^2 + B_i(e_{i+1} - e_i) + C_i = 0$ has according to (7) the following expression

$$e_{i+1} \approx -\frac{e_i e_{i-1} e_{i-2} f'''(x^*)}{3B_i - 6A_i e_i + \operatorname{sgn}(3B_i - 6A_i e_i)\sqrt{9(B_i - 2A_i e_i)^2 - 6A_i e_i e_{i-1} e_{i-2} f'''(x^*)}}. \tag{13}$$

If parameters (12) are inserted in (13) and the result is further simplified, one can get

$$e_{i+1} \approx -e_i e_{i-1} e_{i-2} \frac{f'''(x^*)}{6f'(x^*)}. \tag{14}$$

Suppose that the sequence converges with order r where $r > 0$ to the zero of the function x^*. In this case, the following expression holds

$$\lim_{i \to \infty} \frac{|e_{i+1}|}{|e_i|^r} = K, \tag{15}$$

where K is the rate of convergence. This equation can be rearranged into

$$\begin{cases} |e_{i+1}| \approx K|e_i|^r, \\ |e_i| \approx K|e_{i-1}|^r, \\ |e_{i-1}| \approx K|e_{i-2}|^r. \end{cases} \tag{16}$$

By combining (14) and (16), one can calculate

$$K^{r^2-1}|e_{i-2}|^{r^3} \approx |e_{i-2}|^{r^2+r+1}\left|\frac{f'''(x^*)}{6f'(x^*)}\right|, \tag{17}$$

where $r \approx 1.839$ is the largest positive root of $r^3 = r^2 + r + 1$. This polynomial equation can be rewritten as $(r^2 - 1)(r - 1) = 2$ and from (17) K can be expressed

$$K = \left|\frac{f'''(x^*)}{6f'(x^*)}\right|^{\frac{r-1}{2}}. \qquad \square \tag{18}$$

Theorem 2. *Let $f(x)$ contain only one zero x^* in the closed interval $[x_a, x_b]$ with $f(x_a)f(x_b) < 0$ and $f'(x^*) \neq 0$. Suppose $f \in C^3$, at least in a neighbourhood of x^*. If the second order polynomial $p_i(x) = A_i(x - x_c)^2 + B_i(x - x_c) + C_i$ is defined by* parameters (8)*, then the sequence of polynomial roots $\{x_i\}$ defined in* Eq. (7) *converges to x^* with the convergence order of 1.618 in case where one endpoint of the closed interval converges to x^* while the other one remains fixed and the polynomial roots $\{x_i\}$ are in the neighbourhood of x^*.*

Proof. In some cases of the proposed method only one side of the interval converges to x^* while the other side of the interval remains fixed. Assume that $x_b = x_0$ remains fixed and x_a converges to x^* because $f(x_a)f(x_i)$>=0. In this case x_i has the value of x_c in each iteration and before the end of the iteration x_c accepts the value of x_p=x_{i+1} while x_a receives the value $x_a = x_i$. During the iteration the three points $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$ and $(x_0, f(x_0))$ are available. Through these points the interpolating polynomial can be constructed and the root of the interpolating polynomial on the closed interval $[x_a, x_b]$ gives the next interpolating value x_{i+1}.
If error for the constant value is $e_{i-2} = e_0$, then Eq. (14) becomes

$$e_{i+1} \approx -e_i e_{i-1}\frac{e_0 f'''(x^*)}{6f'(x^*)}. \tag{19}$$

Here the same procedure as in the proof of Theorem 1 is used and similar to Eq. (17) the result is

$$K^r|e_{i-1}|^{r^2} \approx |e_{i-1}|^{r+1}\left|\frac{e_0 f'''(x^*)}{6f'(x^*)}\right|. \tag{20}$$

From this equation it also follows that the polynomial $r^2 = r + 1$ has a positive root $r = \frac{1+\sqrt{5}}{2} \approx 1.618$, and finally

$$K = \left|\frac{e_0 f'''(x^*)}{6f'(x^*)}\right|^{\frac{1}{r}}. \qquad \square \tag{21}$$

4. Error estimates

The following theorem holds for an error estimate of the proposed algorithm on the closed interval.

Theorem 3. *Let $f(x)$ contain only one zero x^* in the closed interval $[x_a, x_b]$ with $f(x_a)f(x_b) < 0$ and $f'(x^*) \neq 0$. Suppose $f \in C^3$, at least in a neighbourhood of x^*. If the iterative equation for the root calculation on the closed interval is defined by* Eq. (7) *then the error estimate is $x^* - x_i \approx x_{i+1} - x_i$ if the interval $[x_a, x_b]$ is small enough and the endpoints of the interval x_a and x_b lie in the neighbourhood of x^*.*

Proof. The parameter A_i defined by (8) can be inserted into (7) and the result is

$$x_{i+1} - x_i = -\frac{2f(x_i)}{B_i + \operatorname{sgn}(B_i)\sqrt{B_i^2 - \frac{4f(x_i)}{(x_a - x_b)}\left(\frac{f(x_a)-f(x_i)}{(x_a - x_i)} - \frac{f(x_i)-f(x_b)}{(x_i - x_b)}\right)}}. \tag{22}$$

If the interval $[x_a, x_b]$ is small enough then a forward difference quotient and backward difference quotient under square root are approximately equal to the derivative in x_i

$$f'(x_i) \approx \frac{f(x_a) - f(x_i)}{(x_a - x_i)} \approx \frac{f(x_i) - f(x_b)}{(x_i - x_b)}. \tag{23}$$

Taking into account this approximation, Eq. (22) can be simplified

$$x_{i+1} - x_i \approx -\frac{f(x_i)}{B_i}. \tag{24}$$

Furthermore, the parameter B_i defined by (8) can also be inserted into (24)

$$x_{i+1} - x_i \approx -\frac{f(x_i)(x_a - x_b)}{\left(\frac{[f(x_i)-f(x_a)](x_b-x_i)}{(x_a-x_i)} - \frac{[f(x_i)-f(x_b)](x_a-x_i)}{(x_b-x_i)}\right)}. \tag{25}$$

Introducing derivative (23) in the achieved equation leads to

$$x_{i+1} - x_i \approx -\frac{f(x_i)(x_a - x_b)}{f'(x_i)(x_a - x_i) - f'(x_i)(x_b - x_i)}, \tag{26}$$

and by further simplification to

$$x_{i+1} - x_i \approx -\frac{f(x_i)}{f'(x_i)}. \tag{27}$$

Now the first two terms in a Taylor series $0 = f(x^*) \approx f(x_i) + (x^* - x_i)f'(x_i)$ can be used and by inserting this equation into Eq. (27) the error estimate is achieved

$$x^* - x_i \approx x_{i+1} - x_i. \quad \square \tag{28}$$

If this estimate is inserted in Eq. (14) and by taking into account (10), the error estimate for the alternate interval endpoints is

$$e_{i+1} \approx (x_{i+1} - x_i)(x_i - x_{i-1})(x_{i-1} - x_{i-2})\frac{f'''(x^*)}{6f'(x^*)}, \tag{29}$$

and similarly by using Eqs. (19) and (10) for the case of the one constant endpoint

$$e_{i+1} \approx -(x_{i+1} - x_i)(x_i - x_{i-1})\frac{e_0 f'''(x^*)}{6f'(x^*)}. \tag{30}$$

5. Proposed algorithm

In the rest of the text, the proposed algorithm is named Alg. In the algorithm, the inputs are the left side x_a and right side x_b of the initial interval, function $f(x)$, calculated precision ε and maximum number of iterations $N_{\max}$. The interval $[x_a, x_b]$ must contain the root, so the initial condition $f(x_a)f(x_b) < 0$ is fulfilled.

Start of the algorithm Alg.

```
Step 1.  Input: $x_a, x_b, f, \varepsilon, N_{\max}$.
Step 2.  Initial mid-point: $x_c = \frac{x_a+x_b}{2}$, $n = 1$, $xo_c = x_c$.
Step 3.  Function values: $f_a = f(x_a)$; $f_b = f(x_b)$; $f_c = f(x_c)$.
Step 4.  Initial condition test: If sgn($f_a$) = = sgn($f_b$) then stop.
Step 5.  $A = \frac{f_a-f_c}{(x_a-x_c)(x_a-x_b)} + \frac{f_c-f_b}{(x_b-x_c)(x_a-x_b)}$,
         $B = \frac{[f_c - f_a](x_b - x_c)}{(x_a - x_c)(x_a - x_b)} - \frac{[f_c - f_b](x_a - x_c)}{(x_b - x_c)(x_a - x_b)}$,
         $C = f_c$.
Step 6.  Root of the polynomial: $x_p = x_c - \frac{2C}{B+\mathrm{sgn}(B)\sqrt{B^2-4AC}}$.
Step 7.  If $x_p > x_b$ or $x_p < x_a$ then $x_p = x_c - \frac{B+\mathrm{sgn}(B)\sqrt{B^2-4AC}}{2A}$.
Step 8.  $f_p = f(x_p)$.
Step 9.  New endpoints of the interval:
           If $f_a f_c < 0$ then $x_b = x_c, f_b = f_c$ else $x_a = x_c, f_a = f_c$.
Step 10. $x_c = x_p, f_c = f_p$.
Step 11. If $n > N_{\max}$ then stop.
Step 12. If $n > 1$ and $|xo_c - x_c| < \varepsilon$ then print $x_c, f_c$ and stop.
Step 13. $n = n + 1$, $xo_c = x_c$.
Step 14. Go to step 5.
```

End of the algorithm Alg.

The algorithm (step 3) contains three function evaluations before the main algorithm loop begins. Step 8, which is part of the loop, also contains the function evaluation. If the number of iterations is n, then the total number of function evaluations is $n + 3$ (see the last row in Table 1).

The speed of convergence of the proposed method is estimated by Eqs. (14) and (19) if $f'(x^*) \neq 0$ and $f(x)$ is three times continuously differentiable. If $f'(x^*) = 0$ this method may suffer due to the slow convergence (see Table 1, examples 17 and 18).

6. Numerical examples

In this section, some numerical examples that are commonly used in literature are presented. In Table 1 some functions $f(x)$ are listed and the roots of the proposed nonlinear equations $f(x)$=0 are calculated with different methods on the different intervals denoted by $[x_a, x_b]$. The number of iterations n is presented for the bisection, regula falsi, Illinois', Brent's and the presented algorithm named Alg. The Brent's method used in this work is actually the standard routine “fzero” for finding roots of nonlinear equations in the Matlab.

The bottom row of the Table 1 includes terms for calculating the number of the function evaluations in each particular column based on the number of iterations n. The stopping condition is $|x_{i+1} - x_i| < \varepsilon$.

In most cases, presented algorithm performs better than the other conventional bracketing methods shown in Table 1. If approaching to the root alternates between the interval endpoints (see examples 3, 5, 6, 8, 13, and 16) then the convergence is fast. But in some cases presented algorithm slowly converges to the root of the nonlinear equation due to the fixed endpoint of the interval. This situation is clearly evident in examples 17 and 18 where a power function with odd exponents is presented. These two cases are examples of multiple roots where the derivative of the function is actually zero and the presented method according to Eqs. (14) and (19) is not appropriate. The stopping condition $|x_{i+1} - x_i| < \varepsilon$ is not suitable for the functions which have a small derivative in the neighbourhood of the root. If the stopping condition is replaced by $|f(x_{i+1})| < \varepsilon$ then the algorithm stops in less than 10^5 iterations in both cases. In particular, example 17 stops after 522 iterations and example 18 after 5695 iterations. In this case, step 12 in the algorithm is replaced with the following code: “If $n > 1$ and $|f_c| \leqslant \varepsilon$ or $f_c == 0$ then print x_c, f_c and stop” and the results are shown in Table 1, column named Alg*.

It can be observed from Table 1 that the algorithm is particularly fast and outperforms the other methods in cases where lower power terms prevail (see examples 6, 9, 14, and 15). The fastest convergence presents example 6, where prevails the square term of the function.

The slow convergence is observed also in cases where the derivative in one endpoint of the beginning interval is nearly zero, remains low in the zero of the function while in the other endpoint is large in absolute value (see examples 2, 4, and 10). This type of function causes huge arc of the calculated parabola as depicted in Fig. 1. In these cases all algorithms suffer due to the relative increase in the number of iterations except the bisection. But even in this situation where approaching to the zero of the function by the different iterative methods is slow the proposed method overcomes other methods except in example 2, where Brent's method is faster.

Table 1
Comparison of the different bracketing methods by presenting the number of iterations n.

$\varepsilon = 10^{-15}$, $N_{\max} = 10^5$			Bisection	Regula falsi	Illinois	Brent	Alg	Alg*
No	$f(x)$	$[x_a, x_b]$	n					
1	$\ln x$	[.5,5]	52	29	10	8	8	7
2	$(10-x)e^{-10x} - x^{10} + 1$	[.5, 8]	53	$>10^5$	37	15	25	24
3	$e^{\sin x} - x - 1$	[1,4]	52	33	10	12	7	6
4	$11x^{11} - 1$	[.5, 1]	49	108	13	10	9	8
5	$2\sin x - 1$	$[.1, \frac{\pi}{3}]$	50	15	8	6	5	4
6	$x^2 + \sin\frac{x}{10} - .25$	[0, 1]	50	34	11	8	4	3
7	$(x-1)e^{-x}$	[0, 1.5]	51	74	12	9	7	6
8	$\cos x - x$	[0, 1.7]	51	18	9	6	5	4
9	$(x-1)^3 - 1$	[1.5, 3]	51	61	12	9	7	6
10	$e^{x^2+7x-30} - 1$	[2.6, 3.5]	50	4020	20	10	9	8
11	$\arctan x - 1$	[1, 8]	53	27	11	8	8	7
12	$e^x - 2x - 1$	[.2, 3]	52	157	15	12	8	7
13	$e^{-x} - x - \sin x$	[0, .5]	49	13	8	5	5	4
14	$x^3 - 1$	[.1, 1.5]	51	36	11	7	6	5
15	$x^2 - \sin^2 x - 1$	[−1, 2]	52	34	12	9	7	6
16	$\sin x - \frac{x}{2}$	$[\frac{\pi}{2}, \pi]$	51	33	9	7	6	5
17	x^3	$[-0.5, \frac{1}{3}]$	48	$>10^5$	95	144	$>10^5$	522
18	x^5	$[-0.5, \frac{1}{3}]$	48	$>10^5$	185	122	$>10^5$	5695
Number of function evaluations			$n + 1$	$n + 2$	$n + 2$	$n + 2$	$n + 3$	$n + 3$

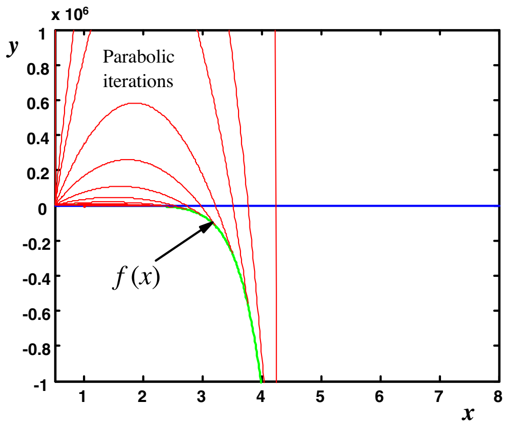

Fig. 1. Function $f(x) = (10 - x)e^{-10x} - x^{10} + 1$ on the initial interval [.5,8] (example 2 in Table 1).

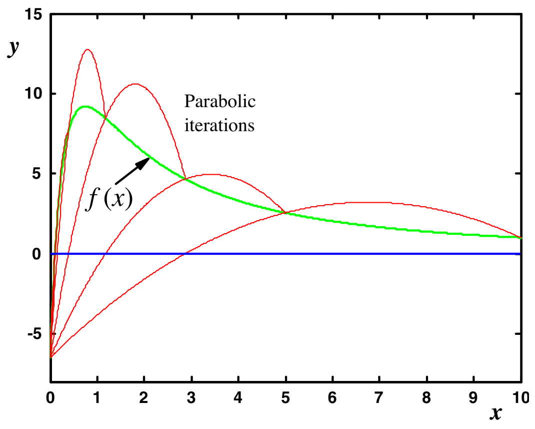

Fig. 2. Function $f(x) = \frac{50}{(x+.9)^2}\ln(x + .9)$ on the initial interval [0,10].

Table 2
Number of iterations n of the function $f(x) = \frac{50}{(x+a)^2}\ln(x + a)$ on the initial interval [0, 10].

a ($\varepsilon = 10^{-15}$)	n
.9	10
.99	7
.999	6
.9999	5
.99999	4
.999999	4
.9999999	4
.99999999	4
.999999999	3
.9999999999	3
.99999999999	3
.999999999999	3
.9999999999999	3
.99999999999999	3

The presented method also calculates the simple real roots of the polynomial. In first step, the particular polynomial root must be bracketed with the initial interval and then the algorithm can be executed. Special care must be taken on the close real roots of the polynomial where the initial interval is not easy to determine. The method is not suitable to find multiple roots of the polynomials with an even multiplicity, since the polynomial does not change sign at such roots. Only multiple

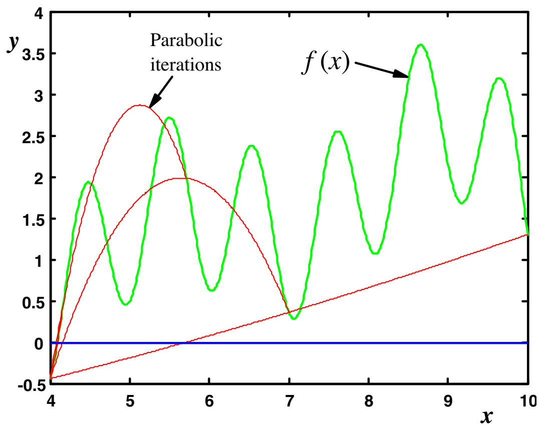

Fig. 3. Function $f(x) = \sin(x) + \sin(1.5x) + \sin(6x) + 1.5$ on the initial interval [4, 10].

roots with an odd multiplicity can be bracketed by monitoring the sign of the polynomial but the convergence is slow as mentioned before.

The next example shows the stability of the proposed method with the zero of the function close to the origin of the coordinate system. In the calculation, function $f(x) = \frac{50}{(x+a)^2}\ln(x+a)$ is used where the parameter a approaches to the one while the value of the function approaches to the zero (Fig. 2). In Table 2 the value of a and the number of iterations are shown where $\varepsilon = 10^{-15}$. The example shows that the stability of the method is not affected while the zero of the function approaches to the origin of the coordinate system.

The last example in Fig. 3 shows the function $f(x) = \sin(x) + \sin(1.5x) + \sin(6x) + 1.5$ with large oscillations. The proposed algorithm calculates the zero of the function in 8 iterative steps with $\varepsilon = 10^{-15}$.

7. Conclusions

The paper presents a new bracketing algorithm for solving nonlinear equations. The algorithm is based on Muller's method with a superlinear order of convergence. The proposed algorithm is convergent, robust and stable. The advantage of the proposed method is also the simplicity of the algorithm which is not the combination of several different methods. The algorithm can be used as a good alternative to the well-known bracketing methods. In the paper, some examples are introduced and a comparison with other methods is given.

References

[1] H.M. Antia, Numerical Methods for Scientists and Engineers, Tata McGraw-Hill, New Delhi, 1991.
[2] M. Dowell, P. Jarratt, A modified regula falsi method for computing the root of an equation, BIT 11 (1971) 168–174.
[3] T.J. Dekker, Finding a zero by means of successive linear interpolation, in: B. Dejon, P. Henrici (Eds.), Constructive Aspects of the Fundamental Theorem of Algebra, Interscience, New York, 1969.
[4] R.P. Brent, An algorithm with guaranteed convergence for finding a zero of a function, Comput. J. 14 (1971) 422–425.
[5] G.E. Alefeld, F.A. Potra, Some efficient methods for enclosing simple zeros of nonlinear equations, BIT 32 (1992) 334–344.
[6] C.J.F. Ridders, A new algorithm for computing a single root of a real continuous function, IEEE Trans. Circuits Sys. 26 (1979) 979–980, http://dx.doi.org/10.1109/TCS.1979.1084580.
[7] D.E. Muller, A method for solving algebraic equations using an automatic computer, Math. Tables Other Aids Comput. 10 (1956) 208–215.
[8] B.K. Park, S. Hitotumatu, A study on new Muller's method, Publ. Res. Inst. Math. Sci. 23 (1987) 667–672.
[9] B.K. Park, A study on the evaluation of numerical values in an algebraic equation, Commun. Kor. Math. Soc. 4 (1989) 59–71.
[10] X. Wu, Improved Muller method and bisection method with global and asymptotic superlinear convergence of both point and interval for solving nonlinear equations, Appl. Math. Comput. 166 (2005) 299–311.
[11] A. Suhadolnik, Combined bracketing methods for solving nonlinear equations, Appl. Math. Lett., 25 (2012) 1755-1760, http://dx.doi.org/10.1016/j.aml.2012.02.006.
[12] F. Costabile, M.I. Gualtieri, R. Luceri, A modification of Muller's method, Calcolo 43 (2006) 39–50.
[13] J. Chen, New modified regula falsi method for nonlinear equations, Appl. Math. Comput. 184 (2007) 965–971.
[14] P.K. Parida, D.K. Gupta, A cubic convergent iterative method for enclosing simple roots of nonlinear equations, Appl. Math. Comput. 187 (2007) 1544–1551.
[15] P.K. Parida, D.K. Gupta, An improved regula falsi method for enclosing simple zeros of nonlinear equations, Appl. Math. Comput. 177 (2006) 769–776.
[16] J. Chen, Z.H. Shen, On third-order convergent regula falsi method, Appl. Math. Comput. 188 (2007) 1592–1596.
[17] M.A. Noor, F. Ahmad, Numerical comparison of iterative methods for solving nonlinear equations, Appl. Math. Comput. 180 (2006) 167–172.
[18] X. Wu, Z. Shen, J. Xia, An improved regula falsi method with quadratic convergence of both diameter and point for enclosing simple zeros of nonlinear equations, Appl. Math. Comput. 144 (2003) 381–388.
[19] A.M. Ostrowski, Solution of Equations and Systems of Equations, Academic Press, NewYork, 1960.