Applications and Applied Mathematics: An International Journal (AAM)

Volume 6 | Issue 1 Article 15

6-2011

Shooting Neural Networks Algorithm for Solving Boundary Value Problems in ODEs

Kais I. Ibraheem
Mosul University

Bashir M. Khalaf
Mosul University

Available at
http://pvamu.edu/aam
Appl. Appl. Math.
ISSN: 1932-9466

Applications and Applied Mathematics:
An International Journal
(AAM)

Vol. 6, Issue 1 (June 2011) pp. 187 – 200
(Previously, Vol. 6, Issue 11, pp. 1927 – 1941)

Shooting Neural Networks Algorithm for Solving Boundary Value Problems in ODEs

Kais Ismail Ibraheem and Bashir M. Khalaf

Department of Computer science
Mosul University
Mosul, Iraq
kaisismail@yahoo.com

Received: July 12, 2010; Accepted: March 4, 2011

Abstract

The objective of this paper is to use Neural Networks for solving boundary value problems (BVPs) in Ordinary Differential Equations (ODEs). The Neural networks use the principle of Back propagation. Five examples are considered to show effectiveness of using the shooting techniques and neural network for solving the BVPs in ODEs. The convergence properties of the technique, which depend on the convergence of the integration technique and accuracy of the interpolation technique are considered.

Keywords: Shooting, Neural Networks, Back propagation, BVPs, ODEs

MSC (2010) No.: 65L10, 30E25, 68T27, 92B20.

1. Introduction

Some well-known numerical method for solving BVPs in ODEs include i) the Finite different method ii) the shooting method iii) the collocation method. The Finite difference method consists of dividing the given interval of the independent variable by node and then approximating the differential equation by a given set of finite difference formulas at each node that will produce a set of algebraic equations mostly non-linear, which may be solved by Newton iteration or one of its alternatives. [For more detail see Khalaf (1988) and Kais and Abudu (2001, March 5)]. To get accurate results for these methods, we have to increase the number of nodes which will produce a greater number of algebraic equations. This also increases the complexity of the solution and takes a lot of computer time. Mostly, the iteration processes at the nodes create a noisy set of data that is accumulated by the iteration processes and tends to render the solution meaningless.

The Shooting method consists of dividing the integration interval into subintervals. At the beginning of each subinterval, values are estimated for the given dependent variables then and the ODEs of the problem integrated using the estimated values and then at the end of each subinterval the corresponding estimated and the integrated values of the corresponding dependent variables are matched, i.e., at the end of each subinterval matching functions are defined. The estimated values are then readjusted by Newton iteration or one of its alternatives.

The problem with these methods is that the estimated values need to be very close to the real solution, otherwise the iteration processes will diverge. [For more detail see, for example, Keller (1968), Barto & Sutton (1981), Khalaf (1990)]. The collocation methods [see Mattheij et al. (1988) and Khalaf (1997)] based on approximating the solution of the ODE by a linear combination of a set of independent simple functions. The coefficients of the combination may then be estimated (using the boundary conditions BCs and substituting the differentiation of the approximate solution at the given nodes of the iterative methods). Hence the iteration will mostly diverge if there are noises or error in the combination coefficients; so we seek here to develop a new method for solving BVDs in ODEs that uses integration and interpolation instead of iteration processes. The results offer a significant contribution to the field by successfully developing Neural Network methods for solving BVPs.

2. Artificial Intelligence

Artificial intelligence (AI) is a broad field that includes topics such as expert systems [Shu-Hsien (2005)] fuzzy logic [Turksen (1997)] artificial neural networks [Meireles and Simoes (2003)], evolutionary computing [Dolin and Merelo (2002)] and data mining [Zaslavsky and Krishnaswamy (2005)]. Each of these activities strives to enable a computer to complete tasks which normally require human intelligence. The classic definition of AI from the Turing test [Turing (1950)] requires simply that a human observer fail to distinguish between artificial and human responses to the same task. For instance, passengers would be hard-pressed to differentiate between human pilots and autopilots in a commercial airliner. A so-called intelligent system as defined by the Turing test is relatively simple to develop as long as the operating domain is kept small; the human pilot must take control when the operating conditions stray outside the bounds that the artificial system was developed to handle such as during inclement weather.

The Turing test requires only that an artificial system behave like a human, and not necessarily think like a human. Many believe that the latter must be true as well, for a system to truly be intelligent. Artificial neural networks (ANNs) are based on biological neural networks composed of interconnected neurons which communicate with each other by transmitting binary electrical pulses. Biological neural networks contain billions of such neurons which act in parallel with an individual switching speed of approximately a millisecond, where a larger brain generally signifies higher intelligence.

ANNs are based on this model but have at most hundreds of neurons operating with continuous signals in series at speeds in the nanosecond range, where the principle of Occam's razor dictate that the smallest network capable of solving the problem provides the best solution. Critics of ANNs maintain that they hardly resemble their biological counterparts: not only they do not think like a human, but they are black boxes whose inner workings are impossible to interpret even when they do produce reasonable responses. ANNs were conceived before significant computing power was available [Khalaf (1990) and Khanna and Noback (1963)] but their research did not begin to blossom [Kuznetsova et al. (1980) and Barto (1981)], until the advent of the personal computer after which they have been applied to vast numbers of engineering problems [McFall (2006)].

3. The New Method

3.1. Shooting Technique

Let us consider a two point BVP (this will not affect the generality of the method):

$$\frac{d^2y}{dx^2} = f(x, y, y'), \quad y(a) = \alpha\ , \quad \mathrm{y}(b) = \beta, \quad x \in (a, b). \tag{1}$$

That above problem can be reduced to the following system:

$$\begin{aligned} \frac{dy_1}{dx} &= y_2 \\ \frac{dy_2}{dx} &= f(x, y_1, y'), \qquad \mathrm{y}_1(a) = \alpha, \quad y_1(b) = \beta\ . \end{aligned} \tag{2}$$

To integrate system (2) in the interval (a, b) we need a value of y_2 (a) which is unknown. To get this value, we proceed as follows:

Process 1: Estimate a value *S* for $y_2(a)$ and integrate the system (2) in the interval (*a*, *b*), we get $y_1(b) = m_0$.

Process 2: Estimate another different value s_1 for y_2 (a) and integrate the system (2) in the interval (a, b), we get y_1 (b) $=m_1$ and so on.

.

.

.

Process *n*: Estimate another different value s_n for y_2 (a) and integrate the system (2) in the interval (a, b), we get $y_1(b) = m_n$.

Hence, we get the following table of data:

Table 1-a.

$s_i = y_2(a)$	s_0	s_1	s_2	$\dots$	s_n
$m_i = y_1(b)$	m_0	m_1	m_2	$\dots$	m_n

We can write the above table as:

Table 1-b.

P	m_0	m_1	m_2	$\dots$	m_n
T	s_0	s_1	s_2	$\dots$	s_n

Then, by using neural networks, we can find the value of $y_2(a)$ corresponding to $y_1(b) = \beta$ by.

3.2. Neural Networks Technique

Step 0. Initialize weights. (Set to small random values).

Step 1. While stopping condition is false, do Steps 2-9.

Step 2. For each training pair, do Steps 3-8.

Feed Forward

Step 3. Each input unit (m_i, i=1,…, n) receives. Input signal x_i and broadcasts this signal to all units in the layer above (the hidden units).

Step 4. Each hidden unit (z_j, j=1,…,r) sums its weighted input signals, $z-in_j = v_{oj} + \sum_{i=1}^{n} m_j v_{ij}$ and applies its activation function to compute its output signals $z_j = f(z-in_j)$ and sends this signal to all units in the layer above (output units).

Step 5. Each output unit (p_h. k = 1, …, r) sums $y-in_k = w_{oj} + \sum_{i=1}^{n} z_j w_{jk}$ and applies its activation function to compute its output signal $y_k = f(y-in_k)$.

Back propagation of error:

Step 6. Each output unit $(y_k,\ k=1,\dots,m)$ receives a target pattern corresponding to the input training pattern, computes its error information term $\delta_k = (t_k - y_k) f'(y-in_k)$. Calculates its weight correction term (used to update w_{jk} later) $\Delta w_{jk} = \alpha \delta_k z_j$, and sends S_h to units in the layer oclow. Each hidden unit $(z_k\,,\ j=1,...,p)$ sums its delta inputs (from units in the layer) $\Delta w_{ak}=\alpha\delta_k$ and sends S$_h$ to units in the layer oclow.

Step 7. Each hidden unit $(z_j\,,\ j=1,......., p)$ sums its Delta inputs (from units in the layer above) $\delta - in_j = \sum_{k=1}^{m}\delta_k w_{jk}$ multiplies by the derivative of its activation function to calculate its error information term. Calculates its weight correction term (used to updated later) and calculators its bias correction term (used to update later) $\delta_j = \delta - in_j f'(z - in_j)$. Update weights and biases: $\Delta v_{ij} = \alpha\delta_j x_i$ and $\Delta v_{0j} = \alpha\delta_j$.

Step 8. Each output unit $(z_k\,,\ k=1,...,p)$ updates its bias and weights $(j=0,...,p)$: $w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$. Each hidden unit $(z_j\,,\ j=1,...,r)$ updates its bias and weights $(i=0,...,n)$: $v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$.

Step 9. Test stopping condition. [Fausett (1994)].

4. Convergence Properties of the Technique

The convergence properties of integration and interpolation technique are well studied by Khalaf (2008). Here, we summarize their study: Convergence of integration algorithm accuracy of the new technique.

5. Testing the New Method

In this section of the research, five examples are considered to test the new algorithm for the determination of the results and the Magnitude of error. The network's input ranges from [0 to 5]. The first layer has neurons; the second layer has one Purelin-neuron.

Example 1. (Linear Problem)

Consider the following linear BVPs:

$$\frac{d^2y}{dx^2} = -\frac{dy}{dx} + 2y\,,\qquad y(0)=1,\quad y(1)=e.$$

Exact solution ($y = e^x$).

We can reduce the problem to the following form: to integrate system by the new method, we give the following value for $y_2(0) = -1,0,2,3$ and we integrate the system by Runge-Kutta method we get the following table:

Table 2-a.

$y_2(a)$	-1	0	1	2.	3
$y_1(b)$	0.9963	1.8573	2.7183	3.5793	4.4402

We can write the above table as:

Table 2-b.

P	0.9963	1.8573	2.7183	3.5793	4.4402
T	-1	0	1	2.	3

Here is a problem consisting of inputs *P* and targets *T* that we would like to solve with neural network.

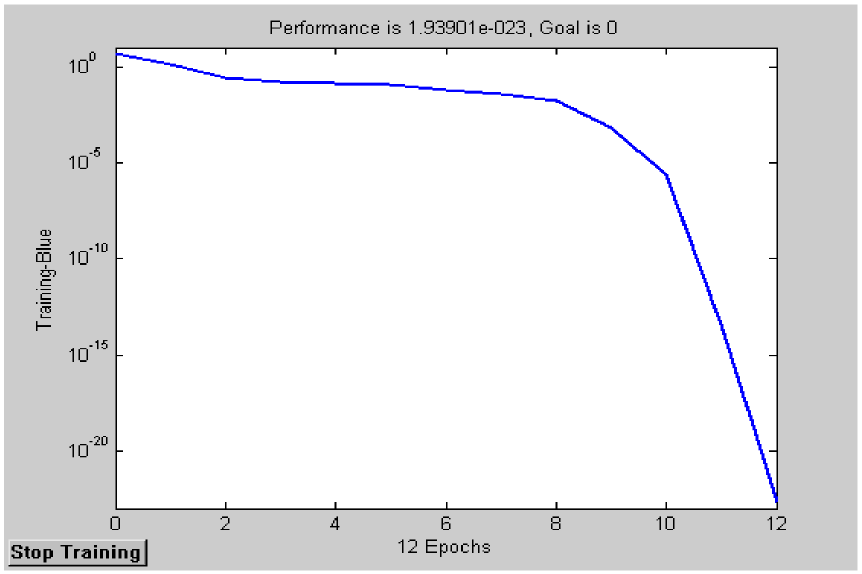

Figure1a. Learning curve of NN for Example 1

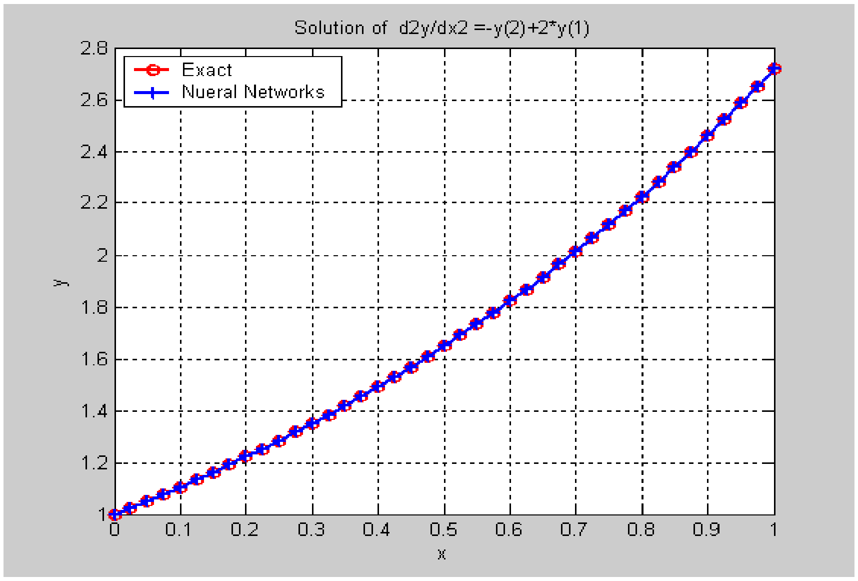

Figure1b. Curve of NN and exact for Example 1

The result obtained by the neural network $y_2(0)$ is 1 where the exact $y_2(0)$ is 1. The accuracy of the final value of $y_2(0)$ can be increased by reducing the step size and increasing the number of estimations or the accuracy can be increased by using higher order integration method. The maximum error value is $\max\left|y_{exact} - y_{NN}\right|$ =1.2089E-008

Example 2. (Non-linear Problem)

Consider the following nonlinear BVPs:

$$\frac{d^2y}{dx^2} = \frac{1}{2x^2}(y^3 - 2y^2), \quad y(1) = 1, \quad y(2) = \frac{4}{3}.$$

Exact solution $\left(y(x) = \frac{2x}{x+1}\right)$.

$y_2(1)$= -1, 0, 2, 3, and we integrate the system by Range-Kutta method using step h=0.5 we get the following table:

Table 3-a.

$y_2(a)$	-1	0	1	2.	3
$y_1(b)$	-0.1026	0.8499	1.8307	2.8977	4.1382

We can write the above table as:

Table 3-b.

P	-0.1026	0.8499	1.8307	2.8977	4.1382
T	-1	0	1	2.	3

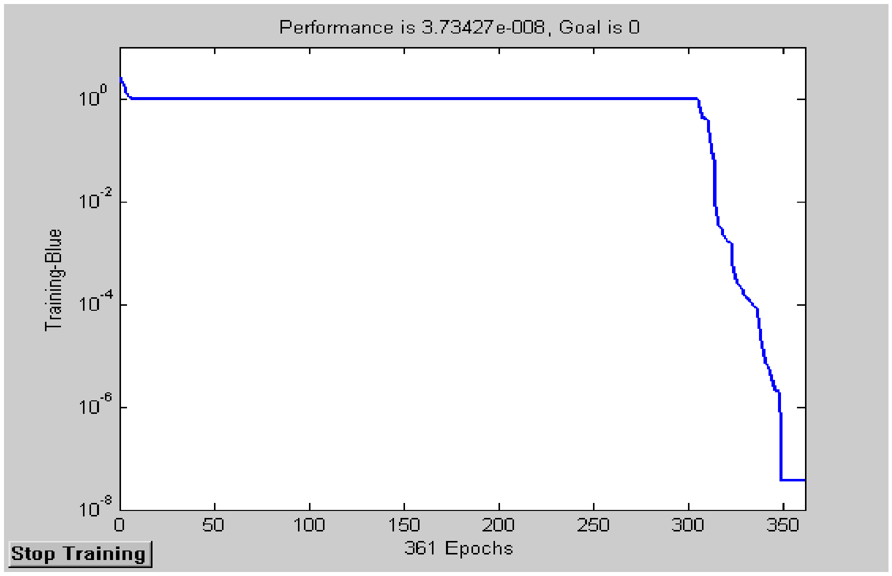

Figure2a. Learning curve of NN for Example 2

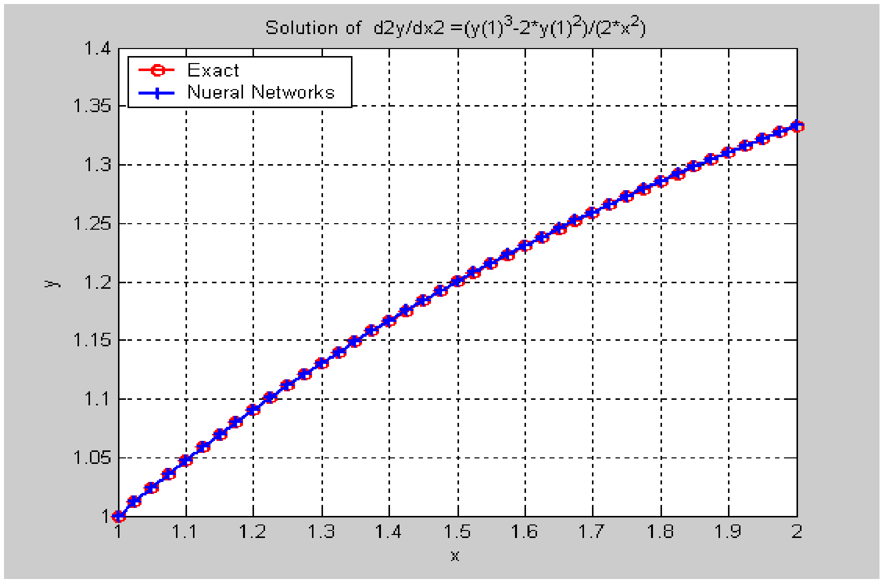

Figure2b. Curve of NN and exact for Example 2

The result obtained by the neural network $y_2(1)$ is 0.4997 where the exact $y_2(1)$ is 0.5. The accuracy of the final value of $y_2(1)$ can be increased by reducing the step size and increasing the number of estimations or the accuracy can be increased by using higher order integration method. Maximum error value is $\max|y_{exact} - y_{NN}|$ 44.3729E-004.

Example 3. (Linear Problem)

Consider the following linear BVPs:

$$\frac{d^2y}{dx^2} = x\frac{dy}{dx} + 3y + 4.2x \quad , y(0) = 0, \quad \mathrm{y}(1) = 1.9\,.$$

Exact solution $y = x^3 + 0.9x$.

$y_2(0)$=-1, 0, 2, 3, and we integrate the system by Runge-Kutta method we get the following table:

Table 4-a.

$y_2(a)$	-1	0	1	2.	3
$y_1(b)$	-0.6309	0.7000	2.0356	3.3713	4.7071

We can write the above table as:

Table 4-b.

P	-0.6309	0.7000	2.0356	3.3713	4.7071.
T	-1	0	1	2.	3

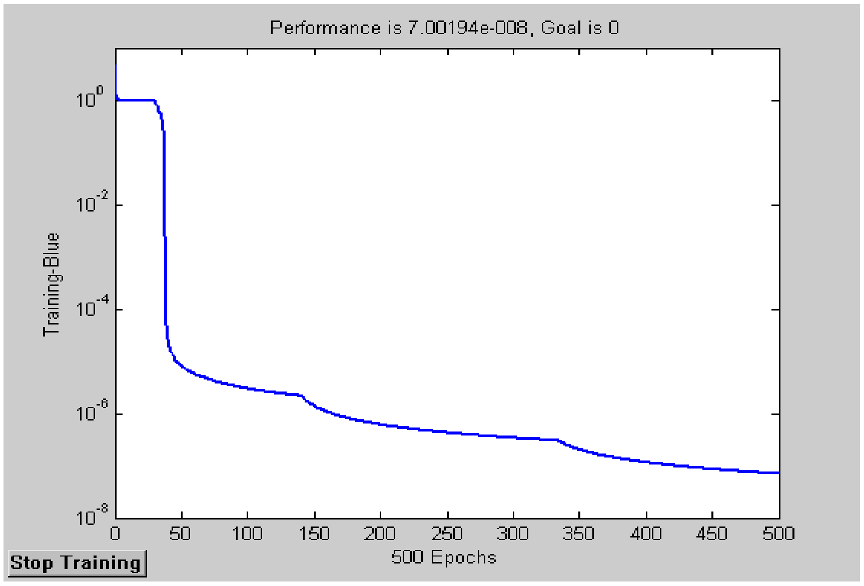

Figure3a. Learning curve of NN for Example 3

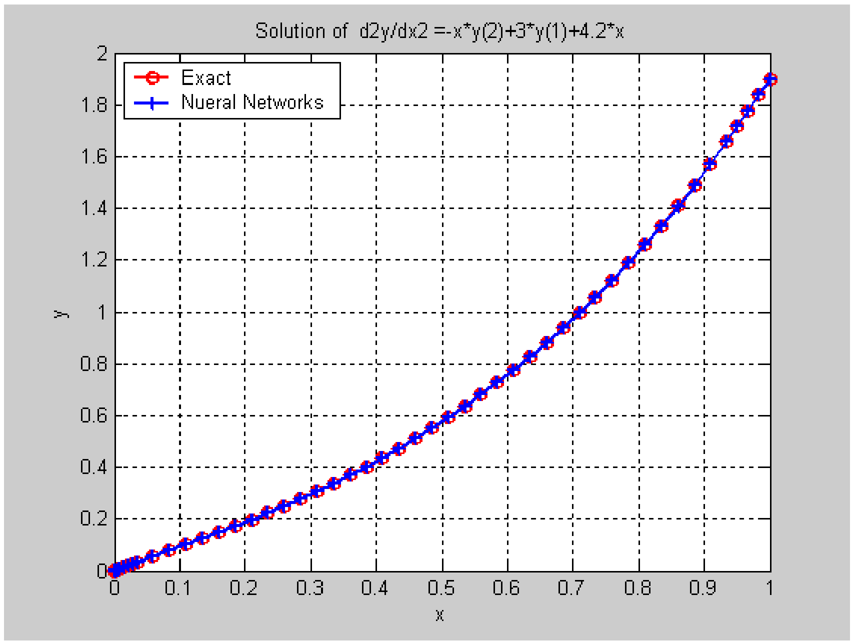

Figure3b. Curve of NN and exact for Example 3

The result obtained by the neural network $y_2(0)$ is 0.9266 where the exact $y_2(0)$ is 0.9. The accuracy of the final value of $y_2(0)$ can be increased by reducing the step size and increasing the number of estimations or the accuracy can be increased by using higher order integration method. The maximum error value is $\max|y_{exact} - y_{NN}|$ =5.9884E-004.

Example 4. (Non-linear Problem)

Consider the following nonlinear BVPs:

$$\frac{d^2y}{dx^2} = y^3 - y\frac{dy}{dx} \quad ,$$

$$y(1) = \frac{1}{2} \quad y(2) = \frac{1}{3} \ .$$

Exact solution $y = \dfrac{1}{x+1}$.

$y_2(0)$=-1, 0, 2, 3, and we integrate the system by Runge-Kutta method, we get the following table:

Table 5-a.

$y_2(a)$	-1	0	1	2.	3
$y_1(b)$	0.3953-	0.5564	1.4510	2.4812	3.8444

We can write the above table as:

Table 5-b.

P	0.3953-	0.5564	1.4510	2.4812	3.8444
T	-1	0	1	2.	3

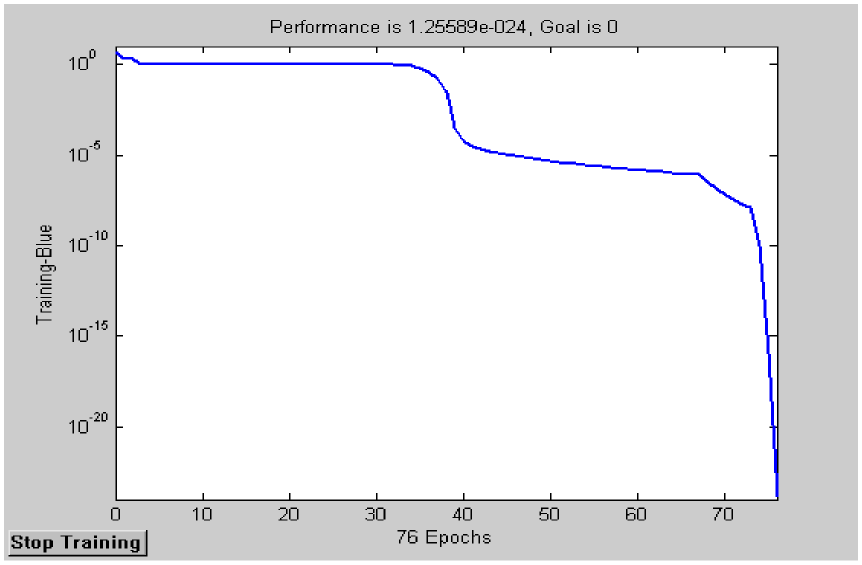

Figure 4a. Learning curve of NN for Example 4

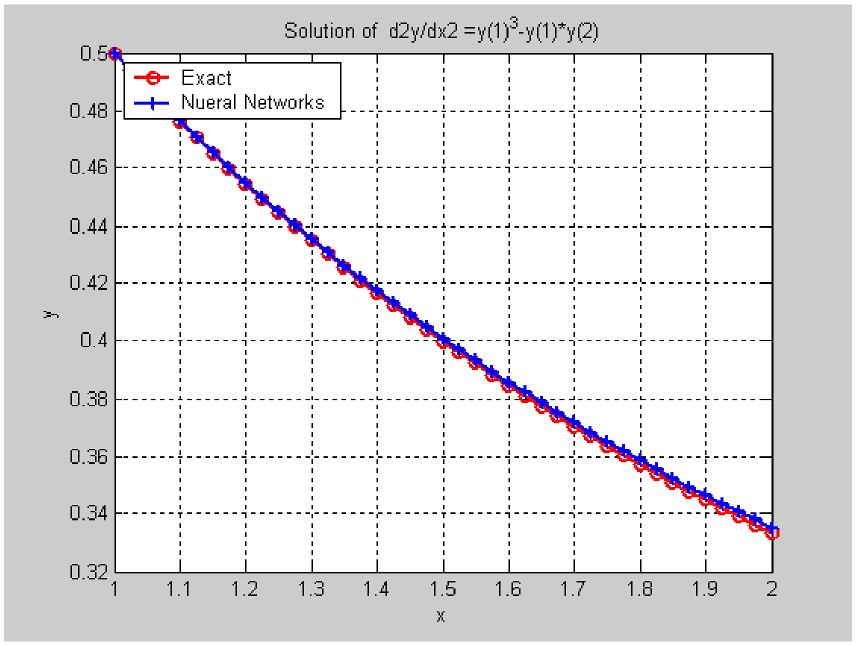

Figure 4b. Curve of NN and exact for Example 4

The result obtained by the neural network $y_2(0)$ is 0.2082 where the exact $y_2(0)$ is 0.25. The accuracy of the final value of $y_2(0)$ can be increased by reducing the step size and increasing the number of estimations or the accuracy can be increased by using higher order integration method. The maximum error value is $\max|y_{exact} - y_{NN}|$ =0.0097.

Example 5.

Consider the following linear BVPs:

$$\frac{d^2y}{dx^2} = \frac{2y}{x^2} - \frac{1}{x} \quad , \qquad y(2) = y(3) = 0.$$

Exact solution $y = \frac{1}{38}(19x - 5x^2 - \frac{36}{x})$.

y2(2)=-1,0,2,3,and we integrate the system by Rungge-Kutta method .we get the following table:

Table 5-a.

$y_2(a)$	-1	0	1	2.	3
$y_1(b)$	1.2778-	-0.2222	0.8333	1.8889	2.9444

We can write the above table as:

Table 5-b.

P	1.2778-	-0.2222	0.8333	1.8889	2.9444
T	-1	0	1	2.	3

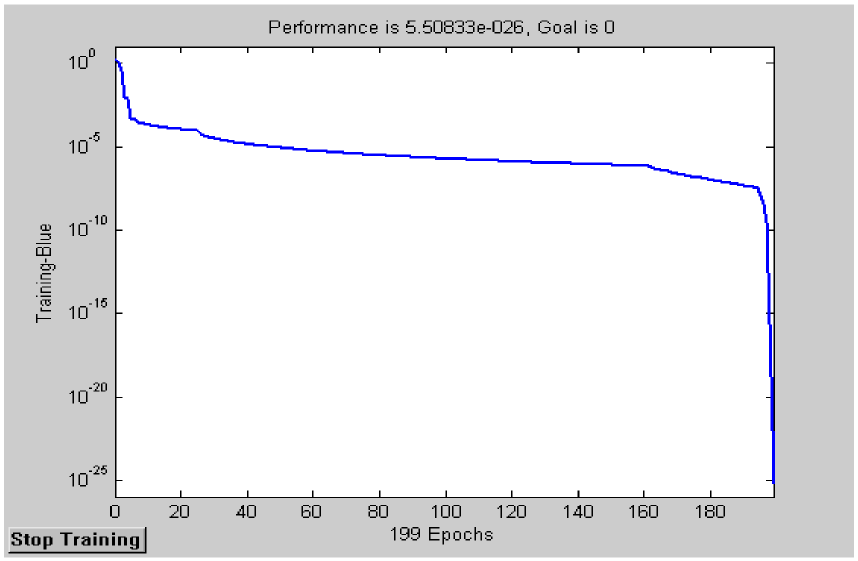

Figure 4a. Learning curve of NN for Example 5

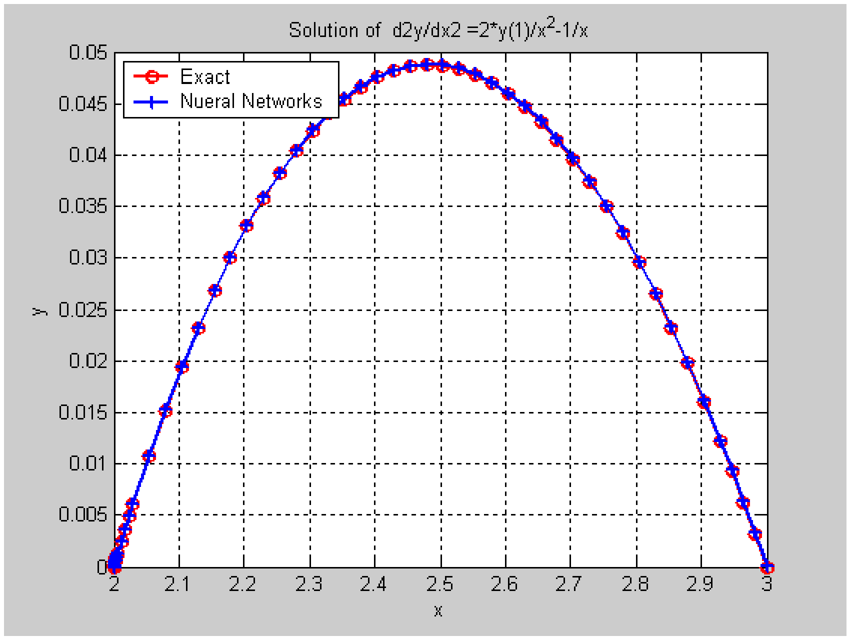

Figure5b. Curve of NN and exact for Example 5

The result obtained by the neural network $y_2(2)$ is 0.2101 where the exact $y_2(2)$ is 0.2105. The accuracy of the final value of $y_2(2)$ can be increased by reducing the step size and increasing the number of estimations or by using higher order integration method. The maximum error value is $\max|y_{exact} - y_{NN}|$ =0.1 E-006.

6. Conclusion

The result, in this research, for solving BVPs in ODEs obtained by using (integration and interpolation techniques) and Neural Networks we tabulated. Software developed, called (SNN), helps the user to specify and solve their problems. The system is developed using Matlab. Five examples are considered to show the effectiveness of the ANN for solving BVPs. The Maximum error and solution time of the technique is computed for each problem.

Table 6. Maximum error and time for each example

No. Of Examples	Maximum error	Time
1	1.2089E-008	5.9070
2	44.3729E-004	3.8750
3	5.9884E-004	6.4380
4	0.0097	5.7660
5	0.1 E-006	4.5110

REFERENCES

Barto, A. and Sutton, R. (1981). "Landmark learning: an illustration of associative search", *Biological Cybernetics* vol. 42, pp. 1-8.

Bushor, W. (1960). "The perceptron-an experiment in learning", *Electronics* vol. 33, pp. 56-59.

Dolin, B. and Merelo, J. (2002). "Resource review: a web-based tour of genetic programming", *Genetic Programming and Evolvable Machines* vol. 3 pp. 311-313.

Fausett, L. (1994). "Fundamentals of Neural Network Architectures Algorithms and applications", Prentice-Hall Englewood Cliffs.

Grant, E. and Zhang, B. (1989). "A neural-net approach to supervised learning of pole balancing", presented at IEEE International Symposium on Intelligent Control Albany, NY USA.

Kais, I. I. and Abudu, R.M. (2001, March 5). "Solving Stiff Two Point Boundary Value Problem by Use of Shooting Method Problems", a. paper presented to the 2nd International Seminar on Numerical Analysis in Engineering University of North Sumatra Medan, 14-15, 24 – 31.

Keller, H. (1968). "Numerical methods for two-point BVPs ", Blaisdeell Mass.

Khalaf, B. (1988). "Boundary Value Techniques in continues system simulation", M. Phil Thesis Bradford University.

Khalaf, B. (1990). "Parallel numerical algorithm for solving ODEs", Ph.D. Thesis Leeds University.

Khalaf, B. (1997). "Techniques for controlling the stability of numerical solution of initial value", Raf. Jour. Sci. Vol.27, pp. 135-146.

Khalaf, B. and Al-Nema, M. (2008). Generalized Parallel Algorithms for BUPs in ODEs, The Proceeding of The second Conference on Mathematical Sciences (CMS 2008) Jordan, pp. 275-284.

Khanna, S. and Noback, C. (1963). "Neural nets and artificial intelligence", *Artificial Intelligence,* pp. 83-88.

Kuznetsova, Kuzmenko V. and Tsygelnyi, I. (1980). "Problems and future trends in neuron engineering", *Otbor i Peredacha Informatsii* vol. 62 pp. 49-55.

Mattheij, Ascher U. R. and Russel, R. (1988). "Numerical Solution of Boundary for ODEs", Prentice-Hall New Jersey.

Maxwell, T.C., Giles, Y. Lee and Chen, H. (1986)."Nonlinear dynamics of artificial neural systems", presented at Neural Networks for Computing Snowbird UT USA.

McFall, Kevin S. (2006). "An Artificial Neural Network Method For Solving Boundary Value Problems With Arbitrary Irregular Boundaries", Ph.D. Thesis Georgia Institute of Technology.

Meireles, M.P. Almeida and Simoes, M. (2003). "A comprehensive review for industrial applicability of artificial neural networks", *IEEE Transactions on Industrial Electronics* vol. 50, pp. 585-601.

Shu-Hsien, L. (2005). "Expert system methodologies and applications-a decade review from 1995 to 2004", *Expert Systems with Applications* vol. 28.

Tsoi, A. (1989). "Multilayer perceptron trained using radial basis functions", *Electronic Letters*

Turing, A. (1950). "Computing machinery and intelligence", *Mind* vol. 59, pp. 433-460.

Turksen, I. (1997). "Fuzzy logic: review of recent concerns", presented at IEEE International Conference on Systems Man and Cybernetics. vol. 25, pp. 1286-1297.

Zaslavsky, Gaber M.A. and Krishnaswamy, S. (2005). "Mining data streams: a review", *SIGMOD Record* vol. 34, pp. 18-26.