Ibn Al-Haitham Journal for Pure and Applied Science مجلة إبن الهيثم للعلوم الصرفة و التطبيقية
No. 3 Vol. 25 Year 2012 2012 السنة 25 المجلد 3 العدد

Designing Feed Forward Neural Network for Solving Linear VolterraIntegro-Differential Equations

L. N. M. Tawfiq, G .H. Ibraheem
Department of Mathematics , College of Education Ibn Al - Haitham , University of Baghdad
Received in: 18April 2011 , Accepted in: 30 May 2011

Abstract

The aim of this paper, is to design multilayer Feed Forward Neural Network(FFNN)to find the approximate solution of the second order linear Volterraintegro-differential equations with boundary conditions. The designer utilized to reduce the computation of solution, computationally attractive, and the applications are demonstrated through illustrative examples.

Keywords :Boundary value problems, VolterraIntegro-differential, Feed Forward Neural Network, Back Propagation training algorithm .

Introduction

We consider the second order linear Volterraintegro-differential equations of the form :

$$y''(x) + p(x)\, y'(x) + q(x)\, y(x) = f(x) + \int_a^x K(x,t)\, y(t)\, dt \quad , \quad x \in [a, b] \ , \qquad (1)$$

with boundary conditions :

$y(a) = y_a$, $y(b) = y_b$,

where K(x, t), f(x), y(x), and p(x), q(x), are analytic functions and y_a , y_b are real constants. y(x) is the solution to be determined. Numerical methods for solution of linear Volterraintegro-differential equations have been studied by the authors[1 − 5]. There have been considerable interest in solving integro-differential equation (1). Theorems which list the conditions for the existence and uniqueness of solutions of such problems are contained in a book by Agarwal [1]. Two point boundary value problem for integro-differential equation of second order is discussed by J. Morchalo in [2]. Also, J. Morchalo [3] studied two point boundary value problem for integro-differential equation of higher order. A reliable algorithm for solving boundary value problems for higher-order integro-differential equation has been proposed by A. M-Wazwaz [4]. The aim of mentioned paper is to present an efficient analytical and numerical procedure for solving boundary value problems for integro-differential equations. E. Babolian et al [5], applied operational matrices of piecewise constant orthogonal functions for solving Volterra integral and integro-differential equations. They first obtained Laplace transform of the problem and then found numerical inversion of Laplace transform by operational matrices. Sinc methods have increasing been recognized as powerful tools for attacking problems in applied physics and engineering. The book [6] provide excellent overviews of methods based on Sinc functions for solving ordinary and partial differential equations and integro-differential equations. In [7], the Sinc collocation procedures for the eigenvalue problems are presented. J. Rashidinia, M. Zarebnia in [8, 9], used Sinc methods for numerical solutions of integral equations.

In this paper, we design a multilayer Feed Forward Neural Network(FFNN) to approximate the solution of the equation (1) .

What are Artificial Neural Networks? [10]

An Artificial Neural Network (Ann) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. Ann's, like people, learn by example. An Ann is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of Ann's as well. That is Artificial Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This brain modeling also promises a less technical way to develop machine solutions. This new approach to computing also provides a more graceful degradation during system overload than its more traditional counterparts. These biologically inspired methods of computing are thought to be the next major advancement in the computing industry. Even simple animal brains are capable of functions that are currently impossible for computers. Computers do rote things well, like keeping ledgers or performing complex math. But computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into actions of the future.

Multilayer Feed Forward Neural Network [11]

In a layeredneural network the neurons are organized in the form of layers. We have at least two layers: an *input* and an *output layer*. The layers between the input and the output layer (if any) are called *hidden layers*, whose computation nodes are correspondingly called *hidden neurons* or *hidden units*. The source nodes in the input layer of the network supply respective elements of the activation pattern (input vector), which constitute the input signals applied to the neurons (computation nodes) in the second layer (i.e., the first hidden layer). The output signals of the second layer are used as inputs to the third layer, and so on for the rest of the network. A layer of nodes projects onto the next layer of neurons (computation nodes), but not vice versa. In other words, this network is strictly a *feed forward* or *acyclic* type. The neurons in each layer of the network have as their inputs the output signals of the preceding layer only. The set of output signals of the neurons in the output (final) layer of the network constitutes the overall response of the network to the activation pattern supplied by the source nodes in the input (first) layer.

The function of hidden neurons is to intervene between the external input and the network output in some useful manner. By adding one or more hidden layers, the network is enabled to extract high-order statistics. In a rather loose sense the network acquires a global perspective despite its local connectivity due the extra synaptic connections and the extra dimension of neural interactions. The Ann is said to be fully connected in the sense that every node in each layer of the network is connected to every other node in the adjacent forward layer (otherwise the network is called partially connected

Description of the Method

In this section we will illustrate how our approach can be used to find the approximate solution of the equation(1), let y(x)denotes the solution to be computed, $y_t(x,\ p)$ denotes a trial solution with adjustable parameters p , and $y_a(x)$ denotes analytic solution .

In the our proposed approach, the trial solution y_t employs a FFNN and the parameters p correspond to the weights and biases of the neural architecture. We choose a form for the trial

function $y_t(x)$ such that it satisfies the BC's. This is achieved by writing it as a sum of two terms :$y_t(x_i, p) = A(x) + F(x, N(x, p))$ (2)
where N(x, p) is a single-output FFNN with parameters p and one input unit fed with the input vector x .The term A(x) contains no adjustable parameters and satisfies the boundary conditions. The second term F is constructed so as not to contribute to the BC's, since $y_t(x)$ satisfy them. This term can be formed by using a FFNN whose weights and biases are to be adjusted in order to deal with the minimization problem .

Computation of the Gradient

An efficient minimization of (2) can be considered as a procedure of training the FFNN, where the error corresponding to each input vector x_i is the value E (x_i) which has to forced near zero. Computation of this error value involves not only the FFNN output but also the derivatives of the output with respect to any of its inputs. Therefore, in computing the gradient of the error with respect to the network weights. Consider a multi layer FFNN with one input unit, one hidden layer with H sigmoid units and a linear output unit .
For a given input vector $X = (x_1, x_2, \dots, x_n)$ the output of the FFNN is :

$$N = \sum_{i=1}^{H} v_i \sigma(z_i), \text{ where } z_i = \sum_{j=1}^{n} w_{ij} x_j + b_i \qquad (3)$$

w_{ij} denotes the weight connecting the input unit j to the hidden unit i,
v_i denotes the weight connecting the hidden unit i to the out put unit,
b_i denotes the bias of hidden unit i, and
σ (z) is the sigmoid transfer function (logsig) .
The gradient of FFNN, with respect to the parameters of the FFNN can be easily obtained as :

$$\frac{\partial N}{\partial v_i} = \sigma(z_i) \qquad (4)$$

$$\frac{\partial N}{\partial b_i} = v_i \sigma'(z_i) \qquad (5)$$

$$\frac{\partial N}{\partial w_{ij}} = v_i \sigma'(z_i) x_j \qquad (6)$$

Once the derivative of the error with respect to the network parameters has been defined, then it is a straight forward to employ any minimization technique and we will use CG method. It must also be noted, the batch mode of weight updates may be employed .

Conjugate Gradient Algorithms [12]

All conjugate-gradient (CG) algorithms start bysearching in the steepest descent direction (negative of the gradient) for the first iteration, we set $\rho_0 = -g_0$ where g_0 is the gradient of the error with respect to the weights, bias on the first iteration. A line search is then performed to determine the optimal distance to move along the current search direction : $w_{k+1} = w_k + \alpha_k \rho_k$.
Then the next search direction is determined so that it is conjugate to previous search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search direction:
$\rho_k = -g_k + \beta_k \rho_{k-1}$

The various versions of conjugate gradient are distinguished by the manner in which the constantβ_k is computed. In this paper ,we use β_k for the Polak-Ribiere update, the constantβ_k is computed from : $\beta_{PR}=\dfrac{\Delta g_{k-1}^{T} g_k}{g_{k-1}^{T} g_{k-1}}$

Illustration of the Method

To illustrate the method, we will consider the equation (1) ,where $x \in [0, 1]$ and the BC $y(0) = A$ and $y(1) = B$. A trial solution can be written as :

$y_t(x) = A(1- x) + B x + x(1- x) N(x, p)$ (7)

where N(x, p) is the output of FFNN with one input unit for x and weights p .

Note that ***$y_t(x)$ satisfies the BC by construction. The error quantity to be minimizedis given by : $E[p]= \sum_i(y_a(x_i) - y_t(x_i))^2$ (8)***

where the x_i's are points in [0, 1]. It is straightforward to compute the gradient of the error with respect to the parameters p using (4) – (6). The same holds for all subsequent model problems.

Numerical examples

In this section we report some numerical result and the solution of two model problems. In all cases we used a multi layer FFNN having one hidden layer with 5 hidden units (neurons) and one linear output unit. The sigmoid activation of each hidden unit is logsig that is :$\sigma(x) = \dfrac{1}{1+e^{-x}}$

For each test problem the analytic solution $y_a(x)$ was known in advance. Therefore we test the accuracy of the obtained solutions by computing the deviation :

$\Delta y(x) = | y_t(x) - y_a(x) |$.

In order to illustrate the characteristics of the solutions provided by the neural network method, we provide figures displaying the mean square error of FFNN for the train ,validation, test ,best and goal at the few points (training points) that were used for training and at many other points (test points) of the domain of each equation. To perform the error minimization we use CG method with best search direction,β_{PR} .

Example 1

We consider the Volterraintegro-differential equation :

$$y'' - \frac{1}{x} y' + y = f(x) + \int_0^x K(x,t)\, y(t)\, dt \quad , \ y(0) = 0 \ , y(1) = 0 ,$$

with exact solution $y_a(x) = x(1 - x^2)$.

Where $K(x, t) = e^t \dfrac{\sin(t)}{x+t}$, $f(x) = - x^3 - x(2 + e^x \dfrac{1}{\sqrt{2}} \sin(x - \Pi/4)) - \dfrac{1}{x}(x \cos x - \sin x)$.

According to (7) the trial neural form of the solution is taken to be: $y_t(x) =x(1-x) N(x, p)$

The FFNN trained using a grid of ten equidistant points in [0, 1]. Figure(2) displays the mean square error of FFNN for the train ,validation, test ,best and goal at the grid points in 1745 epoch and in figure (3) displays the performance of FFNN at many other points in [0, 1]. It is clear that the solution is of high accuracy, although training was performed using a small number of points. Moreover, the extrapolation error remains low for points near the equation domain. The numerical results of FFNN with search direction, β_{PR}, introduced in table (1) .

This example solved in [13] using sinc collocation method for different number of sinc grid points N and give the maximum of absolute errors , that is , $\| E_s \| = 5.92003 \times 10^{-3}$,its clear that the results of FFNN is more accurate than given in [13] .

Example 2

Consider the following Volterraintegro-differential equation with the exact

Solution : $y_a(x) = e^x$

$y'' - (1/(1-x))y' = f(x) + \int_a^x K(x,t)\, y(t)\, dt$, $y(0) = 1$, $y(1) = e$

where
$f(x) = e^x\{1 - (1/(1-x))\} - \{7(-2 + 2e^x + x^2)\cos x\}/ 16(e + x)$

$K(x, t) = 7/8 \{(e^t + t)/(x + e)\}\cos x$.
According to (7) the trial neural form of the solution is taken to be: $y_t(x) = (1 - x) + e\, x + x(1 - x)N(x, p)$.
Again, as before, we use a grid of ten equidistant points in [0, 1]. Figure(4) displays the mean square error of FFNN for the train ,validation, test ,best and goal at the grid points in 1390 epoch and in figure (5) displays the performance of FFNN at many other points in [0, 1]. It is clear that the solution is of high accuracy, although training was performed using a small number of points. Moreover, the extrapolation error remains low for points near the equation domain. The numerical results of FFNN with search direction, β_{PR}, introduced in table (2) .

This example solved in [13] using sinc collocation method for different number of sinc grid points N and give the maximum of absolute errors , that is, $\| E_s \| = 2.91837 \times 10^{-3}$, its clear that the results of FFNN is more accurate than given in [13] .

Conclusion

The Feed Forward Neural Network is used to solve the second order linear Volterraintegro-differential equations with boundary conditions. The numerical examples show that the accuracy improve with increasing the number of epoch when we train the network and the results of testing network is more accurate than training network .

References

1.Agarwal, R. P. (1983) ,Boundary value problems for higher order integro-differential equations, Nonlinear Analysis, Theory, Meth. Appl., 7: 259-270.
2.Morchalo, J. (1975) , On two point boundary value problem for integro-differential equation of second order, Fasc. Math., 9: 51-56 .
3.Morchalo, J. (1975) ,On two point boundary value problem for integro-differential equation of higher order, Fasc. Math., 9:77-96 .
4.Wazwaz,A. (2001) ,A reliable algorithm for solving boundary value problems for higher-order integro-differential equations, Appl. Math. Comp., 118: 327-342 .
5.Babolian, E. and SalimiShamloo , A. (2008) , Numerical solution of Volterra integral and integro-differential equations of convolution type by using operational matrices of piecewise constant orthogonal functions, J. comput. Appl. Math., 214: 495-508 .
6.Stenger, F. (1993), Numerical Methods Based on Sinc and Analytic Functions, Springer-Verlag, New York .
7.Eggert, N.;Jarratt, M. and Lund, J. (1987), Sinc function computations of the eigenvalues of Sturm-Liouville problems, J. Comput. Phys., 69: 209-229.
8.Rashidinia, J. Zarebnia,M. (2007) ,Solution of a Volterra integral equation by the Sinc-Collocation method, J. Comput. Appl. Math., 206: 801-813.
9.Rashidinia, J. and Zarebnia,M. (2007) ,Convergence of approximate solution of system of Fredholm integral equations, J. Math. Anal. Appl., 333: 1216-1227.
10.Stergiou ,C. andSiganos, D. (2008), NEURAL NETWORKS ,IEEE Neural Networks Council
11.Hristev, R. M. (1998) ," The ANN Book ", Edition 1.

12.Tawfiq,L.N.M. and Naoum, R.S. (2005) ,"On Training of Artificial Neural Networks", AL-FathJornal, No 23.

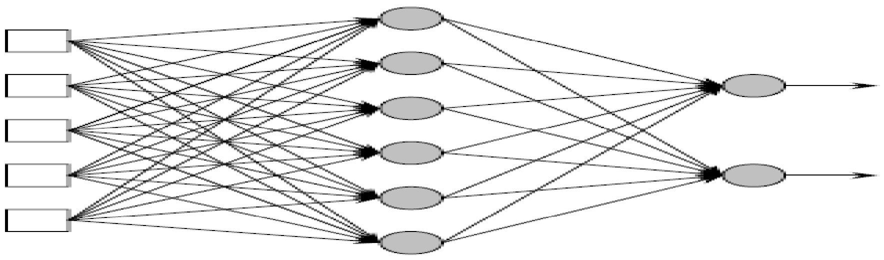
13.Zarebnia,M.andNikpour, Z. (2010), " Design Feed Forward Neural Network for Solving Linear VolterraIntegro-Differential Equations" , International Journal of Applied Mathematics and Computation ,2(1) : 1–10.

Table (1): The numerical results of FFNN with search direction,β_{PR} for example (1)

Input X_i	Exact Solution y_a	Output of FFNN $y_t(x)$ when we use CG with β_{PR}	Deviation $\Delta y(x) = \|y_t(x) - y_a(x)\|$,
1.0	0	0	0
0.0	0	0	0
0.1	0.099	0.098891	0.000109
0.2	0.192	0.192098	0.000098
0.3	0.273	0.272895	0.000105
0.4	0.336	0.33602	0.00002
0.5	0.375	0.37515	0.00015
0.6	0.384	0.38406	0.00006
0.7	0.357	0.3571	0.0001
0.8	0.288	0.288021	0.000021
0.9	0.171	0.17103	0.00003

Table (2): The numerical results of FFNN with search direction,β_{PR} for example (2)

Input	***Exact Solution***	**Output of FFNN$y_t(x)$ when we use CG with**	***Deviation$\Delta y(x) = \|y_t(x) - y_a(x)\|$,***
x	$y_a(x)$	β_{PR}	β_{PR}
1.0	2.718281828459046	2.718281828459017	0.000000000000029
0.0	1.000000000000000	1.000000000000000	0.000000000000000
0.1	1.105170918075648	1.105170917035035	0.000000001040613
0.2	1.221402758160170	1.221402758150140	0.000000000010030
0.3	1.349858807576003	1.349858807574302	0.000000000001701
0.4	1.491824697641270	1.491824697547110	0.000000000904160
0.5	1.648721270700128	1.648721270502115	0.000000000198013
0.6	1.822118800390509	1.822118800361503	0.000000000029006
0.7	2.013752707470477	2.013752706680191	0.000000000790286
0.8	2.225540928492468	2.225540925492324	0.000000003000144
0.9	2.459603111156950	2.459603111052913	0.000000000104037



Fig(1): A fully connected feed-forward neural network with one hidden layer

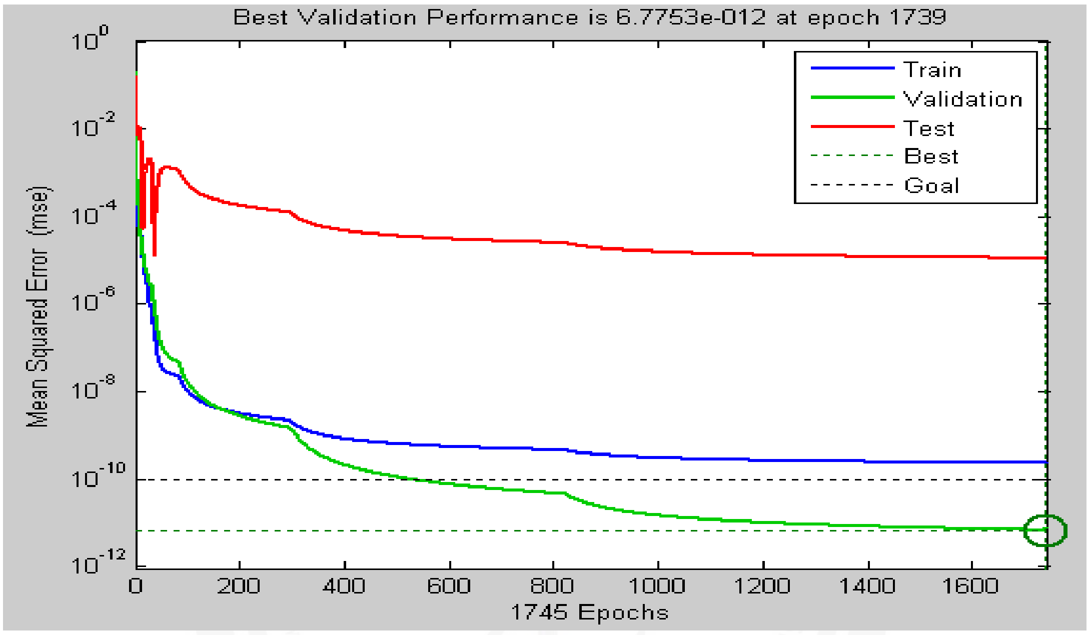


Fig.(2) : the mean square error of FFNN for example (1)

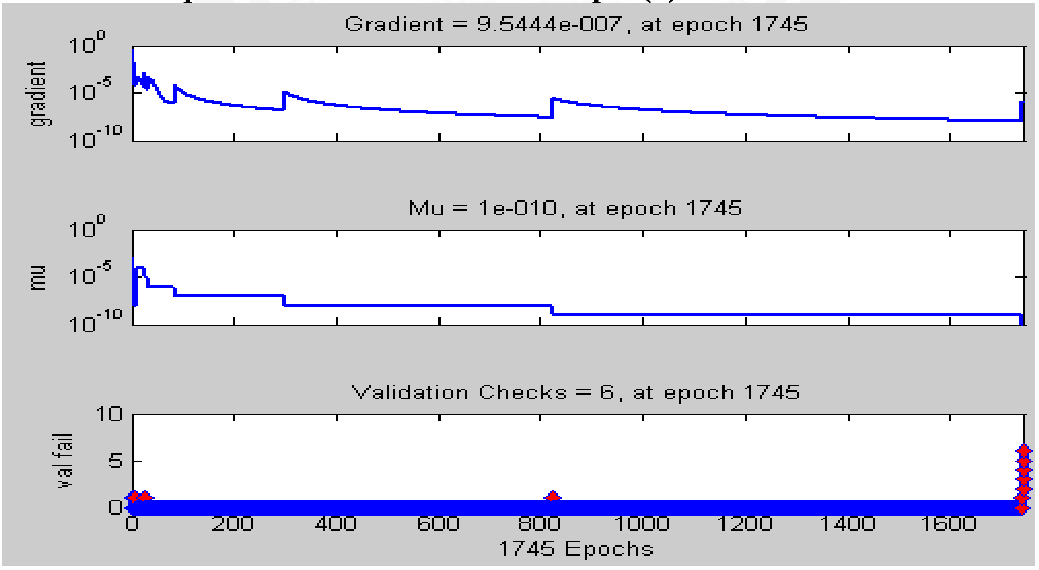


Fig.(3) the performance of FFNN for example (1)

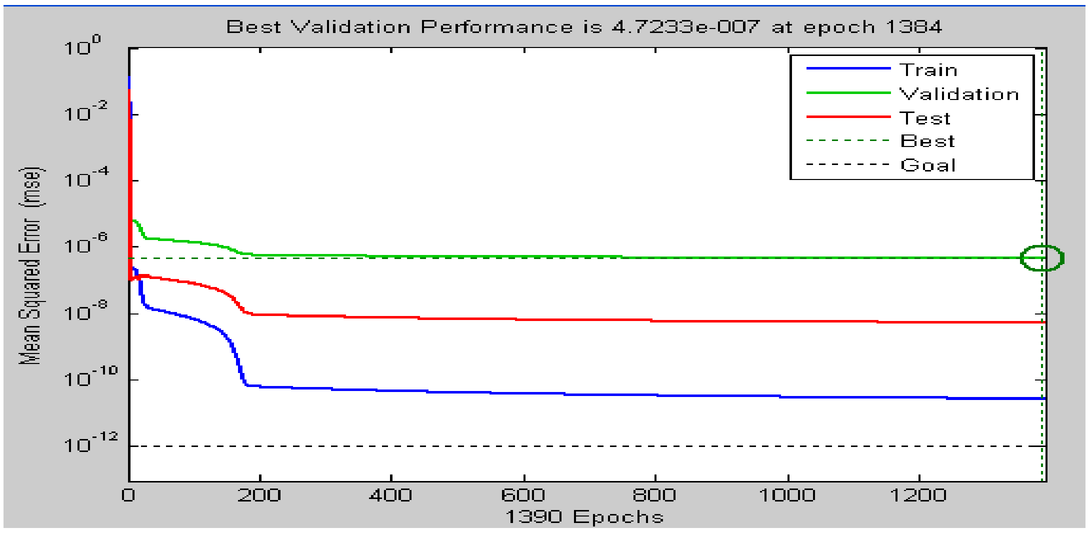


Fig.(4) : the mean square error of FFNN for example (2)

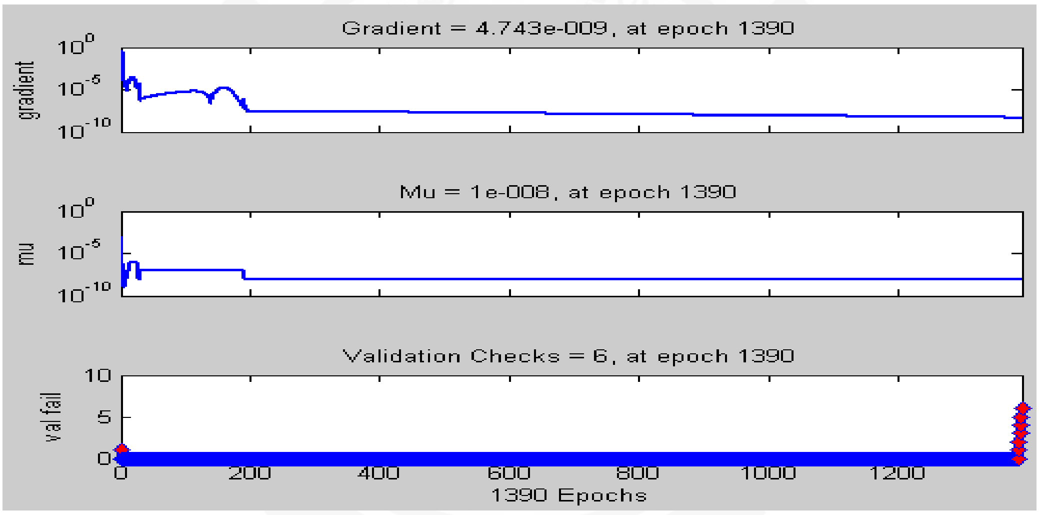


Fig.(5) : the performance of FFNN for example (2)

تصميم شبكة عصبيـة ذي تغذية تقدمية لحل معادلات فولتيرا التفاضليـة- التكاملية الخطية

لمى ناجي محمد توفيق و غادة حسن إبراهيم
قسم الرياضيات، كلية التربية ابن الهيثم ، جامعة بغداد

استلم البحث في: 18نيسان 2011 قبل البحث في: 30 ايار 2011

الخلاصة

الهدف من البحث هو تصميم شبكة عصبية ذي تغذية تقدمية متعددة الطبقات لايجاد الحل التقريبي لمعادلة فولتيرا التفاضلية التكاملية الخطية من الرتبة الثانية ذي الشروط الحدودية. استعمال هذه الطريقة ساعد على تقليل الحسابات في أثناء الحل بشكل جذاب . وضحنا كيفية استعمال الخوارزميات من خلال الأمثلةأيضا.

الكلمـات المفتاحيــة: مسـائل القيم الحدوديـة، معادلـة فولتيرا التفاضـلية-التكامليـة ، شـبكة عصـبية ذو تغذيـة تقدميـة، خوارزمية تدريب للتوليد الخلفي.