[1]The Neural-Network Approaches to Solve Nonlinear Equation

Xiangde GUO
College of Electrical & Information Engineering, Changsha University of Science & Technology, Changsha, China
Email: guoxiangde@126.com

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

***Abstract*—In this paper, we proposed two neural-network approaches for solving nonlinear equations or polynomials. The first method is suitable for finding simple roots of nonlinear equation or polynomial, and the second approach is fit to finding both the multiple and simple roots of nonlinear equations or polynomial, which were not well solved by the other methods. The convergence of algorithm proposed was researched. The convergence theorem provides the theory criterion selecting learning rate of neural network. The specific examples showed that the proposed method can find the simple or multiple roots of nonlinear equations or polynomials at a very rapid convergence and very high accuracy with less computation.**

***Index Terms*—neural-network, nonlinear equations, polynomials, multiple and simple roots**

I. INTRODUCTION

Finding rapidly and accurately the roots of nonlinear equations is an important problem in various areas of control and communication systems engineering, signal processing and in many other areas of science and technology. The problem of finding the zeros of nonlinear equations has fascinated mathematicians for centuries, and the literature is full of ingenious methods, analysis of these methods, and discussions of their merits [1-3]. Over the last decades, there exist a large number of different methods for finding nonlinear equations roots either iteratively or simultaneously. Most of them yield accurate results only for small degree or can treat only special polynomials, e.g., polynomials with simple real or complex roots [4].

So far, some better modified methods of finding roots of nonlinear equations or polynomials cover mainly the Jenkins/Traub method [5], the Markus/Frenzel method [4], the Laguerre method [6], 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. Although the Laguerre method is faster convergent than all other methods mentioned above, it has more computation. 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.

In this paper, we consider a neural network algorithm to find a simple root α, i.e., $f(\alpha)=0$ and $f'(\alpha)\neq 0$, of a nonlinear equation $f(x)=0$.

Newton's method is the well-known iterative method for finding α by using

$$x^{k+1}=x^k-\frac{f(x^k)}{f'(x^k)} \tag{1}$$

that converges quadratically in some neighborhood of α [14-15].

In recent years, some iterative methods have been proposed and analyzed for solving nonlinear equations that improve order of convergence of the classical methods such as Jarratt's method, Euler-Chebyshev methods, Ostrowski's method [14] given by

$$y^k=x^k-\frac{f(x^k)}{f'(x^k)} \tag{2}$$

$$x^{k+1}=y^k-\frac{f(x^k)}{f(x^k)-2f(y^k)}\frac{f(y^k)}{f'(x^k)} \tag{3}$$

which are improvements of Newton's method; the order increases by at least two at the expense additional function evaluation at another point iterated by the classical methods [15].

Grau et al. [16] developed the sixth-order variant of Ostrowski's method which is given by

$$y^k=x^k-\frac{f(x^k)}{f'(x^k)} \tag{4}$$

$$z^k=y^k-\frac{f(x^k)}{f(x^k)-2f(y^k)}\frac{f(y^k)}{f'(x^k)} \tag{5}$$

$$x^{k+1}=z^k-\frac{f(x^k)}{f(x^k)-2f(y^k)}\frac{f(z^k)}{f'(x^k)} \tag{6}$$

Supported by the key project of the Education Department of Hunan Province (08A006)

doi:10.4304/jcp.5.3.410-416

Another sixth-order family of modified Ostrowski's method was considered by Sharma et al. [17].

$$y^k = x^k - \frac{f(x^k)}{f'(x^k)} \tag{7}$$

$$z^k = y^k - \frac{f(x^k)}{f(x^k)-2f(y^k)}\frac{f(y^k)}{f'(x^k)} \tag{8}$$

$$x^{k+1} = z^k - \frac{f(x^k)+(\beta-2)f(y^k)}{f(x^k)+\beta f(y^k)}\frac{f(z^k)}{f'(x^k)} \tag{9}$$

where $\beta \in \mathbf{R}$

Changbum Chun and YoonMee Ham proposed some sixth-order variant of Ostrowski's method which is given by [15]

$$y^k = x^k - \frac{f(x^k)}{f'(x^k)} \tag{10}$$

$$z^k = y^k - \frac{f(x^k)}{f(x^k)-2f(y^k)}\frac{f(y^k)}{f'(x^k)} \tag{11}$$

$$x^{k+1} = z^k - H(u^k)\frac{f(z^k)}{f'(x^k)} \tag{12}$$

where $u^k = f(y^k)/f(x^k)$ and $H(t)$ represents a real-valued function, it is observed that if we take $H(t) = 1/(1-2t)$ or $H(t) = [1+(\beta+2)t]/(1+\beta t)$, then the substep defined by (12) reduces to (6) and (9), respectively.

In order to solve the problems finding the multiple roots of nonlinear equations or polynomial, Dong [18], Neta and Johnson[19], Neta [20], etc. developed some effective methods.

In this paper, we proposed two neural-network methods for finding simple or multiple roots of nonlinear equations or polynomials. In section 2 and 3 we constructed a neural network algorithm and analyzed its convergence, respectively. We give the algorithm steps in section 4. The efficiency of the considered method was illustrated by numerical examples in section 5.

II. CONSTRUCTION OF THE NEURAL NETWORK ALGORITHM

A. For the function f(x) with only simple roots

We suppose that x^0 is close initial approximations of the root α and x^k is a weight of nonlinear equation $f(x)=0$. The algorithm of neural network is as follows

Algorithm 2.1. Define the error function and the objective function as follows

$$e(k) = 0 - f(x^k) = -f(x^k) \tag{13}$$

$$J = \frac{1}{2}e^2(k) \tag{14}$$

To minimize the objective function J, the weight x^k is recursively calculated via using a simple gradient descent rule:

$$y^k = x^k - \mu(k)\frac{dJ}{dx^k} \tag{15}$$

where $\eta(k)$ is learning rate and $0<\mu(k)<1$, and $\Delta x^k = -\mu(k)\frac{dJ}{dx^k}$.

Differentiating (14) with respect to x^k, it can be obtained that

$$\Delta x^k = -\mu(k)\frac{\partial J}{\partial x^k} = \mu(k)e(k)\ f'(x^k) \tag{16}$$

Substituting (16) into (15), we have

$$y^k = x^k + \mu(k)e(k)f'(x^k) \tag{17}$$

According to [3], let

$$z^k = y^k - \frac{f(x^k)}{f(x^k)-2f(y^k)}\frac{f(y^k)}{f'(x^k)} \tag{18}$$

Then

$$x^{k+1} = z^k - \frac{f(x^k)}{f(x^k)-2f(y^k)}\frac{f(z^k)}{f'(x^k)} \tag{19}$$

B. For the function f(x) with multiple roots

Consider the nonlinear equation or polynomial of the type

$$f(x) = 0 \tag{20}$$

We assume that x is a multiple root of the Eq. (20) and x_0 is an initial guess sufficiently close to x. Using the Taylor's series expansion of the function $f(x)$, we have

$$f(x_0) + (x-x_0)f'(x_0) + \tfrac{1}{2}(x-x_0)^2 f''(x_0) = 0 \tag{21}$$

This alternative equivalence formulation has been used to develop a class of iterative methods for solving nonlinear equation or polynomial $f(x)=0$. We can rewrite (21) in the following form:

$$f''(x_0)(x-x_0)^2 + 2f'(x_0)(x-x_0) + 2f(x_0) = 0 \tag{22}$$

This is a quadratic equation. From which we have

$$x - x_0 = \frac{-2f'(x_0) \pm \sqrt{4[f'(x_0)]^2 - 8f''(x_0)f(x_0)}}{2f''(x_0)} \tag{23}$$

or

$$x - x_0 = \frac{-f'(x_0) \pm \sqrt{[f'(x_0)]^2 - 2f''(x_0)f(x_0)}}{f''(x_0)} \tag{24}$$

This formula gives two possibilities for x, depending on the sign proceeding the radical term. Thus,

$$x = x_0 - \frac{f'(x_0) + \sqrt{[f'(x_0)]^2 - 2f(x_0)f''(x_0)}}{f''(x_0)} \tag{25}$$

$$x = x_0 - \frac{f'(x_0) - \sqrt{[f'(x_0)]^2 - 2f(x_0)f''(x_0)}}{f''(x_0)} \tag{26}$$

This fixed point formulation allows us to suggest the following method.

Algorithm 2.2. For a given x_0, compute approximate solution x_{k+1} by the iterative schemes

For the smaller root, using the following iterative formula

$$x_{k+1} = x_k - \frac{f'(x_k) + \sqrt{[f'(x_k)]^2 - 2f(x_k)f''(x_k)}}{f''(x_k)} \tag{27}$$

and for the larger root, using the following iterative formula

$$x_{k+1} = x_k - \frac{f'(x_k) - \sqrt{[f'(x_k)]^2 - 2f(x_k)f''(x_k)}}{f''(x_k)} \tag{28}$$

Since the denominator of both formula (27) and (28) does not include the term of $f'(x_n)$, they are specially fit to finding double roots of nonlinear equation or polynomial.

III. Analysis of Convergence

In this section we analyzed the convergence property of the **Algorithm 2.1**. The following theorem deals with the choice of the variable parameters $\eta(k)$.

Theorem 1. Let $f(x) = 0$ be a nonlinear equation or polynomial, with simple root α. If x^0 is the initial approximations of the root α, only when the parameter satisfies

$$0 < \mu(k) < 2/\left|f'(x^k)\right|^2 \tag{29}$$

The **Algorithm 2.1** is convergent.

Proof. Define an error function and a *Lyapunov* function respectively as follows

$$V(k) = \frac{1}{2}e^2(k) \tag{30}$$

Then we have

$$\Delta V(k) = \frac{1}{2}e^2(k+1) - \frac{1}{2}e^2(k) \tag{31}$$

Since

$$e(k+1) = e(k) + \Delta e(k) = e(k) + \frac{de(k)}{dx^k}\Delta x^k \tag{32}$$

and

$$\Delta e(k) = \frac{de(k)}{dx^k}\Delta x^k = -f'(x^k)\Delta x^k \tag{33}$$

According to formula (31) , (32), and (33), we have

$$\begin{aligned}\Delta V(k) &= \Delta e(k)e(k) + \frac{1}{2}[\Delta e(k)]^2 \\ &= -e(k)f'(x^k)\Delta x^k + \frac{1}{2}[f'(x^k)\Delta x^k]^2\end{aligned} \tag{34}$$

Known from the (16) that

$$\Delta x^k = \mu(k)e(k)\ f'(x^k) \tag{35}$$

Substituting formula (35) into the formula (34) gives

$$\begin{aligned}\Delta V(k) = &-[e(k)f'(x^k)]^2\mu(k) \\ &+ \frac{1}{2}\left\{\mu(k)e(k)[f'(x^k)]^2\right\}^2\end{aligned} \tag{36}$$

If the formula (15) is convergent, i.e. $\Delta V(k) < 0$, then it is easy to see from (36) that

$$0 < \frac{1}{2}\left\{\mu(k)e(k)[f'(x^k)]^2\right\}^2 < [e(k)f(x^k)]^2\mu(k) \tag{37}$$

Since $\mu(k) > 0$, hence we have from formula (37)

$$0 < \mu(k) < \frac{2}{\left|f'(x^k)\right|^2} \tag{38}$$

which proves the theorem.

IV. Algorithm2.1 Steps

To find zero of nonlinear equation or polynomial $f(x) = 0$ given one approximation x^0:

INPUT: x^0 ; tolerance Tol ; maximum number of iterations N ; let $k = 0$;

OUTPUT: approximate solution x^k or message of failure.

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

Step 2: compute: $e(k) = -f(x^k)$, $f'(\mathbf{x}^k)$, and

$$\mu_{opt}(k) = \begin{cases} 0.5, & \left|f'(x^k)\right| < 1 \\ (1.0-1.6)/\left|f'(x^k)\right|^2, & \left|f'(x^k)\right| \ge 1 \end{cases}$$

Step 3: Estimate: $y^k = x^k + \mu(k)e(k)f'(x^k)$,

$\hat{e}(k) = -f(y^k)$, and

$z^k = y^k - \mu(k)e(k)\hat{e}(k)f'(x^k)/[2\hat{e}(k) - e(k)]$;

Update weight:

$x^{k+1} = z^k + \mu(k)e(k)f'(x^k)f(z^k)/[2\hat{e}(k) - e(k)]$

Step 4: If $\left|f(x^{k+1})\right| \le Tol$ then

OUTPUT (x^{k+1}); (The procedure was successful.)

STOP

Step 5: Set $k = k + 1$

Go back to step 2

Step 6: OUTPUT ('the method failed after N iterations, $n = ' k$);
(The procedure was unsuccessful.)
STOP

V. NUMERICAL EXAMPLES

A. For the function f(x) with only simple roots

To confirm the validity of the algorithm2.1 proposed, we gave two examples to evaluate the polynomial or nonlinear equation at the initial values, and compared results with other methods.

Example 1. Let

$$f(z) = z^7 + z^5 - 10z^4 - z^3 - z + 10$$

with zeros $\xi_1 = 2$, $\xi_2 = 1$, $\xi_3 = -1$, $\xi_4 = i$, $\xi_5 = -i$, $\xi_6 = -1 + 2i$, $\xi_7 = -1 - 2i$. As initial approximations, the following complex numbers have been chosen [8]:

$z_1^0 = 1.66 + 0.23i$, $z_2^0 = 1.36 - 0.31i$,
$z_3^0 = -0.76 + 0.18i$, $z_4^0 = -0.35 + 1.17i$,
$z_5^0 = 0.29 - 1.37i$, $z_6^0 = -0.75 + 2.36i$,
$z_7^0 = -1.27 - 1.62i$.

The Table 1 display the absolute errors:
$e_i^k = \left| z_i^k - \xi_i \right|$ ($i = 1,2,3,4,5,6,7$) using the neural network algorithm proposed and the formula (11) in [21]. Where, the formula (11) in [21] is as follows

$$x_i^{k+1} = x_i^k - \frac{2W_i^k}{1 + s_i^k + p_i^k}$$

$$p_i^k = \sqrt{(1+s_i^k)^2 + 4W_i^k \sum_{j \neq i} \frac{u_j^k}{(x_i^k - x_j^k)(x_i^k - W_i^k - x_j^k)}}$$

$$s_i^k = \sum_{j \neq i} \frac{W_j^k}{(x_i^k - x_j^k)}, \text{ and } u_j^k = \frac{f(y_j^k)}{f(x_j^k)}.$$

TABLE 1
THE ABSOLUTE ERRORS FOR EXAMPLE 1

Method proposed		Iteration formula (11) [21]	
e_i^k	k=12	e_i^k	k=3
e_1^k	exact	e_1^k	0.296524e-13
e_2^k	exact	e_2^k	0.292952e-13
e_3^k	exact	e_3^k	0.582335e-20
e_4^k	0.674498 e-18	e_4^k	0.750127e-18
e_5^k	0.370194 e-18	e_5^k	0.335764e-17
e_6^k	exact	e_6^k	0.111022e-15
e_7^k	exact	e_7^k	0.000000e+00

Example 2. Let

$$f(z) = z^4 - 1$$

with zeros $\xi_1 = 1$, $\xi_2 = -1$, $\xi_3 = i$, $\xi_4 = -i$. As initial approximations, the following complex numbers have been chosen [21]:

$z_1^0 = 0.5 + 0.5i$, $z_2^0 = -1.36 + 0.42i$,
$z_3^0 = -0.25 + 1.28i$, $z_4^0 = 0.46 - 1.37i$

The Table 2 displays the absolute errors:
$e_i^k = \left| z_i^k - \xi_i \right|$ ($i = 1,2,3,4$) using the neural network algorithm proposed and the formula (11) in [21] .

TABLE 2
THE ABSOLUTE ERRORS FOR EXAMPLE 2

Method proposed*		Iteration formula (11) [21]	
e_i^k	k=4	e_i^k	k=3
e_1^k	exact	e_1^k	0.620385 e-24
e_2^k	exact	e_2^k	exact
e_3^k	exact	e_3^k	0.206795 e-24
e_4^k	exact	e_4^k	exact

* $z_1^0 = 0.75 + 0.25i$ in method proposed.

B. For the function f(x) with multiple roots

To confirm the validity of the algorithm2.2 proposed, we gave five examples to evaluate the polynomial or nonlinear equation at the initial values, and compared results with other methods.

For the convenience of comparison, all methods in reference [20] are listed as follows.

(1) Halley's method

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

Where m notes multiplicity m of nonlinear equations or polynomials.

The Halley's method is cubically convergent, and is a special case of the Hansen and Patrick's method.

(2) Chebyshev's method

$$x_{n+1} = x_n + \frac{m(m-3)}{2} u_n \left[1 - \frac{m}{m-3} u_n \frac{f''(x_n)}{f'(x_n)} \right]$$

where $u_n = \frac{f(x_n)}{f'(x_n)}$.

(3) The modified Chebyshev's method

$$x_{n+1} = x_n - u_n \left[\beta + \gamma \frac{f(y_n)}{f(x_n)} \right]$$

where

$$y_n = x_n - \alpha u_n,\ \alpha = -\frac{m(m-3)}{2},\ \beta = -\frac{m}{(m-3)}$$

$$\gamma = \frac{m(m-\alpha)}{\rho\alpha^2}, \rho = \left(\frac{m-\alpha}{m}\right)^m.$$

(4) Neta's method

$$x_{n+1} = x_n + \frac{m(m-3)}{2}u_n\left[1 - \frac{m}{m-3}u_n w_n\right]$$

where

$$w_n = \frac{6[f(x_{n-1}) - f(x_n)] + 2hf'(x_{n-1}) + 4hf'(x_n)}{h^2 f'(x_n)},$$

$$h = x_n - x_{n-1}.$$

(5) The modified Osada method

$$x_{n+1} = x_n - \frac{m(m+1)}{2}u_n + \frac{(m-1)^2}{2w_n}$$

Example 3. In third example we took a quadratic polynomial having a double root at $\xi = 1$[20]

$$f(x) = x^2 - 2x + 1$$

The iteration starts with $x_0 = 0$ and the results are summarized in Table3.

TABLE 3
COMPARISON OF 7 METHODS FOR EXAMPLE 3

Method	Initial value $x_0 = 0$	
	Iterations: n	$\lvert f(x_n)\rvert$
Proposed method (27)	1	0
Proposed method (28)	1	0
Halley's method (2) in reference [20]	1	0
Chebyshev's method (29) in ref. [20]	1	0
The modified Chebyshev's method (39) in ref. [20]	1	0
Neta's method (49) in reference [20]	2	0
The modified Osada method (51) in reference [20]	2	0

Example 4. In the fourth example we took a polynominal having two double roots at $\xi = \pm 1$[20]

$$f(x) = x^4 - 2x^2 + 1$$

The iteration starts respectively with $x_0 = 0.8$ or $x_0 = 0.6$. The results are summarized in Table 4.

Example 5. The fifth example is a polynomial with triple root at $\xi = 1$[20]

$$f(x) = x^5 - 8x^4 + 24x^3 - 34x^2 + 23x - 6$$

The iteration starts with $x_0 = 0$ and the results are summarized in Table 5.

TABLE 4
COMPARISON OF 7 METHODS FOR EXAMPLE 4

Method	$x_0 = 0.8$		$x_0 = 0.6$	
	Iters. n	$\lvert f(x_n)\rvert$	Iters. n	$\lvert f(x_n)\rvert$
Proposed method (27)	6	0	7	0
Proposed method (28)	9	0	15	0
Halley's method (2) in ref. [20]	4	0	4	6.0e-15
Chebyshev's method (29) in ref. [20]	4	0	4	5.8e-15
The modified Chebyshev's method (39) in ref. [20]	3	0	3	2.0e-20
Neta's method (49) in ref. [20]	5	0	5	0
The modified Osada method (51) in ref [20]	5	0	5	0

TABLE 5
COMPARISON OF 7 METHODS FOR EXAMPLE 5

Method	Initial value $x_0 = 0$	
	Iterations: n	$\lvert f(x_n)\rvert$
Proposed method (27)	1	0
Proposed method (28)	22	0
Halley's method (2) in ref. [20]	3	1.0e-18
Chebyshev's method (29) in reference [20]	3	1.0e-15
The modified Chebyshev's method (39) in reference [20]	3	0
Neta's method (49) in ref. [20]	7	1.0e-14
The modified Osada method (51) in ref. [20]	9	0

Example 6. The sixth example is a nonlinear equation with double root at $\xi = 0$ [20]

$$f(x) = x^2 e^x$$

Starting at $x_0 = 0.1$ or at $x_0 = 0.2$, the results are given in Table 6.

Example 7. The seventh example having a double root at $\xi = 1$ is [20]

$$f(x) = 3x^4 + 8x^3 - 6x^2 - 24x + 19$$

We started with $x_0 = 0$ and $x_0 = 0.5$ and the results are summarized in Table 7.

TABLE 6
COMPARISON OF 7 METHODS FOR EXAMPLE 6

Method	$x_0 = 0.1$		$x_0 = 0.2$	
	Ites. n	$\lvert f(x_n)\rvert$	Iters. n	$\lvert f(x_n)\rvert$
Proposed method (27)	7	2.4e-35	8	0.94e-38
Proposed method (28)	7	1. 2e-36	8	0.62e-40
Halley's method (2) in ref. [20]	2	4.0e-26	2	0.80e-20
Chebyshev's method (29) in ref. [20]	2	2.0e-22	2	3.0e-17
The modified Chebyshev's method (39) in ref. [20]	2	2.0e-22	2	3.0e-17
Neta's method (49) in ref. [20]	4	3.0e-15	5	5.0e-26
The modified Osada method (51) in ref. [20]	4	1.0e-14	5	1.0e-24

TABLE 7
COMPARISON OF 7 METHODS FOR EXAMPLE 7

Method	$x_0 = 0$		$x_0 = 0.5$	
	Iterations: n	$\lvert f(x_n)\rvert$	Iterations: n	$\lvert f(x_n)\rvert$
Proposed method (27)	10	0	9	0
Proposed method (28)	20	1.8e-14	11	1.3e-35
Halley's method (2) in ref. [20]	3	1.0e-18	3	0
Chebyshev's method (29) in ref. [20]	5	0	3	0
The modified Chebyshev's method (39) in ref. [20]	4	1.0e-18	3	0
Neta's method (49) in ref. [20]	5	0	5	1.0e-18
The modified Osada method (51) in ref. [20]	5	2.0e-18	5	1.0e-18

Example 8. The sixth example having two simple roots at $\xi = 2$ and $\xi = 3$ is

$$f(x) = x^2 - 5x + 6$$

Starting at $x_0 = 1$ or at $x_0 = 4$, the results are given in Table 8.

TABLE 8
COMPARISON OF 8 METHODS FOR EXAMPLE 8

Method	$x_0 = 0$		$x_0 = 4$	
	Iters. n	$\lvert f(x_n)\rvert$	Iters. n	$\lvert f(x_n)\rvert$
Proposed method (27)	1	0	1	0
Proposed method (28)	1	0	1	0
Halley's method (2) in ref. [20]	6	0	4	0
Chebyshev's method (29) in ref. [20]	5	0	100	1.8e-15
The modified Chebyshev's method (39) in ref. [20]	8	0	6	0
Neta's method (49) in ref. [29]	4	0	3	0
The modified Osada method (51) in ref. [20]	8	0	6	0

VI. CONCLUDING REMARKS

We can see from the table 1 to table 2 that the method proposed can simultaneously find the real or complex roots of polynomials with fast convergence and very high accuracy. Although the iterative numbers using the method proposed are four times as same as the iterative formula (11) in [21] , see Table 1, the iterative formula (11) in [21] has more computation than the method proposed at each iterative. The results in example 2 show that the method proposed has much higher accuracy and faster convergence than the iterative formula (11) in [21]. Furthermore, we have developed two new methods to obtain not only multiple roots but also simple roots. We can know from the table 3 to table 8 that the numerical experiments show the rapid convergence and high accuracy for finding the multiple or simple roots of polynomials or nonlinear equations compared with other methods. It is specially to say that the methods proposed are more efficient than other methods in finding roots of quadratic polynomial or nonlinear equation with double or simple roots (see Table3 and Table8). Hence, the algorithm2.2 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).Thomson Learning, Inc. Aug. 2001，pp47-103
[2] Zeng Zhe-zhao, Wen Hui. Numerical Computation (First Edition). 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. Huazhong Univ. of Sci. & Tech.(Nature Science Edition), 34(4):122-124, 2006
[4] Markus Lang and Bernhard-Christian Frenzel. Polynomial root finding. IEEE Signal Processing Letters, 1(10):141-143, Oct. 1994
[5] Jenkins, M. A. and J. F. Traub. *A three-stage algorithm for real polynomials using quadratic iteration.* SIAM Journal on Numerical Analysis 7, No.4(1970), 545-566
[6] H.J. Orchard. The Laguerre method for finding the zeros of polynomials. IEEE Trans. On circuits and Systems, 36(11):1377-1381, Nov. 1989
[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
[14] A.M. Ostrowski, Solution of equations in Euclidean and Banach space, Academic Press, New York,1973.
[15] Changbum Chun, YoonMee Ham, Some sixth-order variants of Ostrowski root-finding methods, *Appl. Math. Comput.* 193(2007)389-394.
[16] M. Grau, J.L. Díaz-Barrero, An improvement to Ostrowski root-finding method, *J.Math. Anal. Appl.* 173(2006)450 -456.
[17] J.R. Sharma, R.K. Guha, A family of modified Ostrowski methods with accelerated sixth-order convergence, *Appl. Math. Comput.* (in press). Doi: 10.1016/j.amc.2007.01.009.
[18] C. Dong. A family of multipoint iterative functions for finding multiple roots of equations. Int. J. Comput. Math. 21(1987)363-367.
[19] B. Neta, A.N.Johnson. High order nonlinear solver for multiple roots. Comput. Math. Appl., doi: 10.1016/j.camwa. 2007.09.001
[20] B. Neta. New third order nonlinear solver for multiple roots. Appl. Math. Comput., doi: 10.1016/j. amc.2008.01.031
[21] Xin Zhang, Hong Peng, Guiwu Hu, A high order iteration formula for the simultaneous inclusion of polynomial zeros, *Appl. Math. Comput.* 179 (2006)545 -552.

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