Finding Multiple Real Roots by Neural Networks Based on Complete Discrimination System of Polynomial

Xinli Zhang*†, Dayong Zhu†, and Wang Hu†
†Computational Intelligence Laboratory, School of Computer Science and Engineering
University of Electronic Science and Technology of China, Chengdu 610054, P. R. China
*Department of Computation Science, Chengdu University of Information Technology, Chengdu, 610225, P. R. China
Email: zxl@cuit.edu.cn, dayongzhu75@163.com, scuhuwang@126.com

***Abstract*— A new method of solving the multiple real roots of polynomial by neural networks is proposed in this paper. This method combines the symbolic method with the numerical method. Based on the complete discrimination system of polynomial, the number and multiplicities of the distinct real roots of polynomials can be explicit determined. According to the number of the distinct real roots, a neural networks model for finding the multiple real roots of polynomial is established. From the description of the new model, it is not difficult to find that the existent neural networks for finding real roots of polynomial is the special case of the new one, where all of the real roots are treated as different values. Through training the new model by the gradient descent method, the approximate real roots of polynomial can be obtained. From the simulation results, it is shown that, comparing to the existent neural networks of finding real roots, the new method is not only more effective, but also can avoid the inequality between two or more equal real roots after finishing to solve the polynomial.**

I. Introduction

In many branches of science and engineering, the resolution of a problem coming from practice is often reduced to the search of a solution for an equation or a system of equations modeling the considered problem [1]–[6]. So it is of theoretical and practical significance to solve the equation or the system of equations. Solving an equation or a system of equations can be related to the existence or non-existence of complex or real solutions, to the number of real or complex solution, to the approximation of one or several solutions, etc. Determination for the number of roots in a certain range of a polynomial by an explicit criterion is of characteristic of symbolic computation [7]–[9]. And the approximation solutions solving for an equation or a system of equations belongs to the scope of numerical computation.

Many numerical methods for solving equation have been established, such as the Newton iterating methods, and recursive method, etc. However, the difficulties of accuracy and high processing speed were encountered by all of these methods [2], because these methods need a large amount of iterating time when the degree of polynomial become greater.

The application of neural networks to solving equation is a novel research topic. A neural network model was presented to solve the real roots of polynomials with great degree in [10], [11]. This neural networks have solved some polynomials effectively. However, when there exists some multiple real roots in polynomial, the results obtained by this model will be confusable about these multiple real roots, because all the real roots arrived at will be different from each other. Thus we should begin our study of another effective numerical method for solving the polynomial with multiple real roots .

The classification for the roots of polynomial was once an interesting issue, but was always a difficult and tough problem for the polynomial with degree greater than 4. If the numbers and multiplicities of the distinct real/imaginary roots for polynomials can be settled, solving the approximate real roots will go easier. The complete discrimination system of polynomial presented by Yang. et al. [7]–[9] is a perfect result for the determination of the numbers and multiplicities of the real/imaginary roots for polynomials with symbolic coefficients, which proposed an explicit criterion of the classification for the roots of polynomial with arbitrary degree.

Based on the complete discrimination system of polynomial, a new method of solving multiple real roots for polynomial by neural networks is presented in this paper. The number and the multiplicities of distinct real roots will be determined firstly, and a neural network model of finding multiple real roots (RRFNN) will be constructed in terms of the number of distinct real roots in the next place, and then some training steps will be carried into execution on the new model, and finally, the approximate real roots will be obtained. The results can be derived that the existent neural networks model of real roots finding (RFNN) [10], [11] is the special case of the new one, where all of the real roots are treated as different values. From the simulations, it is easy to find that comparing to the existent neural networks of finding roots, the new one presented in this paper is not only more effective, but also can avoid the inequality between two or more equal real roots after finishing to solve the polynomial.

This paper is organized as follows. In Section II, some preliminaries will be presented. The complete discrimination system of polynomial will be expanded in Section III, which is the basis of the method presented in this paper. In Section IV, the new model for finding the multiple real roots by neural

978–1–4244–1674–5/08/$25.00 © 2008 IEEE

networks and the corresponding algorithm based on the complete discrimination system of polynomial will be proposed. Simulation will be illumined in Section V. Conclusions will be given in Section VI.

II. PRELIMINARIES

A. Problem Presentation

Let $f(x)$ be a polynomial with n order described by:

$$f(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n, \qquad (1)$$

where $n \geq 2, a_0 \neq 0$. In this paper, the n roots of polynomial (1) is supposed as real numbers, which is denoted by $w_1, w_2, \cdots, w_n$.

The purpose of this paper is how to find the multiple real roots of polynomial (1) more effectively and faster.

B. The Neural Networks of Real Roots Finding

A neural network model was used to solve the real roots of polynomials with great order in [10], [11]. Assuming the coefficient a_0 of x^n as 1, then the polynomial (1) can be represented as:

$$f(x) = x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n. \qquad (2)$$

Supposing that there exists n approximate real roots of polynomial (2), which are denoted by w_i's, for $i = 1, 2, \cdots, n$, respectively, then polynomial (2) can be factorized as follows:

$$f(x) = x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n \doteq \prod_{i=1}^{n}(x - w_i). \quad (3)$$

The model [10], [11] is shown in Fig.1. There are three layers consisting of two input nodes corresponding to "1" and x, n hidden nodes of representing the difference between the input x and the connected weight w_i's, for $i = 1, 2, \cdots, n$, and one output node, which performs the operation the same as (3).

The target of training this model is to modify the weights w_i's, for $i = 1, 2, \cdots, n$, between the input node "1" and the hidden nodes, to the approximate real roots of polynomial (2). At the mean time, the other weights will be fixed at value 1's.

Some polynomials will be effectively settled by this model. But when there exists some multiple real roots in polynomial (2), the results obtained by this model will be confusable about these multiple real roots, because the n real roots reached will be different from each other. As a result, it is necessary to search for another effective solving method.

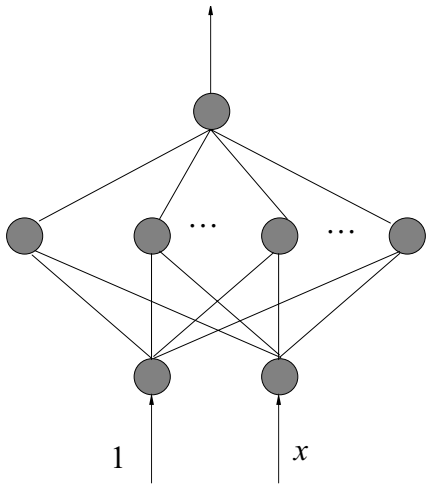

Fig. 1. The neural network model for finding real roots

III. THE COMPLETE DISCRIMINATION SYSTEM OF POLYNOMIAL

A complete discrimination system (CDS) of polynomial was presented by Yang. et al. [7]–[9]. Based on a CDS, an explicit criterion for the determination of the numbers and multiplicities of the real/imaginary roots for polynomials with symbolic coefficients was proposed.

In this section, the complete discrimination system of polynomial will be introduced. For more detail information, see [7]–[9]. Based on this system, a new method to finding real roots of polynomial will be presented in next section.

Definition 3.1: (discrimination matrix)

Given a polynomial (1) with general symbolic coefficients, the following $(2n + 1) \times (2n + 1)$ matrix in terms of the coefficients

$$\begin{bmatrix}
a_0 & a_1 & \ldots & a_n & & & & \\
0 & na_0 & \ldots & a_{n-1} & & & & \\
 & a_0 & \ldots & a_{n-1} & a_n & & & \\
 & 0 & \ldots & 2a_{n-2} & a_{n-1} & & & \\
 & & \cdots & \cdots & & & & \\
 & & \cdots & \cdots & & & & \\
 & & & a_0 & a_1 & \ldots & a_n & \\
 & & & 0 & na_0 & \ldots & a_{n-1} & \\
 & & & & a_0 & \ldots & a_{n-1} & a_n
\end{bmatrix}$$

is called *discrimination matrix* of $f(x)$, and denoted by Discr(f).

By d_k or $d_k(f)$ denote the determinant of the submatrix of Discr(f), formed by the first k rows and the first k columns, for $k = 1, 2, \cdots, 2n + 1$. The discrimination matrix

$$\{d_1, d_2, \cdots, d_{2n+1}\},$$

is called the *principal minor sequence*.

Definition 3.2: (Discriminant Sequence)

Let $D_k = d_{2k}$, for $k = 1, 2, \cdots, n$. The n-tuple

$$\{D_1, D_2, \cdots, D_n\}$$

is called the *discriminant sequence* of polynomial $f(x)$.

Sometimes, the *discriminant sequence* can be denoted by a more detailed notation to specify $f(x)$:

$$\{D_1(f), D_2(f), \cdots, D_n(f)\}.$$

Definition 3.3: (Sign List)
The list

$$\{\text{sign}(D_1), \text{sign}(D_2), \cdots, \text{sign}(D_n)\}$$

is called *sign list* of a given sequence $\{D_1, D_2, \cdots, D_n\}$, where

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases}$$

Definition 3.4: (Revised Sign List)
Given a sign list $[s_1, s_2, \cdots, s_n]$, a new list

$$[\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n]$$

is called the *revised list*, which can be constructed as follows:

(1) If $[s_i, s_{i+1}, \cdots, s_{i+j}]$ is a section of the given list, where

$$s_i \neq 0; s_{i+1} = s_{i+2} = \cdots = s_{i+j-1} = 0; s_{i+j} \neq 0,$$

then, we replace the subsection

$$[s_{i+1}, s_{i+2}, \cdots, s_{i+j-1}]$$

by $[-s_i, -s_i, s_i, s_i, -s_i, -s_i, s_i, s_i, -s_i, \cdots]$, i.e. let

$$\varepsilon_{i+r} = (-1)^{[\frac{r+1}{2}]} \cdot s_i,$$

for $r = 1, 2, \cdots, j-1$.

(2) Otherwise, let $\varepsilon_k = s_k$, i.e. no changes for other terms.

The following theorem is sufficient to determinate the number of the distinct real or imaginary roots:

Theorem 3.1: Given polynomial (1) with real coefficients, if the number of the sign changes of the revised sign list of $\{D_1(f), D_2(f), \cdots, D_n(f)\}$ is ν, then, the number of the pairs of distinct conjugate imaginary roots of $f(x)$ equals ν. Furthermore, if the number of non-vanishing members of the revised sign list is l, then, the number of the distinct real roots of $f(x)$ equals $l - 2\nu$.

Definition 3.5: (Multiple Factor Sequence)
Let M = Discr(f), the discrimination matrix of an n-degree polynomial $f(x)$. By M_k denote the submatrix formed by the first $2k$ rows of M, for $k = 1, 2, \cdots, n$; and $M(k, i)$ denote the submatrix formed by the first $2k-1$ columns and the $(2k+i)$-th column of M_k, for $k = 1, 2, \cdots, n, i = 0, 1, \cdots, n-k$, then, construct polynomials

$$\Delta_k(f) = \sum_{i=0}^{k} \det(M(n-k, i)) x^{k-i},$$

for $k = 0, 1, \cdots, n-1$. The n-tuple

$$\{\Delta_0(f), \Delta_1(f), \cdots, \Delta_{n-1}(f)\}$$

is called the *multiple factor sequence* of $f(x)$.

Lemma 3.1: If the number of the 0's in the revised sign list of the discriminant sequence of $f(x)$ is k, then, $\Delta_k(f) =$ g.c.d.$(f(x), f'(x))$. Thus, the g.c.d.(f, f') is always in the multiple factor sequence of $f(x)$.

Definition 3.6: (Complete Discrimination System)
By U denote the union of

$$\{f(x)\}, \{\Delta_k(f)\}, \{\Delta_j(\Delta_k(f))\}, \{\Delta_i(\Delta_j(\Delta_k(f)))\}, \cdots,$$

etc., that is, all the multiple factor sequences at different levels. Every polynomial of U has a discriminant sequence, and all of them form a *Complete Discrimination System* of $f(x)$, denoted by CDS (f).

IV. The Neural Networks of Finding Multiple Real Roots Based on the Complete Discrimination System of Polynomial

For the convenience of view, in this paper, we take the polynomial (2) with real coefficients and real roots into account. Using of the complete discrimination system mentioned in Section III, the number and the multiplicities of distinct real roots will be determined. Then, a neural network model for finding multiple real roots will be constructed in terms of the number of distinct real roots for polynomial (2).

Assuming that there exist s distinct real roots denoted by $w_1, w_2, \cdots, w_s$, and the multiplicities of which are $r_1, r_2, \cdots, r_s$, respectively, where $r_1 + r_2 + \cdots + r_s = n$.

In order to avoid the inequality between two or more equal real roots after finishing to solve the polynomial arising in [10], [11], a neural network model for finding multiple real roots is presented, which is shown in Fig.2.

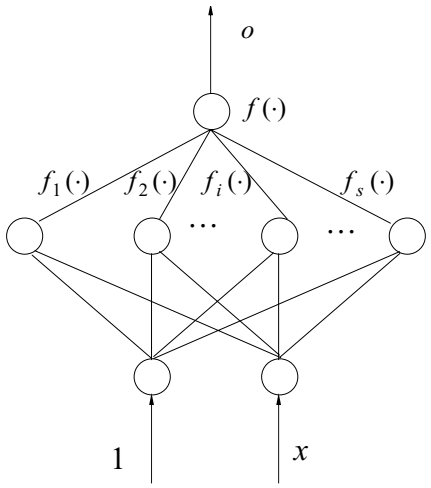

Fig. 2. The neural network model for finding multiple real roots

In this model, there are two input nodes corresponding to the terms of 1 and x, and s hidden nodes, where the number of hidden nodes is just the number of distinct real roots in polynomial (2), and one output node.

The connected weights between the input node 1 and the hidden nodes are denoted by w_i, for $i = 1, 2, \cdots, s$, which

represent the s distinct real roots of polynomial (2). The weights between the input nodes of x and the hidden nodes are set to 1's, so do the weights between the hidden nodes and the output node either.

The training samples are selected from $x \in (-1, 1)$ by average sampling, and the sampling number is denoted by P. That is, there are P training patterns (x_p, y_p), for $p = 1, 2, \cdots, P$, where $y_p = f(x_p)$.

In the training process, for the p'th learning pattern, the state of the i'th hidden nodes is defined by:

$$u_i = x_p - w_i \cdot 1 = x_p - w_i, \tag{4}$$

for $i = 1, 2, \cdots, s$, where w is the weights vector, the i'th component of which is the weight between the input node "1" and the i'th hidden node, that is the i'th root of the polynomial (2).

Then, the output of the i'th hidden node is defined by

$$o_i = f_i(u_i) = u_i^{r_i}, \tag{5}$$

for $i = 1, 2, \cdots, s$, where r_i is the multiplicities of the i'th root.

The output of the output node is defined by

$$o_p = \prod_{i=1}^{s} o_i = \prod_{i=1}^{s} u_i^{r_i}, \tag{6}$$

for $p = 1, 2, \cdots, P$.

Let $E_p = y_p - o_p$, the error cost function of the model in Fig.2 can be defined by

$$E = \frac{1}{2P} \sum_{p=1}^{P} E_p^2 = \frac{1}{2P} \sum_{p=1}^{P} (y_p - o_p)^2. \tag{7}$$

On the foundation of the gradient descent method, we arrive at conclusion as follows:

Theorem 4.1: For the neural networks of finding multiple real roots in Fig.2, when the error cost function is defined by (7), the error E will descend in each iteration step on the basis of the learning rule as follows:

$$\Delta w_i = -\frac{\eta}{P} \sum_{p=1}^{P} \delta_{pi} \cdot o_p, \tag{8}$$

$$\delta_{pi} = r_i (y_p - o_p) \cdot (x_p - w_i)^{-1}, \tag{9}$$

where $\eta > 0$ is the learning rate.

Proof: The gradient of E with regard to w_i is

$$\frac{\partial E}{\partial w_i} = \frac{1}{P} \sum_{p=1}^{P} \left[r_i (y_p - o_p)(x_p - w_i)^{-1} o_p \right].$$

As a result, the modified formulas of weight Δw_i, for $i = 1, 2, \cdots, s$ can be obtained, which is listed by equations (8) and (9).

The proof is complete.

In order to avoid the large output value in output layer, two schemes to transform the formula (6) can be taken into account.

(1) When the sigmoid function

$$g(z) = \frac{1}{1 + e^{-z}},$$

is considered, the following equation can be obtained:

$$\overline{o}_p = g(o_p) = \frac{1}{1 + e^{o_p}}.$$

Then, it is not difficult to arrive at the learning algorithm as follows:

$$\Delta w_i = -\frac{\eta}{P} \sum_{p=1}^{P} \delta_{pi} \cdot \overline{o}_p,$$

$$\delta_{pi} = r_i (\overline{y}_p - \overline{o}_p)(1 - \overline{o}_p) o_p (x_p - w_i)^{-1},$$

where

$$\overline{y}_p = \frac{1}{1 + e^{y_p}}.$$

(2) The logarithmic function $g(z) = \ln |z|$ can be taken as the transformation function either, thus the following equation can be obtained:

$$\overline{o}_p = g(o_p) = \ln |o_p|.$$

And the corresponding learning algorithm follows that:

$$\Delta w_i = -\frac{\eta}{P} \sum_{p=1}^{P} r_i (\overline{y}_p - \overline{o}_p)(x_p - w_i)^{-1},$$

where

$$\overline{y}_p = \ln |y_p|.$$

From the description of the neural networks of finding multiple real roots mentioned above, it can be seen that the neural networks of real roots finding is just the special case of the neural networks for finding multiple real roots, where all of the real roots are treated as different values.

V. Simulation and Discussions

Some examples will be shown to explain the effectiveness and feasibility of the neural networks of finding multiple real roots in this section.

Example 5.1: Give a 4-order polynomial as

$$f_1(x) = x^4 - 4x^3 + 6x^2 - 4x + 1,$$

by using the complete crimination system, it can be determined that there exists one distinct real roots 1.0, which multiplicities is 4. We solve the equation $f_1(x) = 0$ by the neural networks of real roots finding and the neural networks of finding multiple real roots, respectively.

Let the learning rate $\eta = 0.01$, and the learning error limit $\varepsilon = 10^{-35}$, the convergent value of weight is $w = 1.00000000$ by RRFNN after 3000 iterations, which is equal to the true

values. However, with the parameters $\eta = 0.02,\ \varepsilon = 10^{-35}$, the convergent values of weights are $w_1 = 1.12006234,\ w_2 = 0.93855331,\ w_3 = 0.94885331,\ w_4 = 0.99842871$ by RFNN after 3000 iterations. It indicated that the RRFNN is more effective than the RFNN.

The curves of converging errors for the neural networks of real roots finding(RFNN) and the neural networks of multiple real roots finding(RRFNN) are shown in Fig.3. From Fig.3, the results showed that the RRFNN is of faster convergent speed than the RFNN.

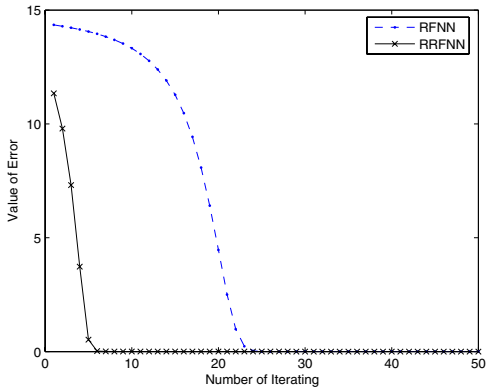

Fig. 3. The learning error curves by two methods in Example 5.1

Example 5.2: [10] Give a 8-order polynomial as

$$f_2(x) = x^8 - 10x^6 + 33x^4 - 40x^2 + 16, \qquad (10)$$

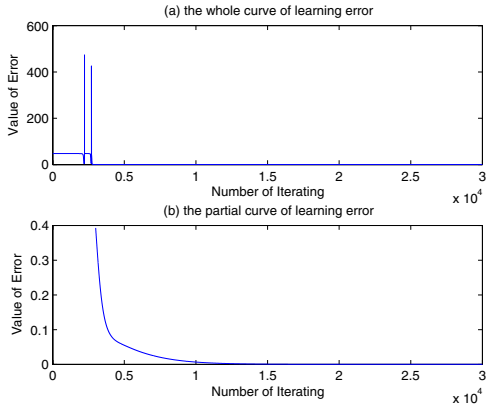

Fig. 4. The learning error curve for Example 5.2

we use the neural networks for finding multiple real roots to solve all of the real roots. According to the complete crimination system, it can be obtained that the distinct real roots are $-1.0, 1.0, -2.0, 2.0$, respectively, and the corresponding multiplicities of which are all 2.

Let the parameters $\eta = 0.00122$, and the learning error limit $\varepsilon = 10^{-10}$, after 30000 iterations, the gained convergent values of weights are $w_1 = -2.00151853,\ w_2 = -0.99971236,\ w_3 = 1.00028649,\ w_4 = 1.99848643$, respectively. That is, the approximative real roots of polynomial (10) can be arrived at by the RRFNN. The corresponding learning error curve is shown in Fig.4, where (a) is the whole curve of learning error, and (b) is the partial amplification of (a), and the learning curves for the four weights are shown in Fig.5.

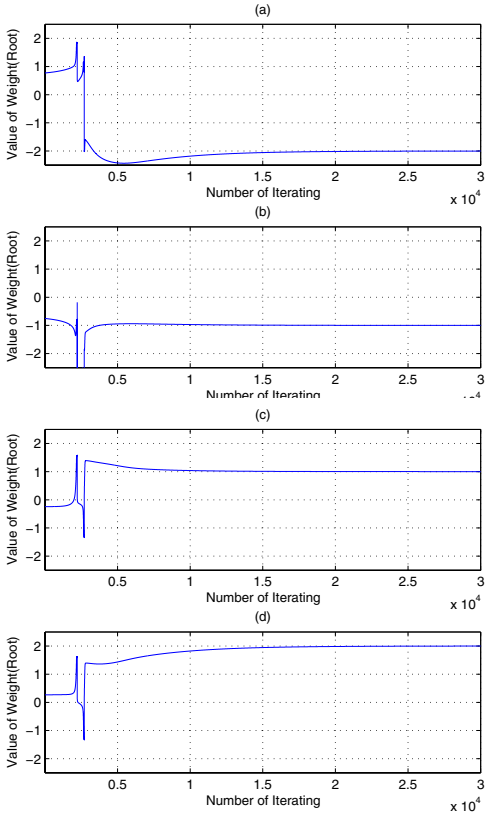

Fig. 5. The learning curves for the four weights of w_1, w_2, w_3, w_4 in Example 5.2

VI. Conclusion

A neural networks model for finding all of the real roots of polynomials with multiple real roots has been derived in this paper, which is the generalized method of the existent neural networks for finding real roots. Results shows that the new method can solve the polynomial with multiple real roots more effectively than the neural networks of real roots finding.

However, both models will encounter the problems of convergent speed and wrong convergence when the degree of polynomial gets greater, which is just the intrinsical defect of the gradient descent method. For the sake of overcoming the difficult mentioned above, constrained learning method had been used in the neural networks of real roots finding [10], [11], which made the learning procedure following the right direction. Therefore, a new clue is illumined for improving the method proposed in this paper, that will be the basis of further generalization of the method presented in this paper.

Acknowledgment

This work was supported partially by Foundation for Youth Scholars of Sichuan Educational Committee of China Under Grant 2006B064, and partially by the National Science

Foundation of Chengdu University of Information Technology Under Grant CSRF200506.

REFERENCES

[1] R. Anthony and R. Philip, *A First Course in Numerical Analysis*, McGraw-Hill International Book Company, 1978.

[2] M. Lang and B. C. Frenzel, "polynomial root finding," *IEEE signal Processing Letters*, Vol. 1, No. 10, pp. 141-143, 1994.

[3] A. Aliphas, S.S. Narayan, and A.M. Peterson, "Finding the zeros of linear phase FIR frequency sampling filters," *IEEE Trans. Acoust. Speech Signal Process*. 31 (1993) 729-734.

[4] L. Hoteit, "FFT-based fast polynomial rooting," *Proc. ICASSP*2000 6 (2000) 3315-3318.

[5] C.E. Schmidt and L.R. Rabiner, "A study of techniques for finding the zeros of linear phase FIR digital filters," *IEEE Trans. Acoust. Speech Signal Process*. 25 (1997) 96-98.

[6] B. Thomas, F. Arani, and B. Honary, "Algorithm for speech model root location," *Electron. Lett*. 33 (1997) 355-356.

[7] L. Yang, X. R. Hou, and Z. B. Zeng, "A complete discrimination system for polynomials," *Sci. China, ser.E*, 1996, 39(6):628.

[8] L.Yang and B. C. Xia, "Explicit criterion to determine the number of positive roots of a polynomial," *MM-Preprints*, 1997, 15:134.

[9] L. Yang, "Recent advances on determining the number of real roots of parametric polynomials," *J. Symb. Comput*. 28: 225-242, 1999.

[10] D.S. Huang and Z. Chi, "A partition-based recursive approach for finding higher-order polynomial roots using constrained learning neural networks", *Sci. China, ser.E*, vol. 33,no. 12, pp. 1115-1124, 2003.

[11] D. S. Huang, H.S.Ip H, Z. R. Chi, and H. S. Wong, "Dilation Methods for Finding Close Roots of Polynomials Based on Constrained Learning Neural Networks". *Physics Letters A*, 2003, 309: 443-451.