# Novel Algorithms Solving Nonlinear Equation in Very Large Scope

Zhezhao ZENG,
College of Electrical & Information Engineering, Changsha University of Science & Technology, Changsha, China
Email: hncs6699@yahoo.com.cn

Dongmei LIN and Lulu ZHENG
College of Electrical & Information Engineering, Changsha University of Science & Technology, Changsha, China
Email: nose127@163.com, zll410730@126.com

***Abstract*—Two new algorithms used in large-scope solution are put forward to solve nonlinear equations which were not solved by some traditional methods. The initial value was arbitrarily chosen in very large scope. The convergence theorem of the algorithm was presented and proved. The computation is carried out by simple steepest descent rule without evaluation of the derivative evaluation of *f*. Thus, computation time is saved. The specific examples showed that the proposed method can choose the initial value in very large scope, without the derivative evaluation, and less computation with high precision and rapid convergence.**

***Index Terms*—nonlinear equations, algorithms, convergence, steepest descent rule, high precision, examples**

## I. INTRODUCTION

Problems associated with nonlinear equation arise in many engineering fields. Traditional methods cover mainly the Bisection method, the Fixed-point iteration, the Newton's method, and the Secant method[1-3] etc. The Newton's method is one of the most powerful and well-known numerical methods solving a root-finding problem. But it has a major weakness: the need to know the value of the derivative of $f$ at each approximation. Frequently, $f'(x)$ is far more difficult and needs more arithmetic operations to calculate than $f(x)$. In addition, Newton's method converges rapidly if the initial approximation is sufficiently close to the solution. Otherwise, it is difficult to converge. The Bisection method is an easy and practical method, but it may converge slowly. Usually, it is used to estimate the initial approximation of root in application. The Secant method is similar to the Newton's method. The derivative evaluation in the Newton's method is replaced by the difference evaluation in the Secant method. Its advantage is obvious when it is not easy or impossible to evaluate the derivative, but, generally speaking, it slowly converges. Fixed-point iteration can be done when it forms a proper iterative function $g(x)$ by a given nonlinear equation $f(x)=0$ : $x_{k+1}=g(x_k)$ , $k=0,1,2,\cdots$. It is well-known that the formation of iterative function $g(x)$ determines convergent speed. An appropriate iterative function $g(x)$ can converge rapidly while an unsuitable one may do very slowly or do not at all. Thus, the formation of iterative function $g(x)$ is very important, but it is not easy how to form an appropriate or the best iterative function $g(x)$ in practical application. There are also some other improved algorithms such as the semi-covered iteration method and an accelerated iteration method to nonlinear equation in large scope[4-6] etc. The semi-covered iteration demands that $f'(x)$ should be not zero in the interval $[a,b]$ and $f(x)$ should include arbitrary order continuous the derivative, so harsh terms are demanded. Some other methods, such as the Routh method [7], the Truong, Jeng and Reed method [8], the Fedorenko method [9], the Halley method[10], and some modified Newton's methods[11-13], etc. Among other methods, some have low accuracy, and some have more computation, especially to say is the modified Newton's methods must have a good initial value near solution. Furthermore, it is very difficult for the all methods mentioned above to find multiple real or complex roots of nonlinear equations or polynomials.

Based on the limitation of the algorithms above, first, we studied a novel method independent of choice of initial approximation and iterative function: a novel algorithm solving nonlinear equation in very large scope. The approach can choose the initial approximation in very large scope, without computing the derivative, and less computation with very high precision and very rapid convergence speed.

Second, for the problem difficult to find multiple real or complex roots of nonlinear equation or polynomials, we propose an algorithm finding multiple real or complex roots of nonlinear equation or polynomials with variable learning rate. The approach can find multiple roots of nonlinear equation or polynomials with less computation, high accuracy and rapid convergence.

## II. THE ADAPTIVE ALGORITHM SOLVING NONLINEAR EQUATION IN LARGE SCOPE


doi:10.4304/jcp.5.4.606-613

*A. The algorithm description*

To find a solution to $f(x)=0$ given the continuous function $f$ on the interval $[a,b]$, where $f(a)$ and $f(b)$ have opposite signs. The main idea of the algorithm is to make function $f$ satisfy $f(x)=0$ through training the weight $x$. The adaptive algorithm is as follows:

Given an initial approximation weight $x_k$, then an error function can be obtained:

$$e(k)=0-f(x_k)=-f(x_k) \tag{1}$$

Define an objective function $J$ as follows:

$$J=\frac{1}{2}e^2(k) \tag{2}$$

To minimize the objective function $J$, the weight $x_k$ is recursively calculated via using a simple gradient descent rule:

$$x_{k+1}=x_k-\eta\frac{dJ}{dx_k} \tag{3}$$

Where $\eta$ is learning rate and $0<\eta<1$.

Differentiating Eq. (35) with respect to $x_k$, it can be obtained that

$$\frac{dJ}{dx_k}=\frac{dJ}{de(k)}\frac{de(k)}{df(x_k)}\frac{df(x_k)}{dx_k}=-e(k)f'(x_k) \tag{4}$$

Substituting Eq. (4) into Eq. (3), we have

$$x_{k+1}=x_k+\eta e(k)f'(x_k) \tag{5}$$

We know from Eq.(5) the need to calculate the evaluation of the derivative of $f$ at each approximation. Frequently, $f'(x)$ is far more difficult and needs more arithmetic operations to compute than $f(x)$. To circumvent the problem of the derivative evaluation of $f$, we introduce a slight variation. By definition,

$$M=\overline{f'(x)}=\frac{f(b)-f(a)}{b-a} \tag{6}$$

Using this approximation for $f'(x_k)$ gives

$$x_{k+1}=x_k+\eta e(k)M \tag{7}$$

It is very obvious that the computation is less using constant $M$ to replace the derivative $f'(x_k)$.

*B. Study of the algorithm convergence*

In order to ensure the absolute convergence of the algorithm, it is important to select a proper learning rate $\eta$. In the section, we present and proof the theorem about convergence of the algorithm. It is as follows:

***Theorem* 1:** Suppose that $f\in C[a,b]$, and $f(a)\cdot f(b)<0$. Only when $0<\eta<2/L^2$, the algorithm is convergent, where $\eta$ is learning rate and $L=\max_{x\in[a,b]}(|f'(x)|$.

***Proof*:** Define a Lyapunov function:

$$V(k)=\frac{1}{2}e^2(k) \tag{8}$$

Then

$$\Delta V(k)=\frac{1}{2}e^2(k+1)-\frac{1}{2}e^2(k) \tag{9}$$

Since

$$e(k+1)=e(k)+\Delta e(k)=e(k)+\frac{de(k)}{dx_k}\Delta x_k \tag{10}$$

and

$$\Delta x_k=-\eta\frac{dJ}{dx_k}=-\eta e(k)\frac{de(k)}{dx_k} \tag{11}$$

According to (9) ,(10) and (11), we have

$$\begin{aligned}\Delta V(k)&=\Delta e(k)[e(k)+\frac{1}{2}\Delta e(k)]\\&=\left|\frac{de(k)}{dx_k}\right|^2e^2(k)(-\eta+\frac{1}{2}\eta^2\left|\frac{de(k)}{dx_k}\right|^2)\end{aligned} \tag{12}$$

Known from the Eq.(1): $\frac{de(k)}{dx_k}=-f'(x_k)$, Substituting it into Eq. (12) gives:

$$\Delta V(k)=|f'(x_k)|^2e^2(k)(-\eta+\frac{1}{2}\eta^2|f'(x_k)|^2) \tag{13}$$

Since $|f'(x_k)|^2e^2(k)\ge 0$, if the algorithm is convergent, i.e. $\Delta V_k<0$, then it is easy to see from Eq. (13) that:

$$-\eta+\frac{1}{2}\eta^2|f'(x_k)|^2<0 \tag{14}$$

Also since $\eta>0$, we have

$$0<\eta<2/|f'(x_k)|^2\le 2/\max_{x\in[a,b]}(|f'(x)|^2 \tag{15}$$

Define $L=\max_{x\in[a,b]}(|f'(x)|$, then we obtain:

$$0<\eta<2/L^2 \tag{16}$$

*C. Evaluation of the optimal learning rate* $\eta_{opt}$

It is important to choose the magnitude of the learning rate $\eta$ during the training of algorithm. ***Theorem 1*** indicates the theory criterion selecting the magnitude of the learning rate $\eta$. If learning rate $\eta$ is too large, the algorithm may produce oscillation phenomenon and is not convergent at all. If learning rate $\eta$ is too small, the algorithm may be slowly convergent with more computation. Since learning rate $\eta$ depends on the

maximum derivative $L = \max_{x\in[a,b]}(|f'(x)|$, hence it is very important to choose the optimal derivative evaluation $L_{opt}$ . How is it chosen? Specially, while function $f(x)$ is too complex to calculate its derivative, it is difficult to obtain the optimal derivative evaluation $L_{opt}$ . In order to solve the problem above, we can obtain the optimum evaluation $L_{opt}$ according to the mean value and the maximum value of $f'(x)$ and the global change tendency of function $f(x)$ through the graph of function $f(x)$ , hence the optimal learning rate $\eta_{opt}$ is equal to $2/L_{opt}^2$ , i.e. $\eta_{opt} = 2/L_{opt}^2$ .

*D. Algorithm steps*

To find a solution to $f(x)=0$ given an initial approximation $x_k$ :

**INPUT:** initial approximation $x_k$ ; tolerance $Tol$ ; maximum number of iterations $N$ ; evaluating maximum value of $f'(x): L = \max_{x\in[a,b]}(|f'(x)|)$ and calculating $M = \overline{f'(x_k)} = \dfrac{f(b)-f(a)}{b-a}$ ; evaluating optimal derivative value $L_{opt}$ according to the $L$ , $M$ and the global change tendency of the function $f(x)$ ; computing optimal learning rate $\eta_{opt} = 2/L_{opt}^2$ .

**OUTPUT:** approximate solution $x_{k+1}$ or message of failure.

***Step 1*:** While $n \le N$ do Steps 2-5

***Step 2:*** Compute error function $e(k)$ and objective function $J$ : $e(k) = -f(x_k)$ ; $J = \dfrac{1}{2}e^2(k)$

***Step 3:*** Update weight: $x_{k+1} = x_k + \eta_{opt} e(k) M$

***Step 4*:** If $J \le Tol$ then

**OUTPUT** ( $x_{k+1}$ ); (*The procedure was successful.*)

**STOP.**

***Step 5:*** Set $n = n+1$.

Go back to step 2.

***Step 6:*** **OUTPUT**('The method failed after $N$ iterations, $n =$', $N$ );

(*The procedure was unsuccessful.)*

**STOP**.

## III. NUMERICAL EXAMPLES

***Example 1***: To find a solution to $f(x) = e^x - 1 - \cos \pi x = 0$ on interval[0,1]. The graph of the function $f(x)$ is shown in Figure 1. It is obvious that the function $f(x)$ has only one zero on the interval [0, 1], and is smooth. We know from the figure 1 that $f'(x)$ exists a maximum value in the [1.75, 2] $L = \dfrac{5.4-4}{2-1.75} \approx 5.6$ , and $M = \dfrac{f(1)-f(0)}{1-0} \approx 3.7183$ . Since $L$ is close to $M$ , hence let $L_{opt} = 5.6$ , and $\eta_{opt} = \dfrac{2}{5.6^2} = 0.0638$ , the results are shown as table 1.

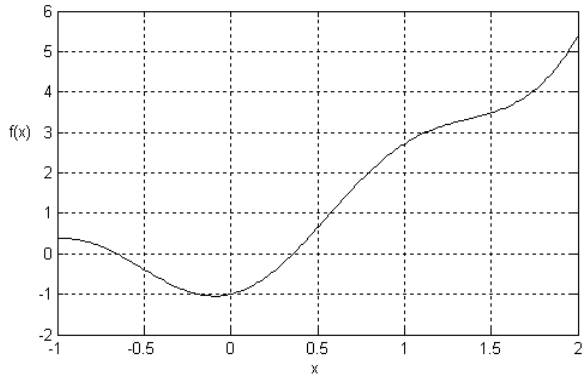



Fig.1 The graph of the function $f(x)$ in example 1

Reference [4] doesn't give iteration number, so it is impossible to know the convergent speed, and computing result depends on given initial value. Because different initial value has different result, the algorithm's stability is not good. We know form reference [5] that *Newton's method* does not converge in example 1.

TABLE 1
THE RESULTS OF THE EXAMPLE 1

| $x_0$ | k | $x_k$ | $f(x_k)$ |
|---|---|---|---|
| Algorithm proposed in the paper | | | |
| Give a random initial value on the interval[0,1] | 3 | 0.35823107 | -4.860543e-6 |
| | 5 | 0.35823223 | 0.809785e-7 |
| | 9 | 0.35823221 | -1.11022e-16 |
| -0.5 | 8 | 0.35823224 | 1.495851e-7 |
| | 13 | 0.35823221 | -1.11022e-16 |
| 0.0 | 4 | 0.35823196 | -1.048646e-6 |
| | 10 | 0.35823221 | -4.99600e-16 |
| 2.0 | 5 | 0.35823338 | 5.015526e-6 |
| | 10 | 0.35823221 | 5.55112e-16 |
| Semi-covered iteration [4] | Non-given | 0.35167154-0.35842877 | -0.2786e-5-8.3863e-4 |
| Newton's method[5] | Diver-gence | - | - |

***Example 2***: $f(x) = |\sin(100x)/x| + x^2 - 15 = 0$ on interval [3, 4](*Reference*[3]). The graph of the

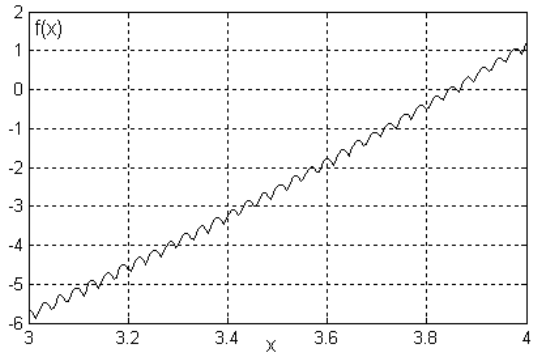
function $f(x)$ is shown in Figure 2. We know from the Figure 2 that the function $f(x)$ has three roots on the interval [3, 4] and is not smooth. Furthermore, it is difficult to compute its derivative.

Let $M = \dfrac{f(4)-f(3)}{4-3} = 6.8795$. We know from the Figure 2(b) that the local maximum of $f'(x)$ is:

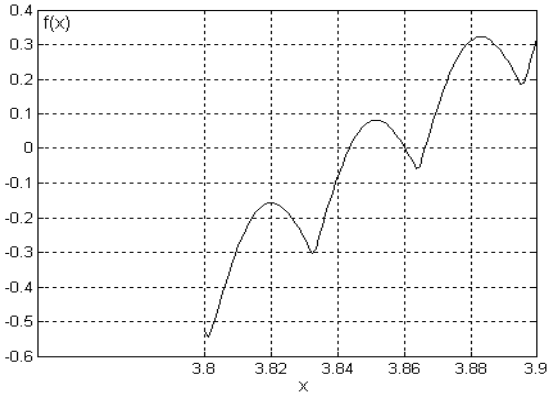
$L = \dfrac{-0.15-(-0.54)}{3.82-3.8} = 19.5$. According to the global change tendency of Figure 2, we may choose optimal derivative value:

$19.5 \times 0.8 = 16.575 \le L_{opt} \le 19.5$. Let $\eta = 2/L_{opt}^2$. The results are shown as table 2.

Why is not the second root obtained? Actually, we only replace $M = 6.8795$ with $-M = -6.8795$, the second root can be obtained. The results are shown as table 3.




(a) The graph of the $f(x)$ on the interval [3, 4]




(b) The graph of the $f(x)$ on the interval [3.84,3.87]

Figure 2 The graph of the function $f(x)$ in example 2

***Example 3***: To find a solution to

$$f(x) = \left|\cos(x^2)\right| + x^2 - 10 = 0$$

On interval [0, 5] (*Reference*[3]). The graph of the function $f(x)$ is shown in Figure 3. We know from the Figure 3 that the function $f(x)$ has only one zero on the interval [0, 5] and is not smooth. Furthermore, it is difficult to calculate its derivative.

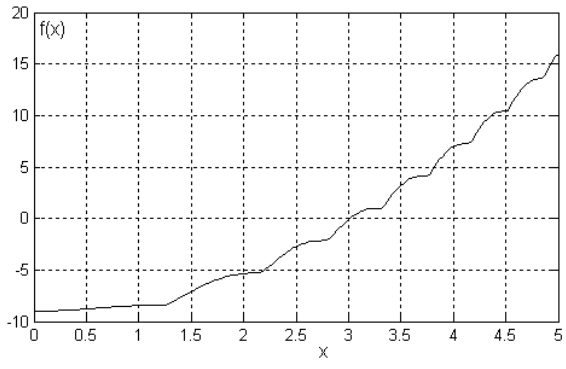
Let $M = \dfrac{f(5)-f(0)}{5-0} = 4.9982$. We know from the Figure 3 that the local maximum of $f'(x)$ is as follows: $L = \dfrac{16-13.6}{5-4.85} = 16$. According to the global change tendency of Figure 3, we may choose an optimal derivative value: $L_{opt} = 9$, hence $\eta = \dfrac{2}{9^2} = 0.02469$. The results are shown as table 4.

TABLE 2
THE RESULTS OF THE EXAMPLE 2

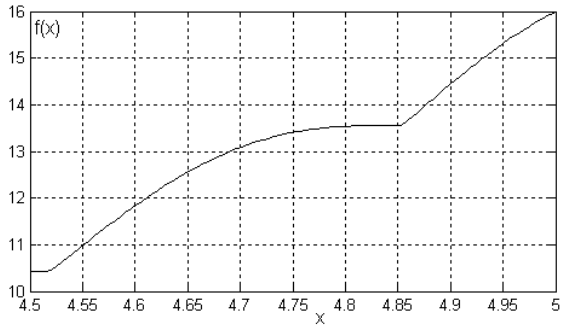
| $x_0$ | k | $x_k$ | $f(x_k)$ |
|---|---|---|---|
| $x_0 \in [3,4]$ | 9 | 3.8434287569 (the first root) | 0.175540e-6 |
| $L_{opt} = 16.57$ | 14 | 3.8434287656 (the first root) | 1.776356e-15 |
| Let $x_k = 2$ | 19 | 3.8434271182 (the first root) | 3.32017e-5 |
| $L_{opt} = 16.57$ | 37 | 3.8434287656 (the first root) | 1.77635e-15 |
| Let $x_k = 6$ | 18 | 3.8662013777 (the third root) | 2.6017e-5 |
| $L_{opt} = 19.5$ | 33 | 3.8662021650 (the third root) | 1.77635e-15 |

TABLE 3
THE RESULTS OF THE SECOND ROOT IN EXAMPLE 2

| $x_0 \in$ | k | $x_k$ | $f(x_k)$ |
|---|---|---|---|
| [3.85,3.86] | 6 | 3.8602629510 (the second root) | 2.221169e-5 |
| $L_{opt}$ = 16.575 | 22 | 3.8602643163 (the second root) | -3.552713e-15 |




(a) The graph of the $f(x)$ on the interval [0, 5]




(b) The graph of the $f(x)$ on the interval [4.5, 5]

Figure 3 The graph of the function $f(x)$ in example 3

TABLE 4
THE RESULTS OF THE SECOND ROOT IN EXAMPLE 3

| $x_0$ | k | $x_k$ | $f(x_k)$ |
|---|---|---|---|
| $x_0 \in$ [0,5] | 5 | 3.0106939005 | -1.737968e-6 |
| | 10 | 3.0106941138 | 1.776357e-15 |
| -2 | 14 | 3.0106941138 | 1.776356e-15 |
| 8 | 14 | 3.0106941138 | -1.776357e-15 |

***Example 4***: To find a solution to

$$f(x) = \left|e^x - 2\right| - 1 = 0$$

on interval [0.5,1.5] (*Reference*[3]). The graph of the function $f(x)$ is shown in Figure 4. We know from the Figure 4 that the function $f(x)$ has only one zero on the interval [0.5, 1.5] and is not smooth. Furthermore, it is difficult to compute its derivative.

Let $M = \dfrac{f(1.5) - f(0.5)}{1.5 - 0.5} \approx 2.0314$. We know from the Figure 4 that the local maximum of $f'(x)$ is as follows: $L = \dfrac{1.5 - (-1)}{1.5 - 0.7} = 3.125$. Since $L$ is close to $M$, choosing $L_{opt} = 3.125$ and $\eta_{opt} = 2/L_{opt}^2 = 0.2048$. The results are shown as table 5.

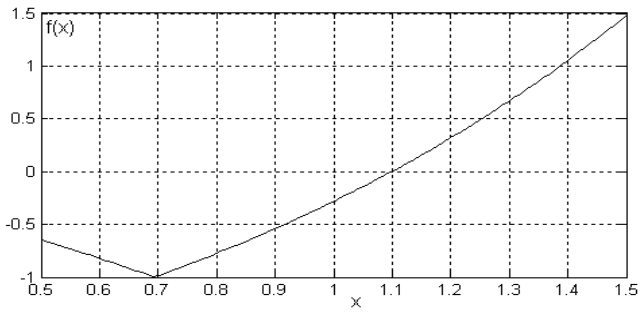



Figure 4 The graph of the function $f(x)$ in example 4

According to the reference [3] and reference [6], when the initial approximation $x_k$ is in within the small neighborhood of roots ($\left|f(x)\right| \le 0.001$), the comparison results from example 2 to example 4 are shown in table 6.

TABLE 5
THE RESULTS OF THE SECOND ROOT IN EXAMPLE 4

| $x_0$ | k | $x_k$ | $f(x_k)$ |
|---|---|---|---|
| $x_0 \in$ [0.5,1.5] | 10 | 1.09861122732361 | -3.18403e-6 |
| | 29 | 1.09861228866811 (*exact value*) | 0 |
| 0.1 | 35 | 1.09861228866811 (*exact value*) | 0 |
| 2.0 | 37 | 1.09861228866811 (*exact value*) | 0 |

TABLE 6
THE COMPARISON RESULTS FROM EXAMPLE 2 TO EXAMPLE 4 (REFERENCE [3] AND REFERENCE [6])

| Algorithms | k | $x_{k+1}$ | $f(x_{k+1})$ |
|---|---|---|---|
| Example 2 | | | |
| Newton method | 4 | 3.8434 | 1.007e-5 |
| Newton-like method | 6 | 3.8434 | 0.520e-4 |
| Secant method | 6 | 3.8434 | 0.645e-6 |
| Ref. [3,6] | 3 | 3.8434 | 0.437e-6 |
| The method in this paper | 3 | 3.8434 | 0.400e-8 |
| Example 3 | | | |
| Newton method | 6 | 3.1874 | 0.902e-6 |
| Newton-like method | Inf | Divergence | Divergence |
| Secant method | 5 | 3.0107 | 0.433e-5 |
| Ref. [3,6] | 3 | 3.0107 | 0.845e-6 |
| The method in this paper | 2 | 3.0107 | 1.42e-7 |
| Example 4 | | | |
| Newton method | 3 | 1.0986 | 1.960e-5 |
| Newton-like method | 5 | 1.0986 | 3.082e-5 |
| Secant method | 4 | 1.0986 | 1.688e-5 |
| Ref. [3,6] | 5 | 1.0986 | 0.656e-6 |
| The method in this paper | 6 | 1.0986 | 0.677e-7 |

## IV. THE ALGORITHM FINDING MULTIPLE ZEROS OF NONLINEAR EQUATION

*A. The algorithm description*

The approaches researched above are only suitable for solving the nonlinear equations with a single root. When nonlinear equations have multiple roots, we must find new method solving nonlinear equations with multiple roots.

Assumed that $x^*$ is the multiple $m$ root of nonlinear equation $f(x)$, i.e. $f(x^*)=0$, then $f^{(m-1)}(x^*)=0$.

Hence the algorithm is as follows:

Given an arbitrary initial value $x_k$, real or complex, an error function can be obtained:

$$e(k)=0-f^{(m_i-1)}(x_k)=-f^{(m_i-1)}(x_k) \tag{17}$$

Define an objective function $J$ as

$$J(k)=\frac{1}{2}e^2(k) \tag{18}$$

To minimize the $J$, the weight $x_k$ is recursively computed via using a simple gradient descent rule with variable learning rate:

$$x_{k+1}=x_k-\eta(k)\frac{dJ(k)}{dx_k} \tag{19}$$

Where $\eta(k)$ is variable learning rate and is usual $0<\eta(k)<1$, then

$$\Delta x_k=-\eta(k)\frac{dJ(k)}{dx_k} \tag{20}$$

On differentiating eqn.18 with respect to $x_k$, the gradient of $J(k)$ with respect to $x_k$ is given by

$$\frac{dJ(k)}{dx_k}=\frac{dJ(k)}{de(k)}\frac{de(k)}{df^{(m-1)}(x_k)}\frac{df^{(m-1)}(x_k)}{dx_k}=-e(k)f^{(m)}(x_k) \tag{21}$$

Substituting (21) into (19), we have

$$x_{k+1}=x_k+\eta(k)e(k)f^{(m_i)}(x_k) \tag{22}$$

here, $\Delta x_k=e(k)\eta(k)f^{(m)}(x_k)$.

To illustrate the point here is that the algorithm finding multiple root of nonlinear equation is also suitable for finding multiple roots of polynomials.

*B. Research of the algorithm convergence*

In order to ensure the absolute convergence of the algorithm above, it is important to select a proper adaptive learning rate: $\eta(k)$. In the section, we present and prove the theorem about convergence of the algorithm proposed as follows:

***Theorem 2*:** the function $f\in C^m[a,b]$ has a zero of multiplicity $m$ at $x^*$ in (a, b) if and only if $f(x^*)=f'(x^*)=\cdots=f^{(m-1)}(x^*)=0$, but $f^{(m)}(x^*)\neq 0$, then only when

$$0<\eta(k)<2/[f^{(m)}(x_k)]^2 \tag{23}$$

the algorithm proposed is convergent, where $\eta(k)$ is variable learning rate.

***Proof:*** the proof of the theorem 2 is similar to the theorem 1. Please reference to the proof method of theorem 1.

*C. Algorithm steps*

To find a zero of multiplicity $m_i$ to $f(x)=0$ given one approximation $x_0$:

**INPUT:** $x_0$ (real number or complex number); tolerance $Tol$; maximum number of iterations $N$; let $k=0$;

**OUTPUT:** approximate solution $x_{k+1}$ or message of failure.

**Step 1:** While $k\le N$ do Steps 2-5

**Step 2:** set $e(k)=-f^{(m_i-1)}(x_k)$; $J=\frac{1}{2}|e(k)|^2$

If $|f^{(m_i)}(x_k)|\le 2$ then $\eta_{opt}(k)=0.5$

Else $\eta_{opt}(k)=1/[f^{(m_i)}(x_k)]^2$

**Step 3**: $x_{k+1}=x_k+\eta_{opt}(k)e(k)f^{(m_i)}(x_k)$

**Step 4:** If $J\le Tol$ then

**OUTPUT** ($x_{k+1}$); (The procedure was successful.)

**STOP**

**Step 5:** Set $x_k=x_{k+1}$; $k=k+1$

Go back to step 2

**Step 6: OUTPUT** ('the method failed after $N$ iterations, $n=$' $k$);

(The procedure was unsuccessful.)

**STOP.**

*D. Results and discussion*

In order to confirm the validity of the algorithm proposed, we will give three examples.

***Example 5***: Consider $f(x)=e^x-x-1$ [1]. Since $f(0)=e^0-0-1=0$ and $f'(0)=e^0-1=0$,

but $f''(0) = e^0 = 1$, $f$ has a zero of multiplicity two at $p = 0$.

The table 7 shows the results of the method proposed and the modified Newton's method. The results in table7 illustrate that the algorithm proposed is much more accurate than the modified Newton's method. It is obvious that the method proposed can find zeros of nonlinear equation.

TABLE 7
THE RESULTS OF THE EXAMPLE 5

| Algorithm proposed | | The modified Newton's method [1] | |
|---|---|---|---|
| k | $x_k$ | k | $x_k$ |
| 0 | 1.0000000 | 0 | 1.0000000 |
| 1 | 3.6787944e-1 | 1 | -2.3421061e-1 |
| 2 | 6.0080069e-2 | 2 | 8.4582788e-3 |
| 3 | 1.7691994e-3 | 3 | 1.1889524e-5 |
| 4 | 1.5641108e-6 | 4 | 6.8638230e-6 |
| 5 | 1.2232654e-12 | 5 | 2.8085217e-7 |
| 6 | 2.1676421e-17 | - | - |

***Example 6***: In order to verify the validity finding double zeros using the method proposed, we give a polynomial as follows:

$$f(x) = (x+1)^2(x+2+j)^2(x+2-j)^2$$
$$= x^6 + 10x^5 + 43x^4 + 100x^3 + 131x^2 + 90x + 25$$

Which has one double real zero at -1, and two double conjugate-complex zeros at $-2 \pm j$. Using the algorithm proposed produces the results in table 8.

The results in table 8 show that the algorithm proposed has very high accuracy in the field of finding double zeros of polynomials. It can find not only double real zeros, but also double conjugate-complex zeros.

TABLE 8
THE RESULTS OF THE EXAMPLE 6

| $x_0$ | k | $x_k$ | $\lvert x_k - p_i \rvert$ |
|---|---|---|---|
| -1000 | 42 | -1.0000000000 (exact value) | 0.00000000 |
| 1000 | 38 | -1.0000000000 (exact value) | 0.00000000 |
| 0.0 | 9 | -1.0000000000 (exact value) | 0.00000000 |
| -3+3i | 12 | -2.000000000 + 1.000000000i (exact value) | 0.00000000 |
| -3-3i | 12 | -2.000000000 - 1.000000000i (exact value) | 0.00000000 |

***Example 7***: The eighth-degree polynomial

$$f(x) = (x+2)^4(x+1+2j)^2(x+1-2j)^2$$
$$= (x+2)^4[(x+1)^4 + 8(x+1)^2 + 16]$$

has one multiplicity four real zero at $-2$, and two double conjugate-complex zeros, at $-1 \pm 2j$. Using the method proposed produces the results in table9.

TABLE 9
THE RESULTS OF THE EXAMPLE 7

| Initial values $x_0$ | k | $x_k$ | $\lvert x_k - p_i \rvert$ |
|---|---|---|---|
| ± 1000 | 37 | -2.0000000000 (exact value) | 0.000 |
| -3.0 | 1 | -2.6440366972 | 0.644 |
| | 2 | -2.3647168555 | 0.365 |
| | 3 | -2.1607425715 | 0.161 |
| | 4 | -2.0413652042 | 0.041 |
| | 5 | -2.003246032 | 3.246e-3 |
| | 6 | -2.000020991 | 2.099e-5 |
| | 7 | -2.000000000 | 0.88e-9 |
| | 8 | -2.000000000 (exact value) | 0.000 |
| 0+3i | 1 | -0.235315507 + 2.657230278i | 1.008 |
| | 2 | -0.438921130 + 2.383656064i | 0.680 |
| | 3 | -0.615532340 + 2.175393618i | 0.422 |
| | 4 | -0.769319340 + 2.031719015i | 0.233 |
| | 5 | -0.902586134 + 1.957131458i | 0.106 |
| | 6 | -1.001191303 + 1.964823096i | 0.035 |
| | 7 | -1.001688399 + 2.004554906i | 0.486e-2 |
| | 8 | -1.000068342 + 2.000038269i | 0.783e-4 |
| | 9 | -1.000000013 + 1.999999984i | 2.068e-8 |
| | 10 | -1.000000000 + 2.000000000i | 1.510e-15 |
| | 11 | -1.000000000 + 2.000000000i (exact value) | 0.000 |
| 0-3i | 11 | -1.000000000 - 2.000000000i (exact value) | 0.000 |
| 1+10i | 18 | -1.000000000 +2.000000000i (exact value) | 0.000 |
| 1-10i | 18 | -1.000000000 - 2.000000000i (exact value) | 0.000 |

## V. CONCLUDING REMARKS

We know from the table 1 to table 5 that the algorithm proposed in this paper can choose the initial approximation in very large scope, without the derivative evaluation, and less computation with high precision and rapid convergence. Specially, the results of the table 5 showed that the algorithm presented can obtain the exact value of roots which other methods can never do. The comparison results of the table 6 showed that the approach proposed has faster convergence and higher accuracy than other methods. We can know from the table 7 to table 9 that the algorithm proposed can rapidly and precisely calculate the multiple real and complex roots of polynomials or nonlinear equation which were not solved by other traditional methods at all. All the results in three examples have high precise values with

less computation. Especially, the results both in table 8 and table 9 can produce exact values. Furthermore, the two novel algorithms proposed can be convergent when an initial approximation was selected in a large scope. Hence, the algorithms proposed will play a very important role in the many fields of science and engineering practice.

## REFERENCES


[1] Richard L. Burden, J. Douglas Faires. Numerical ANALYSIS(Seventh Edition)[M].Thomson Learning, Inc.Aug. 2000，pp47-103

[2] Zeng Zhe-zhao, Wen Hui. Numerical Computation(First Edition)[M]. Beijing: Qinghua University Press，China, Sept. 2005，pp88-108

[3] Xu Changfa, Wang Minmin and Wang Ninghao. An accelerated iteration solution to nonlinear equation in large scope [J]. J. Huazhong Univ. of Sci. & Tech.(Nature Science Edition), 34(4):122-124, 2006

[4] Ni Youying. A semi-covered Iteration Method to solve non-linear equation in one variable [J]. Transaction of Haerbin University of Science and Technology，1998，3(2)：83~86

[5] Wait R. The Numerical Solution of Algebraic Equations [M]. New York: John Wiley & Sons, Ltd., 1979

[6] Wu Xinyuan. A Significant Improvement on Newton's Iterative Method [J]. Applied mathematics and mechanics (China), 1999, 20(8):963-966

[7] T.N. Lucas. Finding roots of polynomials by using the Routh array. IEEE Electronics Letters, 32(16):1519-1521, Aug. 1996

[8] T.K. Truong, J.H. Jeng, and I.S. Reed. Fast algorithm for computing the roots of error locator polynomials up to degree 11 in Reed-Solomon decoders. IEEE Trans. Commun., vol.49, pp.779-783, May 2001

[9] Sergei V. Fedorenko, Peter V. Trifonov. Finding roots of polynomials over finite fields. IEEE Trans. Commun. 50(11):1709-1711, Nov. 2002

[10] Cui Xiang-zhao, Yang Da-di and Long Yao. The fast Halley algorithm for finding all zeros of a polynomial. Chinese Journal of engineering mathematics, 23(3):511-517, June 2006

[11] Ehrlich L.W. A modified Newton method for polynomials. Comm ACM, 10: 107-108, 1967

[12] Huang Qing-long. An improvement on a modified Newton method. Numerical mathematics: A Journal of Chinese Universities, 11(4):313-319, Dec. 2002

[13] Huang Qing-long, Wu Jiancheng. On a modified Newton method for simultaneous finding polynomial zeros. Journal on Numerical methods and computer applications(Beijing, China), 28(4):292-298, Dec. 2006



**ZENG Zhe-zhao** (1963-9) received the B.S., M.S. and D.S. degree from Xiangtan University in 1987, Tsinghua University in 1989 and Hunan University in 2008 respectively. From 1990 to 1993, he worked in Dongguang, Guangdong , China, where he was a technical manager from 1991 to 1993. From 1993 to 1999, he worked in Xiangtan University of Science & Technology, China. From 2000 to now, he has worked in Changsha University of Science & Technology, China. Since 2002, he has been a professor. He is engaging in theory and novel technology of the electronics and information engineering, principally in research on signal processing , theory and application of neural networks, and PID intelligent control.

Email:hncs6699@yahoo.com.cn; hncsdlxdz@vip.sina.com