Journal of Kerbala University , Vol. 10 No.4 Scientific . 2012

Application of Neural Network For Solving Linear Algebraic Equations

Assistant Lecturer
Noor D.K. Al-Shakarchy*
Email: noor.dhiya@yahoo.com

Assistant Lecturer
Enaam H. Abd*
Email: enaam_hadi2004@yahoo.com

*Computer Science Department
Science College, Karbala University

Abstract

In this paper, we present an neural network approach to solve a set of linear algebraic equations. A Black-Box Neuro- Identifier network has been developed to optimize an convergence until getting the optimal solutions. Black- Box approach called also (input-output description) which is used when no information is available about the system except its input and output. The input-output description of a system gives a mathematical relationship between the input and output of the system. In developing this description, the knowledge of the internal structure of a system may be assumed to be unavailable; the only access to the system is by means of the input and output terminals. Our approach called Black Box Neural Network(BBNN) which provided an optimal solution and a faster way to solve the linear systems; and considered the best with large systems (the number of equations and parameters very large) and conceded the first step to apply this approach with nonlinear algebraic equations which needs difficult computes and long time relatively.

الملخص :

في هذا البحث تم تقديم طريقة جديدة لحل أنظمة المعادلات الجبرية الخطية باستخدام المميز العصبي الاصطناعي والذي يسمى بالصندوق الأسود وذلك لأنه لا يوفر أي معلومات عن النظام غير المدخلات والمخرجات . وتم استخدام النظام المقترح للحصول على الحل الأمثل لقيم المتغيرات وذلك بعد تدريب الشبكة على مجموعة من النماذج للأنظمة الخطية. توفر هذه الطريقة نسبة خطا واطئة جدا و هذا يؤدي الى الحصول على نتائج دقيقة جدا وتقارب كبير جدا للحل الأمثل. وطريقتنا تسمى الصندوق الاسود للشبكات العصبية والتي توفر حلول سريعة نسبيا مقارنة بالأنظمة التقليدية ولذلك ممكن اعتبارها مجدية لحلّ الأنظمة الخطيّة الكبيرة, ذات الأبعاد الكبيرة (عدد المعادلات والمتغيرات كبيرة جدا). وتعتبر الخطوة الأولى لتطبيقها مع أنظمة المعادلات الجبرية اللاخطية و التي تحتاج إلى حسابات معقدة ووقت طويل نسبيا.

1.Introduction:

The problem of solving linear equations is common to many scientific and engineering applications. The fundamental problem of the designer of algorithms, for the solution of linear equations, has been to devise means for reducing the solution time. The basic problem in the solution of linear equations is to determine the unknown solution vector, x, in the equation Ax = b (where A is a real non-singular matrix of order N and b is a known right hand side vector).

There are mainly two classes of methods for solving Ax = b, namely iterative and direct (it is also possible to have methods that have the features of both) . Iterative methods employ a starting solution and converge to a value that is close to the exact solution by iteratively refining the starting solution. However, iterative methods do not guarantee a solution for all systems of linear equations and they are more frequently used for solving large sparse systems of linear equations. On the other hand, direct methods are those in which for a given linear system of equations, exact solutions (within a round-off bound determined by machine precision) can be obtained with a pre-estimated number of arithmetic operations. The central idea of direct methods of solving linear equations is :

1. either to find the inverse of A and then compute $x = A^{-1}b$. The method based on semi rings uses this concept without explicitly finding A^{-1} (but computes $A^{-1}b$ implicitly).
2. or to transform the coefficient matrix, A, into an equivalent triangular or diagonal form in which coupling between the unknowns is reduced. The methods based on Gaussian Elimination, LU decomposition, Givens rotations, and Householder reductions use the concept of converting the coefficient matrix A into an equivalent triangular form and then solve the resulting triangular system of linear equations. On the other hand, the methods based on Gauss-Jordan, Cramer's Rule, And Bidirectional Gaussian Elimination convert the given coefficient matrix A into Diagonal form and then obtain the solution vector x through divisions.

The iterative methods are used to solve the linear and nonlinear algebraic equations . Usually it is hard to solve a large system of highly-linear and nonlinear algebraic equations. Although a lot of priori research has been conducted in this area, we still lack an efficient and reliable algorithm to solve this difficult problem. many numerical methods used in computational mechanics, as demonstrated by Zhu, Zhang and Atluri [1], Atluri and Zhu [2], Atluri [3],Atluri and Shen [4], and Atluri, Liu and Han [5] lead to the solution of a system of linear algebraic equations for a linear problem .

Neural networks are used in many different application domains in order to solve various information processing problems. They have proven to be successful in pattern recognition, pattern classification, function approximation, prediction, optimization, and controlling [6]. The basic processing elements of neural networks are artificial neurons. In order to perform complex tasks, these neurons have to be interconnected in an adequate way by arranging them in an appropriate architecture and setting the weights of their interconnections. The setting of weights is performed during a training phase. In a supervised learning process, input data and target data pairs are presented to the network and the weights of the connections are altered by a learning algorithm as long as for (ideally) all input data, the output values of the network match to the target data [7].
To find an efficient solution to a certain problem, preprocessing of input data is essential to reduce the complexity of the network and the time needed for training [8], [9], [10].

The input-output description of an artificial neural network system gives a mathematical relationship between the input and output of the system. In developing this description, the knowledge of the internal structure of a system may be assumed to be unavailable; the only access to the system is by means of the input and output terminals [12]. Under this assumption, a system may be considered a Black-Box. Clearly what one can do to a black box is to apply inputs and measure their corresponding outputs, and then try to abstract key properties of the system from these input output pairs. An input-output model assumes that the new system output can be predicted by the past inputs and outputs of the system [12, 13].

In this research we present the problem of solving linear algebraic equations as input- output description and used that black- box to find the optimal values of variables in linear system.

2. Systems of Linear Algebraic Equations:

Linear systems of equations are the basis for many, if not most, of the models of phenomena in science and engineering, and their efficient. numerical solution is critical to progress in these areas.

2.1 Linear Algebraic Equations

Many physical systems are described by systems of linear equations that describe the relationships between the variables of these systems. The general shape of the system of linear equations .
Given numbers $a_{11}, \ldots, a_{mn}, b_1, \ldots, b_m$, a non-homogeneous system of m linear equations in n unknowns $x_1, x_2, \ldots, x_n$ is the system :

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \qquad \ldots (1) \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

Constants $a_{11}, \ldots, a_{mn}$ are called the coefficients of system (1). Constants $b_1, \ldots, b_m$ are collectively referenced as the right hand side,right side or RHS. The homogeneous system corresponding to system (1) is obtained by replacing the right side by zero:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= 0 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= 0 \qquad \ldots (2) \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= 0 \end{aligned}$$

An assignment of possible values $x_1, \ldots, x_n$ which simultaneously satisfy all equations in (1) is called a solution of system (1). Solving system (1) refers to the process of finding all possible solutions of (1). The system (1) is called consistent if it has a solution and otherwise it is called inconsistent .

Can be classify linear systems to:

1 - square systems: Are those systems where the number of equations equals the number of variables, i.e m = n.

2 - over-determined systems : The systems are those where the number of equations is greater than any number of variables m > n .

3 - under-determined systems: The systems are those where the number of quations is smaller than any number of variables m < n .

Since most systems of linear equations that appear in practical applications and engineering systems is a kind of square, we will look to study methods of finding solutions to these systems only.

2.2 Methods of Solving Linear Equations Systems

Find methods to divide the solutions to these systems of linear equations for the square to the two main groups:

A) Direct methods: the methods in which those Pettm where access solutions to systems of linear equations specified number of calculations. These methods suitable for finding solutions to relatively simple systems, i.e systems that contain a small number of equations. these methods is :

1- Gaussian Elimination with Backward Substitution .
2- Gauss-Jordon
3- Using Matrix Factorization .
4- Using Gramer rule .

B) Iterative methods: the methods in which these are the repetition of step (or set of steps) until reaching values of variables to find the required accuracy. These methods suitable for finding solutions to systems consisting of a large number of equations. these methods is :

1 - Jacobi method.
2 - Gauss – Seidel method.

3. Artificial Neural Networks

Artificial Neural Networks (ANNs) are simplified models of the central nervous system. They are networks of highly interconnected neural computing elements that have the ability to respond to input stimuli. Among the capabilities of ANN, are their ability to learn adaptively from dynamic environments to establish a generalized solution through approximation of the underlying mapping between input and output. This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

In this research we present a Supervised Learning Black-Box approach which is used when no information is available about the system except its input and output .The architecture of proposed

BBNN System used to solve Linear algebraic equations by depending on input-output description of the main system (Linear algebraic system) described in the following section.

3.1 Architecture of Proposed BBNN System:

Identification of system consists of finding a model relationship and determining the system orders and approximation of the unknown function by neural network model, using a set of input and output data. Neuro- identifiers are basically Multi-Layer Feed- Forward artificial neural networks (MLFF) with an input layer , a single or multiple hidden layer with biases, and a linearest /or nonlinear output layer. The proposed Black-Box system used the MLFF with single hidden layer; as following:

a) Input layer

This layer consist of(n+1) neurons (nodes), where n represent the number of equation in Linear algebraic system each neuron represent the coefficients of the equation plus one neuron represent the results of equations.

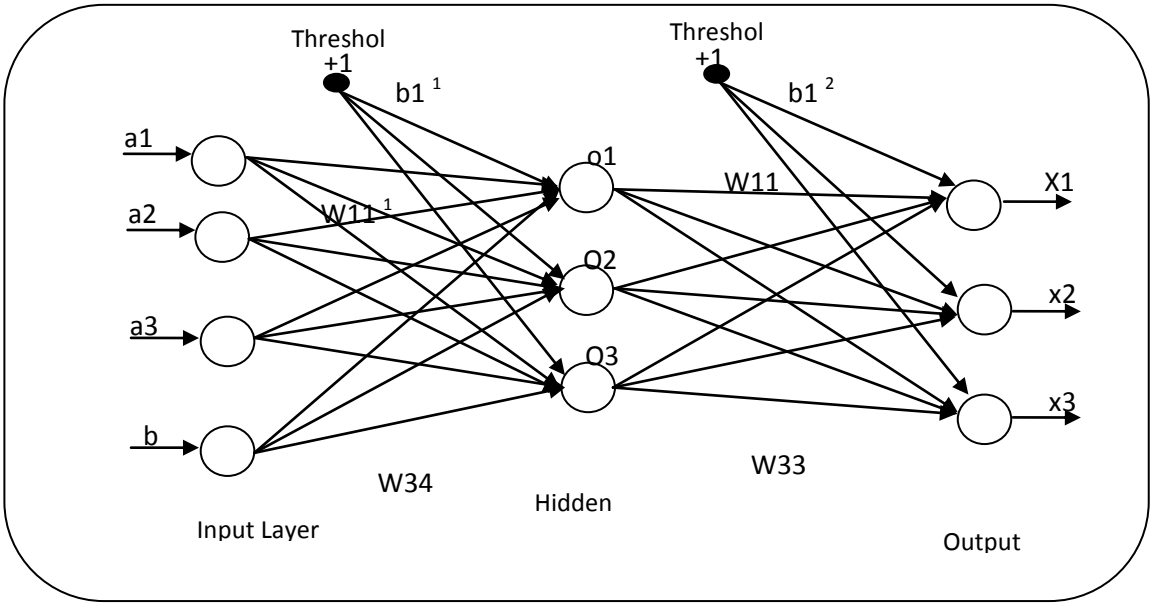
b) Hidden layer

Consist of 48 neuron, each neuron represent one bit of block obtained from Expansion Permutation process.

c) Output layer

This layer consist of n neuron (node), each neuron represent value of one desired variables.

The figure-1 below represent the architecture of BBNN system to 3- equations with 3- variables.



Figure(1) Architecture of BBNN system (3 variables , 3 equation)

3.2 Learning Algorithm of BBNN System

Supervised learning is frequently used to train multi layer feed forward ANN in a lot of applications. Usually, back propagation learning algorithm is used to update the network weights during training in order to improve the network performance. Back propagation (BP) is one of the gradient descent algorithms used to reduce the performance function E through updating the neural network weights by moving them along the negative of the gradient of the performance function. The term back propagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. This method requires that activation functions f are differentiable as the weight update rule is based on the gradient of the error function which is defined in terms of the weights and activation functions. Back Propagation Algorithm is used to learn the proposed neural network by modifying the weights connected between the neurons until the desired value obtained.

The learning algorithm used in this research :

a. determine random values to all weights (W) connected between neurons, the range of these values between (0.1 – 0.3) .

b. select random linear equations then represent the matrices of these equations such as A(the square matrix of coefficients) and B(the matrix of equations results) and X (the matrix of desired variables $x_1, x_2, \ldots, x_n$) .according to example of linear system:

$x_1 + x_2 - x_3 = 0$
$2x_1 - x_2 + x_3 = 6$
$3x_1 + 2x_2 - 4x_3 = -4$

A=

1	2	3
1	-1	2
-1	1	-4

B=

0
6
-4

X=

X1=2
X2=1
X3=3

c. The activation function of the artificial neurons in ANNs implementing the back propagation algorithm is a weighted sum (the sum of the inputs x multiplied by their respective weights w_{ji}):

$$A_j(\bar{x}, \bar{w}) = \sum_{i=0}^{n} x_i w_{ji}$$

We can see that the activation depends only on the inputs and the weights.

d. compute the output values to each layer feed forward direction to obtained final output values from the neural network (O); the most common output function is the sigmoidal function:

$$O_j(\bar{x}, \bar{w}) = \frac{1}{1 + e^{A_j(\bar{x}, \bar{w})}}$$

The sigmoidal function is very close to one for large positive numbers, 0.5 at zero, and very close to zero for large negative numbers. This allows a smooth transition between the low and high output of the neuron (close to zero or close to one). We can see that the output depends only in the activation, which in turn depends on the values of the inputs and their respective weights.

e. Compute the difference between the desired value (D) and the final output value (O)

Now, the goal of the training process is to obtain a desired output when certain inputs are given. Since the error is the difference between the actual and the desired output, the error depends on the weights, and we need to adjust the weights in order to minimize the error. We can define the error function for the output of each neuron:

$$E_j(\bar{x}, \bar{w}, d) = \left(O_j(\bar{x}, \bar{w}) - d_j\right)^2$$

We take the square of the difference between the output and the desired target because it will be always positive, and because it will be greater if the difference is big, and lesser if the difference is small.

f.The error of the network will simply be the sum of the errors of all the neurons in the output layer:

$$E(\bar{x}, \bar{w}, \bar{d}) = \sum_j \left(O_j(\bar{x}, \bar{w}) - d_j\right)^2$$

g. Compute the difference value (δ) to all previous layers using Back Propagation method; The back propagation algorithm now calculates how the error depends on the output, inputs, and weights. After we find this, we can adjust the weights using the method of gradient descendent:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}}$$

h. Updating weights values (W):

$W_{ij}(new) = W_{ij}(old) + \Delta W_{ij}$
$\Delta W_{ij} = \mu \delta O_j$

(h) is used until we find appropriate weights (the error is minimal).

i. Repeats the steps (c, d, e, f, g and h) with another learning data pair.

3.3 Steps of Design BBNN System

The proposed model attempt to calculate the values of variables (x) in Linear system using artificial neural networks; the practical working of this research done using MATLAB and depending on the two main phrase:

Phrase 1: Design System:

this phrase determined the structure of the proposed system as shown in figure-1 above. It's include the number of hidden layers and the number of neurons of each layer with input and output layers to be agreement with the main structure and properties of the linear systems.

Phrase 2:The work flow for the proposed design System: this process done using Matlab implementation by the following steps:

Step 1: Collect Data (preparing data)

The data accrued through the network consist of two kinds:

a. data of neuro identifier (data of Black-Box) this represent input and output data which is the equations (coefficients, results and variables) these presented to the network and obtained from it. Therefore these data is natural data and supplied to the first layer of the network as input data and to the last layer of the network as the output data. The input data captured from the world and transformed to the second layer (hidden layer). The output data received from .hidden layer as the output data of the network

b. training and testing data: the training data combined from the squar matrix of coefficients A and B(the matrix of equations results) as input data provides to first layer and D (the matrix of desired variables $x_1, x_2, \ldots, x_n$) as output data provides to the last layer. (the example of theses matrices illustrate in 3.2.b)

testing data provides the matrix A (square matrix of coefficients) and X (the matrix of values of variables $x_1, x_2, \ldots, x_n$) and the output of the network must be equal to B(the matrix of equations results)

in this research the practical side of this data done by using NN Network/Data Manager screen of MATLAB. This screen used to entered data (the inputs and desired data).

Step 2: Create the Network

In order to use a network we need to first design it as architecture illustrate in 3.3-phase1 , then train it. After this the network is ready for simulations to be performed on it. We change all the parameters on the screen to the values as indicated on the following:

Network name = LINR
Network type = Feed-forward back propagation
Input data = A and B (entered and saved in step 1)
Target data = D (entered and saved in step 1)
Training function= Trainlm
Adaptive learning function= Learngdm
Performance function = Mse
Number of layers = 3
Number of neurons in layer1 = 4
Number of neurons in layer2 = 3
Number of neurons in layer3 = 3
Transfer function = Sigmod

Step 3: Configure the Network

In order to show a network configuration we chose the DES network from data/network manager screen then we can show the configuration of the design network in view page as shown in figure -2 below.

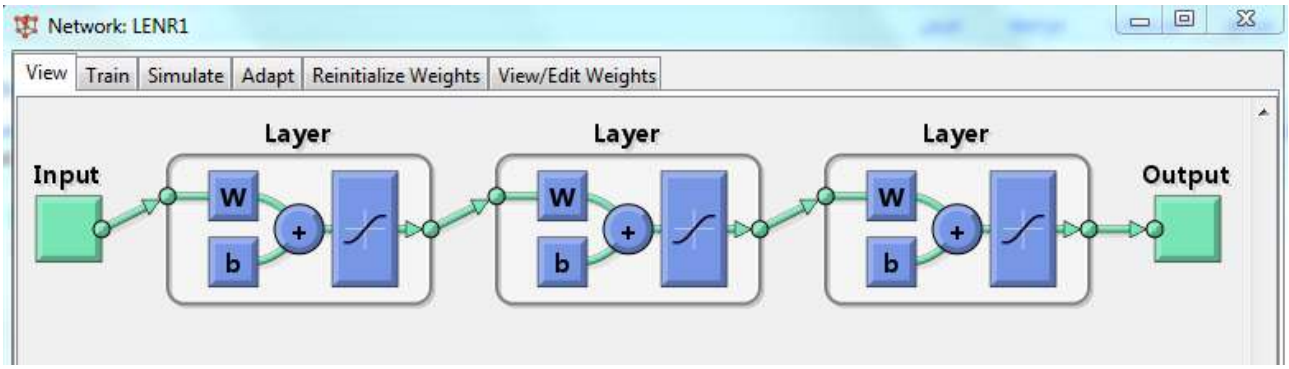


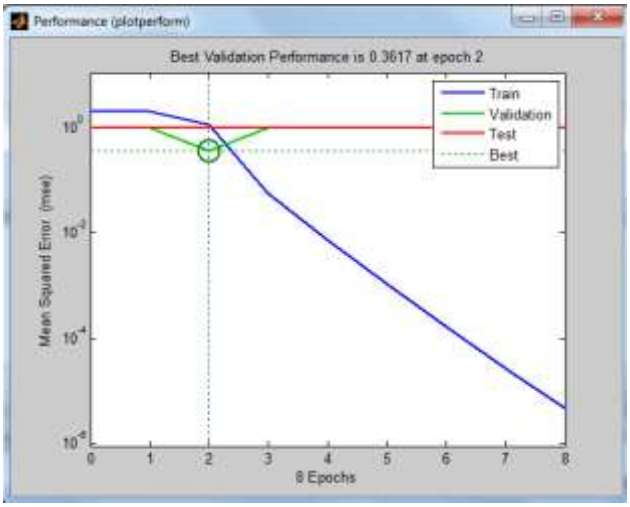
Figure (2) Configuration of BBNN system

Step 4: Initialize the weights and biases

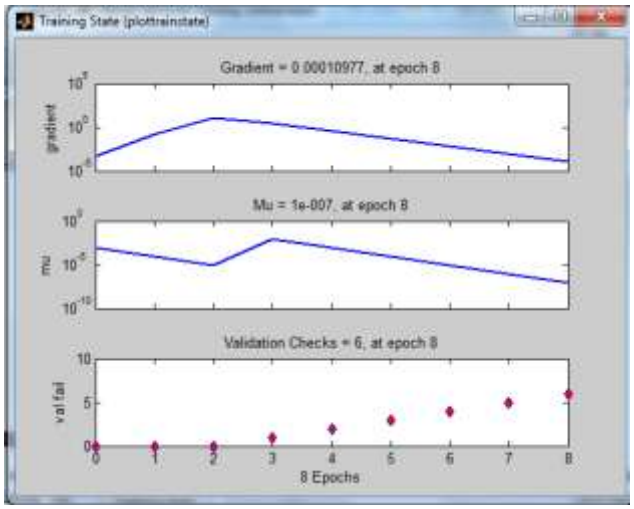
Gives randomly values to each weight matrix and biases for each layer then these weights adaptive and changes during the training process.

Step 5: Network Training

In This step the network training done using feed forward back propagation algorithm, and the training parameters.When the network completed the performance curve which determine the convergence of the training curve with error rate as shown in figure(3) and Training state curves shown in figure(4)



Figure(3)The performance curve



Figure(4) Training state curves

Step 6: Validate the Network

After the network reached to convergence during the training with minimum error rate., we must check the validate work by simulation of this net with testing or simulating data. The input data the matrices of A and B and the network used the saved weights and calculate the output matrix X if this matrix(X) same as the natural matrix then the network is validation .In this case the proposed system become ready to used for solving linear equations.

Step 7: Use the Network

In this step the network used to solving any Linear equations system by inters the matrix of A and B and obtained the matrix of variables X.

4. Experiential Results for Solving Linear Algebra Equations

In this paragraph we compute some examples (find the values of $x_1, x_2, \ldots, x_n$) in traditional methods then calculates the variables ($x_1, x_2, \ldots, x_n$) using proposed Black- Box Neural Network system all values registers in the table below:

Example 1 : Consider the following linear algebraic system :

$x_1 + x_2 - x_3 = 0$
$2x_1 - x_2 + x_3 = 6$
$3x_1 + 2x_2 - 4x_3 = -4$

The matrix of this system is :

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 2 & -1 & 1 & 6 \\ 3 & 2 & -4 & -4 \end{array}\right]$$

by use Gaussian Elimination with Backward Substitution method . $x_1=2$, $x_2=1$, $x_3=3$

Example 2: Consider the following linear algebraic system :

$x_1+2x_2+5x_3=8$
$4x_1+5x_2+x_3=10$
$10x_1+x_2+x_3=12$

ues $x_1^{(0)} = x_2^{(0)} = x_3^{(0)} = 0$

by Gauss-Seidel method we have : $x_1=1$, $x_2=1$, $x_3=1$

Example 3 : Consider the following linear algebraic system :

$10x_1+x_2=12$ ----------(1)
$x_1+10x_2=21$ ----------(2)

The matrix of this system is

$$\left[\begin{array}{cc|c} 10 & 1 & 12 \\ 1 & 10 & 21 \end{array}\right]$$

by use Gaussian Elimination with Backward Substitution method . $x_1=1$, $x_2=2$

And by using Gauss-Seidel method the solution in the following steps :

$10x_1+x_2=12$ ----------(1)
$x_1+10x_2=21$ ----------(2)

And with initial values $x^0{}_1=x^{(0)}{}_2=0$

From (1) we have $x_1=1.2 - 0.1x_2$

From (2) we have $x_2=2.1 - 0.1x_1$

Then iterative equation for this system is

$x_1^{(k+1)}=1.2 - 0.1x_2^{(k)}$
$x_2^{(k+1)} = 2.1 - 0.1x_1^{(k+1)}$

by apply this equations and by use first-order norm ,Then

$x_1^{(1)}=1.2$ & $x_2^{(1)} = 1.98$
$\| x^{(1)} - x^{(0)} \|=3.18$

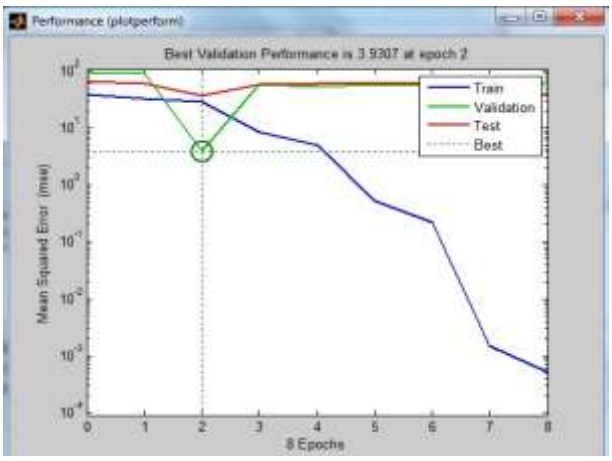
We also apply the proposed method with linear system that have 10 * 10 dimension ,the results of 10*10 system and (3*3 & 2*2) explained in table(1) and table(2) respectively ,the performance curve which determine the convergence of the training curve with error rate as shown in figure(5) and Training state curves shown in figure(6)

Table(1)The experimental result of 10 * 10 system

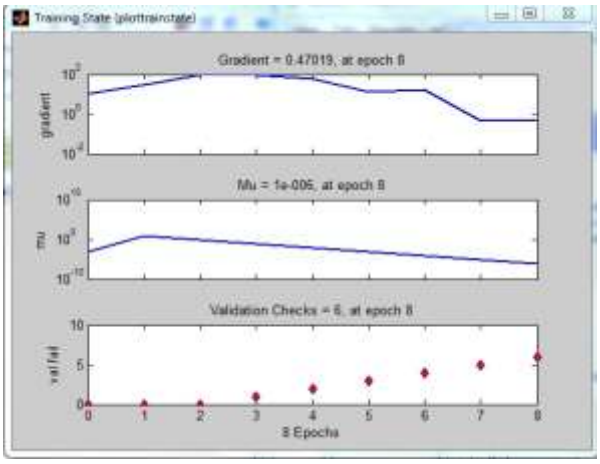
Example 4	No. Equations	No. Variables	Values of variables										Errors Ratio
			X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	
BBNN	10	10	3	-4	6	-6	6	-5	5	0	-5	2	10^{-5}

Table(2) The experiential results of 3 * 3 and 2*2 systems

No. Example	Method Name	No. Equations	No. Variables	Values of variables			Errors Ratio
				X1	X2	X3	
Example1	Direct	3	3	2	1	3	
	BBNN	3	3	2	1	3	10^{-5}
Example2	Iterative	3	3	1	1	1	$2*10^{-4}$
	BBNN	3	3	1	1	1	10^{-5}
Example3	Direct	2	2	1	2	-	-
	Iterative	2	2	1	2	-	$2*10^{-4}$
	BBNN	2	2	1	2	-	10^{-5}



Figure(5) The performance curve of 10* 10 system



Figure(6) Training state curves of 10 *10 system

5.Conclusion

The main purpose of the neural network is to establish the required mapping or at least approximates it to some desired accuracy. System identification using neural networks, is then a problem of finding some weights matrices that represent the transfer function of the unknown system. by concord solving linear algebra system as unknown system and used the final weights matrices that represent the transfer function as system internal representation to the system, we can introduced system used to solve the linear system same as the traditional methods. The main points can be detected from experiment:

- the proposed system (BBNN) used to solve a linear algebra equation system. we can added this system to previous methods such as direct and iterative methods. And the results obtained from this system same as the result obtained from other methods, these results illustrates in table (2) above.
- the proposed system used with large dimension systems with good solution as shown in table (1) above.
- the proposed system provide a good convergence equal to 10^{-5} as shown in figure 3,5 .
- the output values of proposed system to variables have optimal solutions.
- The error propagation in Black-Box system is less. The traditional methods depend on the dependence between the values from each step on previous step. These dependence increase the probability of error happened in any step to other steps. When the Black-Box system provides independent steps.
- The execution time with Black-Box system is very specially when deals with large number of equations.

References

[1]Zhu, T., Zhang, J., and Atluri, S. N., "A meshless local boundary integral equation (LBIE) method for solving nonlinear problems," Computational Mechanics, Vol. 22, pp. 174-186 (1998).

[2]Atluri, S. N. and Zhu, T. L., "A new meshless local Petrov-Galerkin (MLPG) approach in computational mechanics", Computational Mechanics, Vol. 22 , pp. 117-127 (1998)

[3]Atluri, S. N., " Methods of Computer Modeling in Engineering and Sciences" , Tech. Science Press, 1400 pages (2002).

[4]Atluri, S. N. and Shen, S., "The meshless local Petrov-Galerkin (MLPG) method: a simple & less-costly alternative to the finite and boundary element methods," CMES: Computer Modeling in Engineering & Sciences,Vol. 3, pp. 11-51 (2002).

[5]Atluri, S. N., Liu, H. T., and Han, Z. D., "Meshless local Petrov-Galerkin (MLPG) mixed collocation method for elasticity problems," CMES: Computer Modeling in Engineering & Sciences, Vol. 14, pp. 141-152 (2006).

[6]Andreas Scherer, "Neuronale Netze - Grundlagen und Anwendungen", Vieweg Verlag, 1997, ISBN 3-528-05465-4.

[7] Raúl Rojas, "Theorie der neuronalen Netze – Eine systematische Einführung", Springer Lehrbuch, 1996 ISBN 3-540-56353-9.

[8] Howard Demuth and Mark Beale, "Neural Network Toolbox – For Use with MATLAB", The MathWorks, user's guide version 4 edition, 2005. ISBN 3-528-05428-X.

[9] Gesellschaft für Mess- und Automatisierungs-technik, "Computational Intelligence – Neuronale Netze, evolutionäre Algorithmen und Fuzzy Control im industriellen Einsatz", VDI Verlag, 1998, ISBN 3-18-091381-9.

[10]Alberto Sanfeliu, José Francisco Martínez Trinidad, and Jesús Ariel Carrasco Ochoa, "Progress in Pattern Recognition, Image Analysis and Applications", Springer Verlag, 2004, ISBN 3-540-23527-2.

[11]Ljung, Lennart, " System Identification, Theory for the User", Prentic-Hall, 1987.

[12]Pham, Duce Truong & Liu Xing, " Neural Networks for Identification, predication and control", London Berlin Heidelberg New York, 1995.

[13] Patterson, Dan W., "Artificial Neural Networks, theory and Application", Prentic Hall, 1996.

[14]Mahmood Khalel Al-Ubaidy, " Black-Box Attack Using Neuro-Identifier", ph. D. thesis, Univercity of Technology, Baghdad, 2000.

[15]Negia, Leaster S. H. & Jonas Sjoberg, " Some Aspects of Neural Nets and Related Model Structure for Nonlinear System Identification", Internet Explorer, 1998.