Mathematics and Statistics 10(2): 366-377, 2022
DOI: 10.13189/ms.2022.100211 http://www.hrpub.org

Solution of 1st Order Stiff Ordinary Differential Equations Using Feed Forward Neural Network and Bayesian Regularization Algorithm

Rootvesh Mehta[1], Sandeep Malhotra[2], Dhiren Pandit[2,*], Manoj Sahni[3]

[1]Department of Mathematics, Indus University, Ahmedabad, Gujarat, India
[2]Department of Mathematics, Nirma University, Ahmedabad, Gujarat, India
[3]Department of Mathematics, School of Technology, Pandit Deendayal Energy University, Gandhinagar, India

Received January 2, 2022; Revised March 1, 2022; Accepted March 27, 2022

Cite This Paper in the following Citation Styles

(a): *[1] Rootvesh Mehta, Sandeep Malhotra, Dhiren Pandit, Manoj Sahni , "Solution of 1st Order Stiff Ordinary Differential Equations Using Feed Forward Neural Network and Bayesian Regularization Algorithm," Mathematics and Statistics, Vol. 10, No. 2, pp. 366 - 377, 2022. DOI: 10.13189/ms.2022.100211.*

(b): *Rootvesh Mehta, Sandeep Malhotra, Dhiren Pandit, Manoj Sahni (2022). Solution of 1st Order Stiff Ordinary Differential Equations Using Feed Forward Neural Network and Bayesian Regularization Algorithm. Mathematics and Statistics, 10(2), 366 - 377. DOI: 10.13189/ms.2022.100211.*

Abstract A stiff equation is a differential equation for which certain numerical methods are not stable, unless the step length is taken to be extraordinarily small. The stiff differential equation includes few terms that could result in speedy variation in the solution. When integrating a differential equation numerically, the requisite step length should be incredibly small. In the solution curve, much variation can be observed where the solution curve straightens out to approach a line with slope almost zero. The phenomenon of stiffness is observed when the step-size is unacceptably small in a region where the solution curve is very smooth. A lot of work on solving the stiff ordinary differential equations (ODEs) have been done by researchers with numbers of numerical methods that currently exist. Extensive research has been done to unveil the comparison between their rate of convergence, number of computations, accuracy, and capability to solve certain type of test problems. In the present work, an advanced Feed Forward Neural Network (FFNN) and Bayesian regularization algorithm-based method is implemented to solve first order stiff ordinary differential equations and system of ordinary differential equations. Using proposed method, the problems are solved for various time steps and comparisons are made with available analytical solutions and other existing methods. A problem is simulated using proposed FFNN model and accuracy has been acquired with less calculation efforts and time. The outcome of the work is showing good result to use artificial neural network methods to solve various types of stiff differential equations in near future.

Keywords Feed Forward Neural Network, Multilayer Perceptron Neural Network, Stiff Ordinary Differential Equations, Back Propagation Algorithm, Bayesian Regularization Algorithm

1. Introduction

The genesis of differential equations is balance laws of science and engineering. Ordinary differential equations, Partial differential equations appear inevitably in many mathematical models of engineering and physical problems. As a continuation, one class of differential equations known as Stiff differential equations which has many applications in engineering and arises in many physical problems based on time dependency is used by many researchers [1, 2]. In general, stiff differential equations arising from the models which are widely based on the differing time scales. These equations appear in study of electrical circuits, vibrations, chemical reactions,

atmospheric chemistry problems, biosciences and many more fields. In general, stiff differential equations are defined as equations having terms which hints to the quick variations in the solution. Stiff equations define as a differential equation whose analytic solution having a term which decays exponentially to zero with increment in step size, but derivatives of such terms are having much greater magnitude than the term itself [3].

Various numerical methods are available to solve differential equations but having limitations to solve stiff differential equations. There are few numerical schemes available for solving stiff types of differential equations but due to transient and steady state portion availability in the solution, these equations are behaving ill-conditionally in a computational sense and thus they seriously defy numerical methods [4]. For achieving accurate solution of stiff problems over time intervals, due to bounded stability regions of explicit methods all explicit methods are ineffective to solve stiffness issue and thus researchers are required to use implicit numerical schemes with very small step-size which is a serious time taking process [5]. Due to these limitations researchers started looking for alternate approaches to solve stiff differential equations and one of them is artificial neural network approach. Initially it was quite challenging to solve stiff differential equations accurately using artificial neural network method but now due to availability of advanced training algorithms like Levenberg Marquardt algorithm (LMA), Bayesian regularization algorithm and Scaled Conjugate Gradient algorithm (SCGA) which can be clubbed with artificial neural network method give satisfactory results in solving stiff differential equations.

In the present work attempts are made to solve first order stiff ode and stiff system of odes using FFNN model clubbing with training algorithms like Bayesian Regularization Algorithm, Levenberg-Marquardt Algorithm, and Scaled Conjugate Gradient algorithm. The results obtained are compared and conclusion is made based on the comparisons. The error calculations are performed and depicted in the table.

2. Materials and Methods

An Artificial Neural Network (ANN) is basically a data processing system which includes a large number of simple highly interconnected processing elements, known as neurons and a directed graph, where nodes represent neurons and edges represent synaptic lengths [6]. ANN is a data modeling tool that depends upon various parameters and learning methods. Artificial Neural networks are characteristically organized in layers. Layers are made up of a number of interconnected neurons/nodes which contain activation functions. ANN processes information through parallel neurons/nodes to solve specific problems. Using learning ANN, we obtain information which is stored within the interneuron connections and can be expressed by numerical values called weights. These weights are helpful to calculate output signal values for a new testing input signal value. Actual processing is done via system of weighted connections through presented patterns of the input layer, which communicates to one or more hidden layers. The hidden layers then link to an output layer as shown in figure 1.

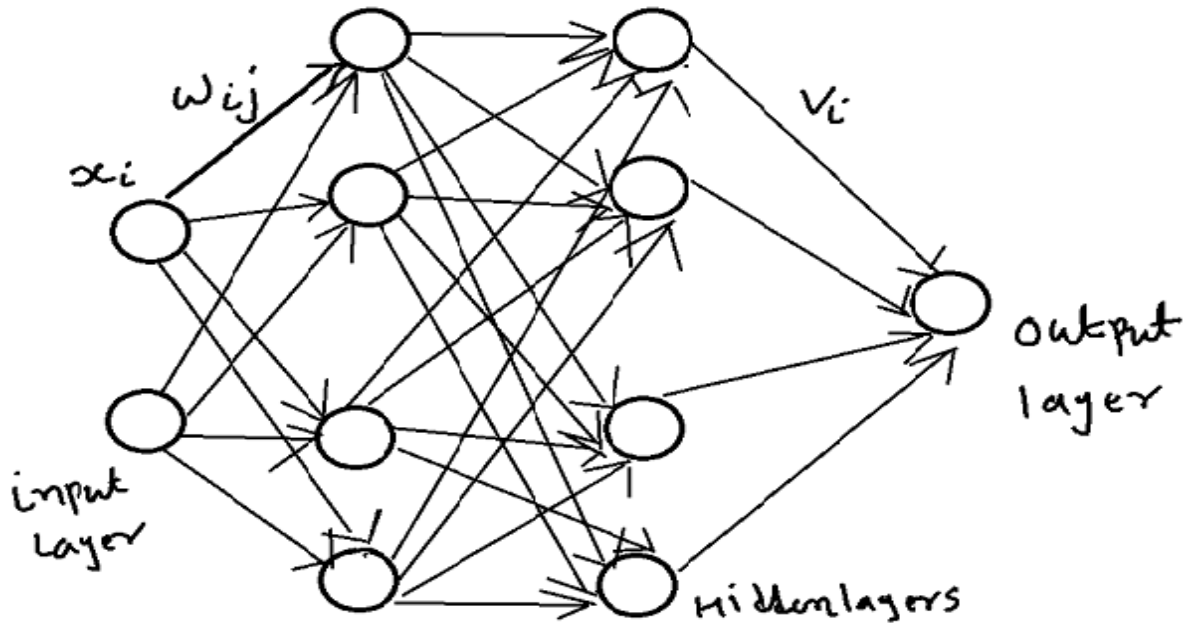

Figure 1. ANN Model

In figure 1, x_j denotes input nodes, w_{ij} represents weights from the input layer to the hidden layer, and v_i and y are the weights from the hidden layer to the output layer and the output node, respectively. Various neural network architectures like Feed Forward Neural Network (FFNN), Feedback Neural Network (FNN), Recurrent Neural Network (RNN), Radial Basis Functional Neural Network (RBFNN), Hopfield Neural Network (HNN) and Cellular Neural Network (CNN) are found in literature.

In the Feed Forward Neural Network architecture which is used in proposed work, the layers are formed using neurons. In a layer the neurons are receiving input from the previous layer and the next layer is fed by their output. Here, the data is strictly moving from input to output nodes in forward way. There is no feedback loop generated, that is why the output of any layer does not affect the same layer [7]. The FFNN architecture works on two models namely Single Layer Perceptron (SLP) Model and Multilayer Perceptron (MLP) Model. The Single-layer perceptron network is the simplest kind of neural network which consists of output nodes in a single layer only. The series of weights feed inputs to the outputs and in each of the neuron node the sum of products of the weights and the inputs are calculated. If $X= (x_1, x_2,\ldots, x_n)$ denotes the input vector, $W = w_{ij}$ denotes the weight matrix or connection matrix, and [WX] is the net input value, which is a scalar product of input vectors and weight vectors w_{ij}, f is the activation function and $O = (o_1, o_2, \ldots, o_m)$ is the output vector, then the block diagram of the single layer perceptron feed-forward ANN can be simply presented as figure 2.

Here FFNN Multilayer perceptron model is more general network architecture [8-10], in which there are hidden layers between input and output layers. Here hidden nodes neither receive inputs directly nor send any outputs to the external environment. The neurons of the first hidden layer getting inputs from the source nodes are the input layer. The neurons of second hidden layer are receiving the outputs of the first hidden layer neurons as inputs and this process continue till the end. Before directing the input to the output layer, intermediate computations are done by the hidden layer then the hidden layer neurons are linked with the input layer neurons. The weights on these links are referred to as input-hidden layer weights and the hidden layer neuron and the corresponding weights are referred to as output-hidden layer weights. The typical diagram of FFNN Multilayer perceptron model can be seen in figure 3.

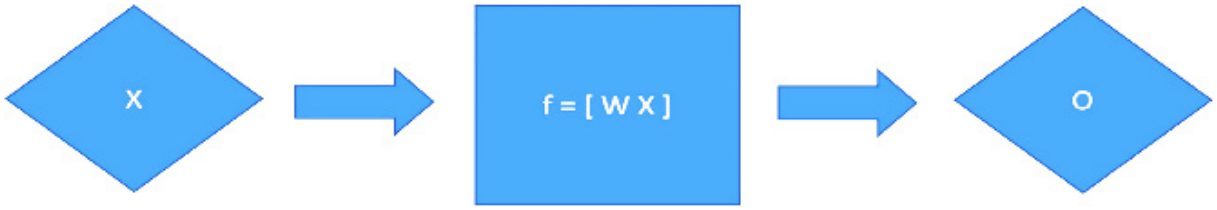

Figure 2. The block diagram of the single layer perceptron feed-forward neural network

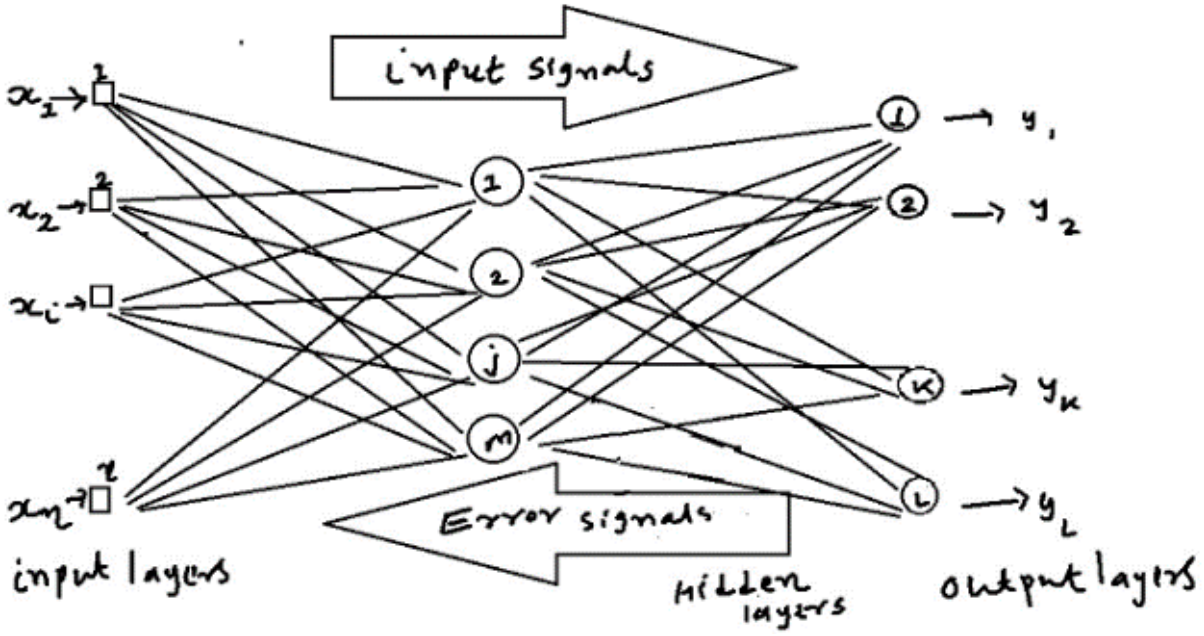

Figure 3. The block diagram of the multilayer perceptron feed-forward neural network

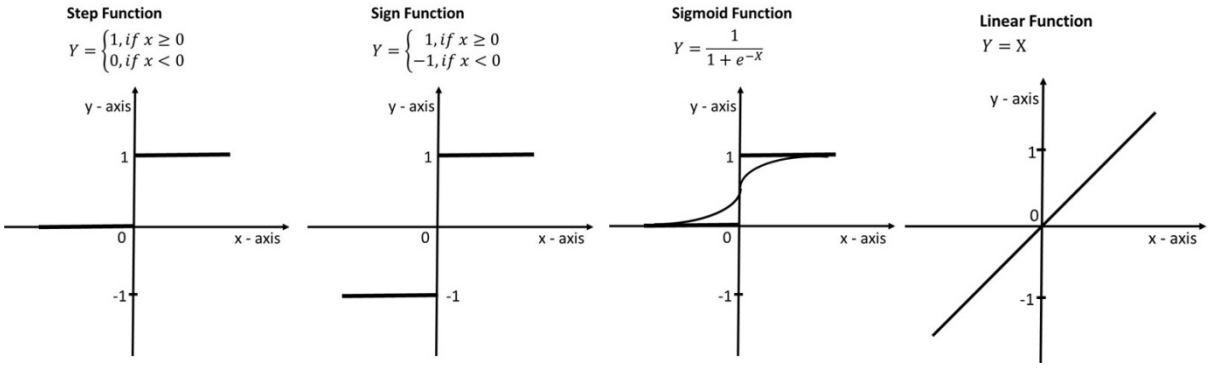

Figure 4. The list of activation functions

An ability to learn and to form a solution or output from the training of data is the most important feature of ANN. This learning can be a type of supervised, unsupervised, reinforced or competitive. To act upon the net input to receive the output of the network we need to use activation function such that the outcome of the neural network can be obtained between values like 0 and 1, or -1 and 1. Some of the activation functions which serve the purpose are shown in figure 4.

In the proposed work, mathematical model based on FFNN has been developed using log-sigmoid activation function and after that training of the network part begins.

In the process of training the weights in the connections between network layers are modified with the objective of achieving the expected output. Training in neural network is mainly of two types - Supervised and Unsupervised. In Supervised training the network is trained by providing it with input and matching output patterns. Supervised training is based on training a data sample from data source with correct classification already assigned. In unsupervised learning algorithm, ANN identifies hidden patterns of unlabeled input data. To evaluate the potential solution this type of unsupervised learning refers the ability to learn and organize information without providing an error signal. Here cluster can be formed by grouping all similar input patterns and new cluster can also be formed if input pattern is not matched. As such there is no feedback in unsupervised learning and that is why patterns, regularities, features for the input data over the output must be discovered by the network and while doing so the network might change its parameters, and this process is called self-organizing. Hence supervised training suits more to Multilayer perceptron FFNN model. In the proposed work back propagation supervised training has been performed by using Levenberg Marquardt algorithm [11, 12], Bayesian regularization algorithm [13, 14] and Scaled Conjugate Gradient algorithm [15] to obtain closed analytical form of the solution.

2.1. Proposed Model

To understand the method, consider a first order ODE

$$\frac{d\psi}{dx} = f(x, \psi) \tag{1}$$

where f is any smooth function of variable x with $x \in [0,1]$ and the initial condition $\psi(0) = A$.

Now, the trial solution of Equation (1) is

$$\psi(x) = A + xN(x, \vec{p}) \tag{2}$$

where, $N(x, \vec{p}) = \sum_{i=1}^{h} v_i \sigma(z_i)$, $z_i = w_i x + u_i$ is the output of a network with one node for x and vector $\vec{p}$.

Here, we assume trial function such that it satisfies the given initial condition. Now, differentiating both sides of equation (2), we get

$$\frac{d\psi}{dx} = N(x, \vec{p}) + x\frac{dN(x,\vec{p})}{dx} \tag{3}$$

We also know that

$$\frac{dN(x, \vec{p})}{dx} = \sum_{i=1}^{n} v_i w_i \sigma(z_i)$$

and hence, we get

$$\frac{d\psi_i}{dx} = \sum_{i=1}^{n} v_i \sigma(z_i) + x \sum_{i=1}^{n} v_i w_i \sigma(z_i) \tag{4}$$

Now, the error quantity to be minimized as

$$E\overrightarrow{(p)} = \sum_k \left\{\frac{d\psi_k(x_k)}{dx} - f(x_k, \psi_k(x_k)\right\}^2, x_k \epsilon [0,1] \tag{5}$$

By updating the parameters using advance algorithms like Levenberg Marquardt algorithm, Bayesian regularization algorithm and Scaled Conjugate Gradient algorithm, we can get results close to the analytic solution of the problem, and by comparing these results we can find an appropriate method to solve 1st order stiff Ordinary Differential Equation (ODE) as well as system of stiff ODEs.

2.2. Solution of 1st Order Stiff Ordinary Differential Equation and Stiff System of Ordinary Differential Equation

To validate our methods, we have considered first order stiff ODE and system of stiff ODE to test the results obtained.

Let us consider the first order stiff ODE as,

$$y' = -100(y - x^3) + 3x^2,\ y(0) = 1, 0 \le x \le 1 \quad (6)$$

The above differential equation can be rewritten as

$$y' + 100y = 100x^3 + 3x^2, \qquad y(0) = 1$$

Integrating Factor of the above differential equation = e^{100x}, and thus solution is given as,

$$y(e^{100x}) = \int e^{100x}(100x^3 + 3x^2)dx + C$$

$$\Rightarrow y(e^{100x}) = \left[(100x^3 + 3x^2)\frac{e^{100x}}{100} - (300x^2 + 6x)\frac{e^{100x}}{10^4} + (600x + 6)\frac{e^{100x}}{10^6} - 600\frac{e^{100x}}{10^8}\right] + C$$

$$\Rightarrow y = \left[x^3 + \frac{3x^2}{100} - \frac{3x^2}{100} - \frac{6x}{10^4} + \frac{6x}{10^4} + \frac{6}{10^6} - \frac{6}{10^6}\right] + Ce^{-100x}$$

$$\Rightarrow y = x^3 + Ce^{-100x}$$

Using the initial condition$y(0) = 1$, we get the value of constant of integration as

$$y(0) = 1 \Rightarrow C = 1$$

The analytical solution of equation (6) is now given as

$$y = e^{-100x} + x^3 \quad (7)$$

Similarly, considering the system of first order stiff ODEs as,

$$\left.\begin{array}{c} y_1' = 198y_1 + 199y_2 \\ y_2' = -398y_1 - 399y_2 \\ y_1(0) = 1, y_2(0) = -1, x \in [0,5] \end{array}\right\} \quad (8)$$

The analytical solution of the equation (8) using the initial condition is given as,

$$y_1(x) = e^{-x}, y_2(x) = -e^{-x} \quad (9)$$

FFNN solutions for the above-mentioned problems are obtained using various training algorithms like Bayesian Regularization Algorithm, Levenberg-Marquardt Algorithm, and Scaled Conjugate Gradient algorithm. The results and comparisons are discussed in the result section 2.3.

2.3. Results

First, we apply the proposed FFNN model on equation (6) using various training algorithms like Bayesian Regularization Algorithm, Levenberg-Marquardt Algorithm, and Scaled Conjugate Gradient algorithm with step size h = 0.05 and results are displayed in figure 5 and tables 1(a), 1(b) below.

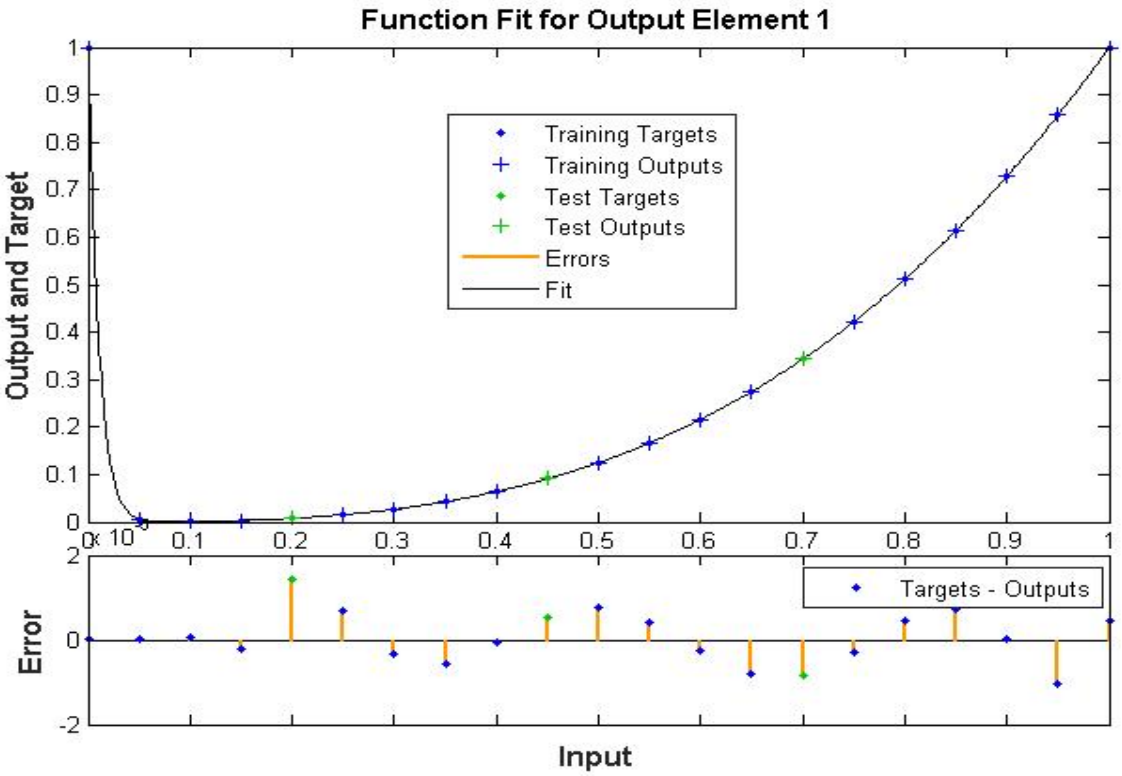

Figure 5. FFNN Solution of equation (6) with h = 0.05

Figure 5 is showing a smooth curve obtained using FFNN and Bayesian Regularization algorithm with step size h = 0.05 for the 1st order stiff ordinary differential equations (6). Comparison of the obtained FFNN and Bayesian Regularization algorithm solution of (6) with analytic solution, numerical solution using ODE23s is shown in table 1(a) and solutions obtained by the other two algorithms are shown in table 1(b).

Table 1(a). FFNN Solution of equation (6) with h = 0.05

Input x	Analytical solution (y)	ODE23s (y-output)	Error	FFNN (Bayesian Output)	Error
0	1	1	0	0.99999	4.88E-08
0.05	0.00686	0.00677	8.45E-05	0.00686	1.89E-09
0.1	0.00104	0.00104	2.11E-06	0.00104	6.20E-07
0.15	0.00337	0.00337	2.64E-06	0.00337	-2.01E-06
0.2	0.00800	0.00799	5.48E-06	0.00798	1.41E-05
0.25	0.0156	0.01561	9.65E-06	0.01561	6.86E-06
0.3	0.027	0.02698	1.51E-05	0.02700	-3.33E-06
0.35	0.04287	0.04285	2.32E-05	0.04288	-5.53E-06
0.4	0.064	0.06396	3.25E-05	0.06400	-6.40E-07
0.45	0.09112	0.09108	4.34E-05	0.09111	5.43E-06
0.5	0.125	0.1249	5.71E-05	0.12499	7.51E-06
0.55	0.16637	0.16630	7.18E-05	0.16637	4.07E-06
0.6	0.216	0.2159	9.02E-05	0.21600	-2.55E-06
0.65	0.27462	0.27451	0.00011	0.27463	-7.93E-06
0.7	0.343	0.34286	0.00013	0.34300	-8.25E-06
0.75	0.42187	0.42171	0.000161	0.42187	-2.90E-06
0.8	0.512	0.51181672	0.00018	0.5119955	4.45E-06
0.85	0.614125	0.61390907	0.00021	0.6141178	7.16E-06
0.9	0.729	0.72874445	0.00025	0.7289996	3.37E-07
0.95	0.857375	0.85707974	0.00029	0.8573853	-1.04E-05
1	1	0.99983908	0.00016	0.9999953	4.66E-06

Table 1(b). Other Solutions of equation (6) with h = 0.05

Levenberg Marquardt out put	Error	Scaled Conjugate Gradient Algorithm output	Error
0.999997902	2.1E-06	0.603838383	0.396161617
0.007219729	-0.000357	0.052442903	-0.045579956
0.002160047	-0.001115	0.031678287	-0.030632887
0.004108646	-0.000733	0.040037173	-0.036661867
0.008279439	-0.000279	0.056426348	-0.048426346
0.015815216	-0.000190	0.081695507	-0.066070507
0.027490551	-0.000491	0.112331080	-0.08533108
0.042040285	0.000835	0.139014800	-0.09613980
0.059042664	0.004957	0.153671430	-0.08967143
0.091222121	-9.71E-05	0.155304154	-0.064179154
0.14129916	-0.016299	0.146775689	-0.021775689
0.166365393	9.61E-06	0.146724128	0.019650872
0.207420358	0.008580	0.207253298	0.008746702
0.274623894	1.11E-06	0.304653186	-0.030028186
0.343003705	-3.7E-06	0.454244234	-0.111244234
0.421869205	5.79E-06	0.551440985	-0.129565985
0.127389784	0.38461	0.606161592	-0.094161592
0.614125086	-8.58E-08	0.820154881	-0.206029881
0.729000004	-4.47E-09	0.598712872	0.130287128
0.554341576	0.303033	0.890552905	-0.033177905
0.999997902	2.1E-06	0.603838383	0.396161617

Error Comparisons of all the methods are shown in below graphs:

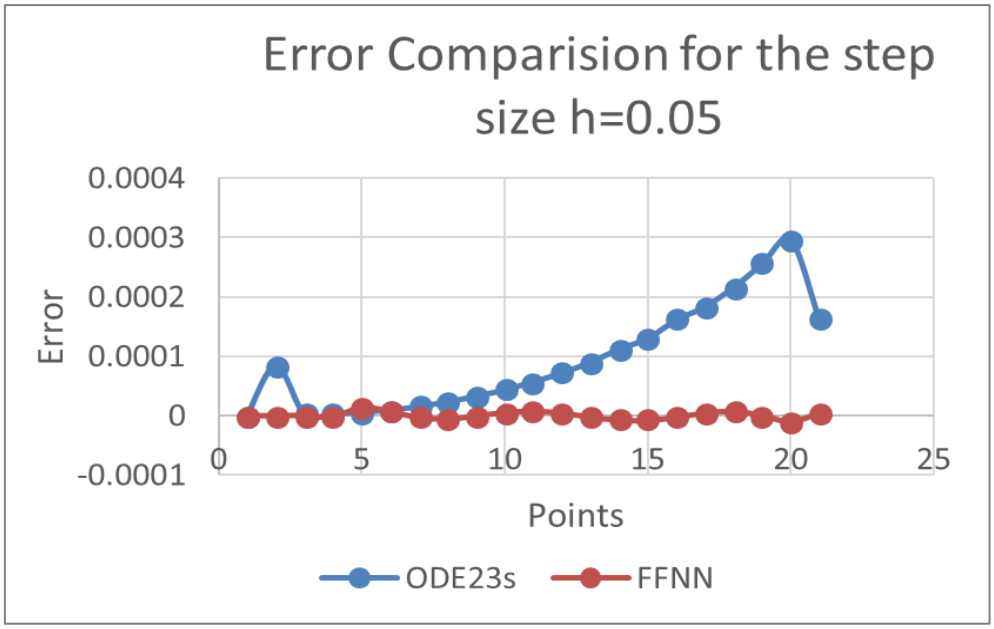

Figure 6. Error Comparisons of ODE23s and FFNN Solution of equation (6) with step size h = 0.05

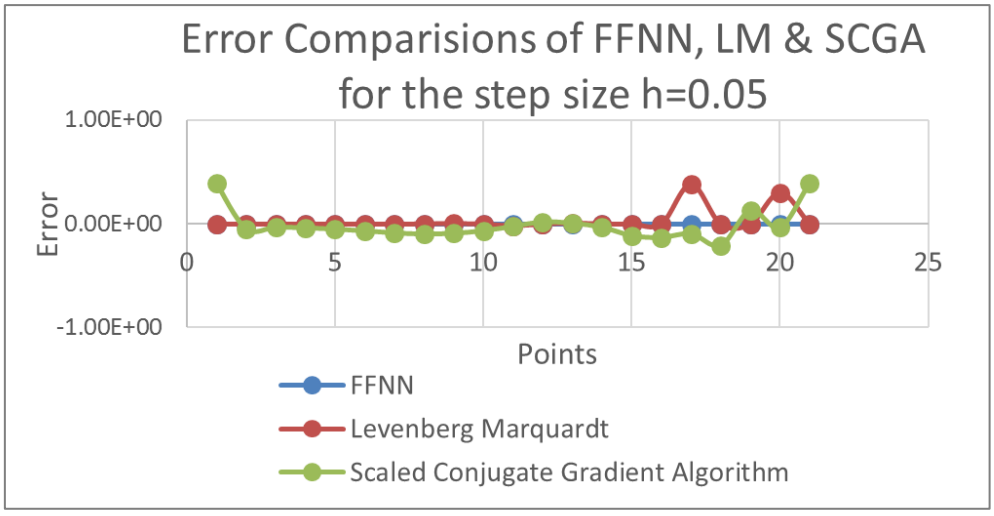

Figure 7. Error Comparisons of FFNN, LM & SCGA Solutions of equation (6) with step size h = 0.05

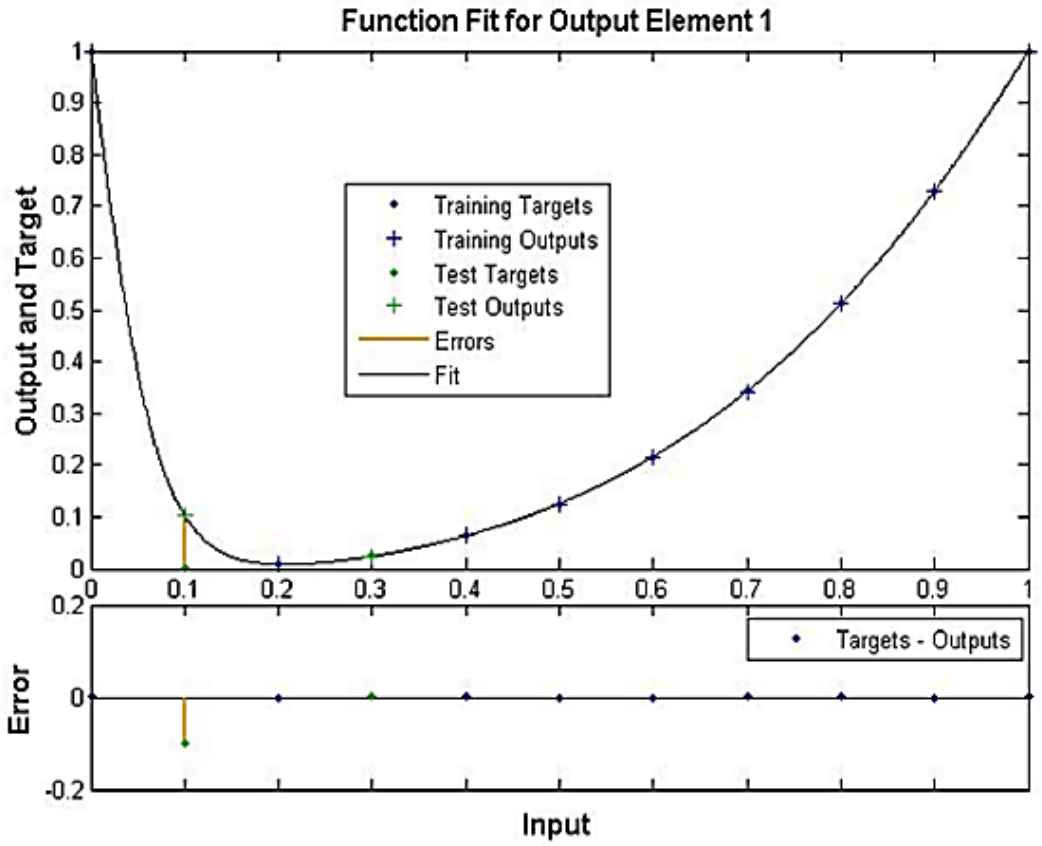

Figure 8. FFNN Solution of equation (6) with h=0.1

Table 2. FFNN Solution of equation (6) with h=0.1

*I*nput x	Analytical solution (y)	ODE23s	Error	FFNN and Bayesian output	Error
0	1	1	0	0.9999927	7.22E-05
0.1	0.00104	0.00104	0	0.0010783	-3.29E-05
0.2	0.008	0.00799	1E-05	0.0079769	2.31E-05
0.3	0.027	0.02698	2E-05	0.0270819	-8.19E-05
0.4	0.064	0.06396	4E-05	0.0636667	0.0003332
0.5	0.125	0.12494	6E-05	0.1254551	-0.000455
0.6	0.216	0.21591	9E-05	0.2160431	-4.31E-05
0.7	0.343	0.34287	0.00013	0.3424799	0.0005200
0.8	0.512	0.51181	0.00019	0.5123075	-0.000307
0.9	0.729	0.72874	0.00026	0.7311481	-0.002148
1	1	0.99983	0.00017	0.9999354	6.45E-05

Second calculation is done with h = 0.1 using proposed FFNN model and Bayesian Regularization Algorithm whose results are shown in figure 6 and table 2 with the comparison to the analytical solution, NDF (ode15s) method and Modified Rosenberck formulas (ode23s).

Figure 8 is showing a smooth curve obtained using FFNN and Bayesian Regularization algorithm with step size h = 0.1 for the 1st order stiff ordinary differential equations (6). Comparison of FFNN and Bayesian Regularization algorithm solution of equation (6) with analytic solution, numerical solution using ODE23s is presented in Table 2 and error comparison is given in figure 9.

Figure 9 is showing for step size h=0.1 still FFNN is producing stable results and thus we can go for further calculations with bigger step size.

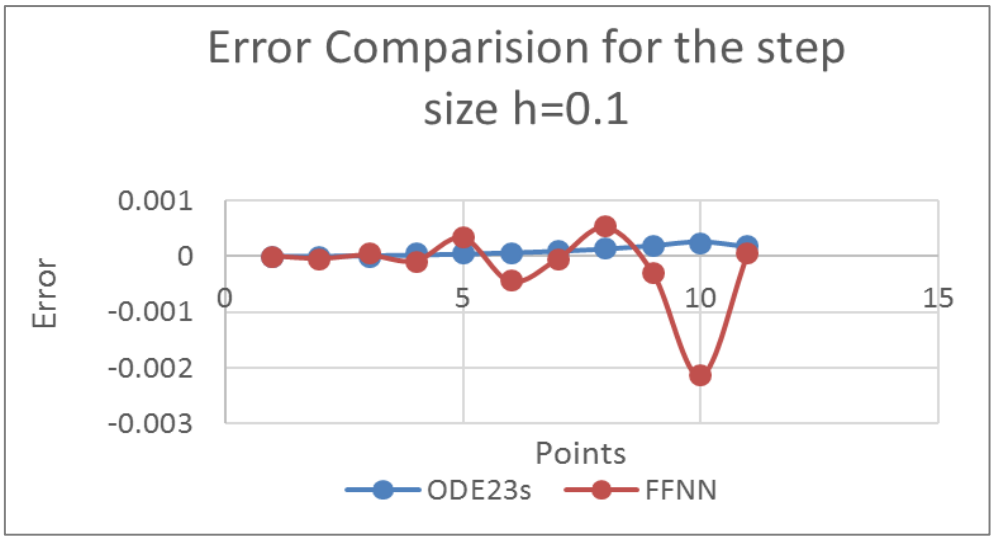

Figure 9. Error Comparisons of ODE23s and FFNN Solution of equation (6) with step size h = 0.1

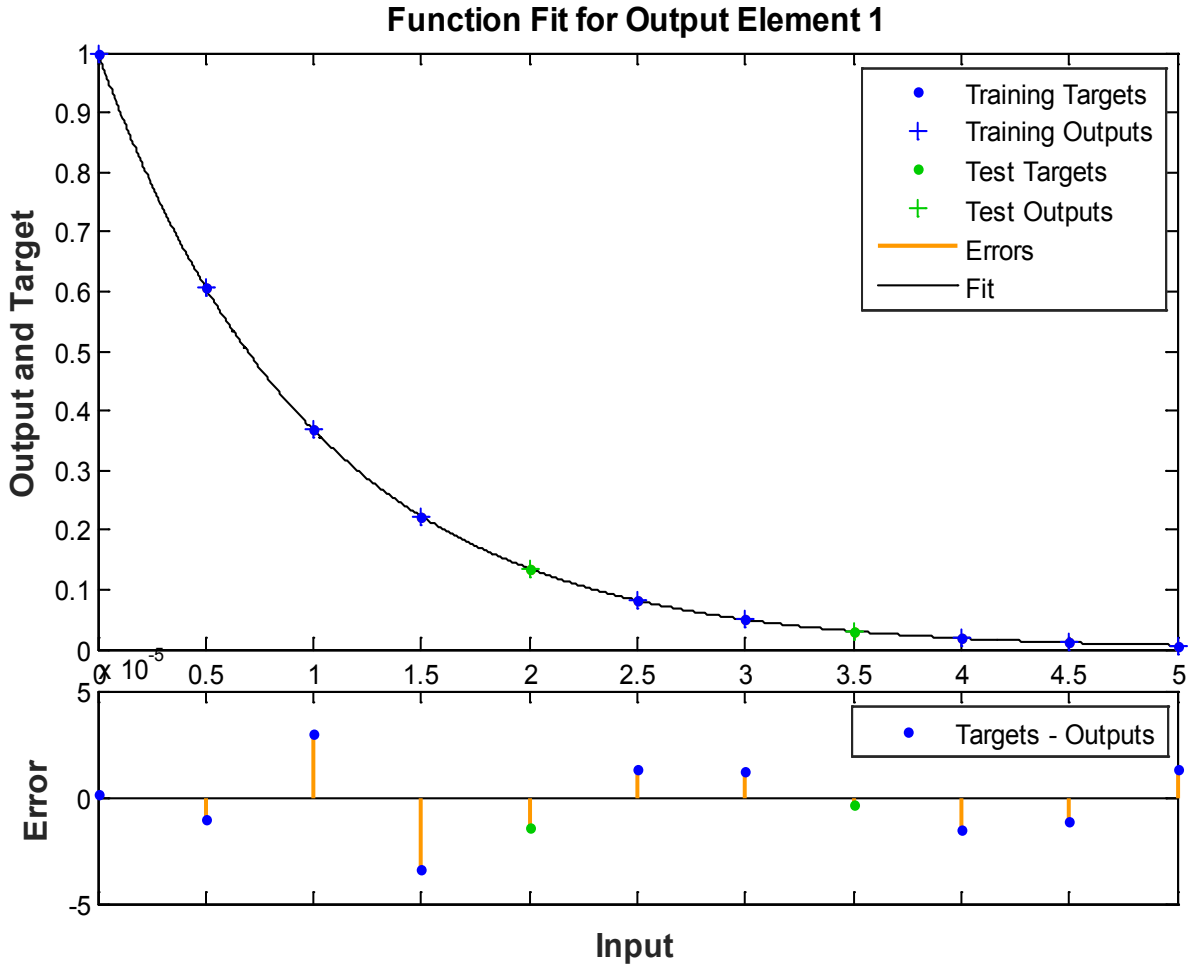

Figure 10. FFNN Bayesian Solution of equation (8) for y_1 with h=0.5

The third calculation is done using FFNN model for 1st order system of ordinary differential equations (8) by considering step size h = 0.5 clubbing training algorithms such as Bayesian Regularization Algorithm, Levenberg-Marquardt Algorithm, and Scaled Conjugate Gradient algorithm. The results obtained are shown in figures 9 and 10 and tables 3(a) and 3(b).

Figures 10 and 11 are showing a smooth curve obtained using FFNN and Bayesian Regularization algorithm with step size taken h = 0.5 for the 1st order stiff system of ordinary differential equations (8). Comparison of the obtained FFNN and Bayesian Regularization algorithm solution of (8) with analytic solution is shown in Table 3a and the solution obtained by the other two algorithms are shown in Table 3b.

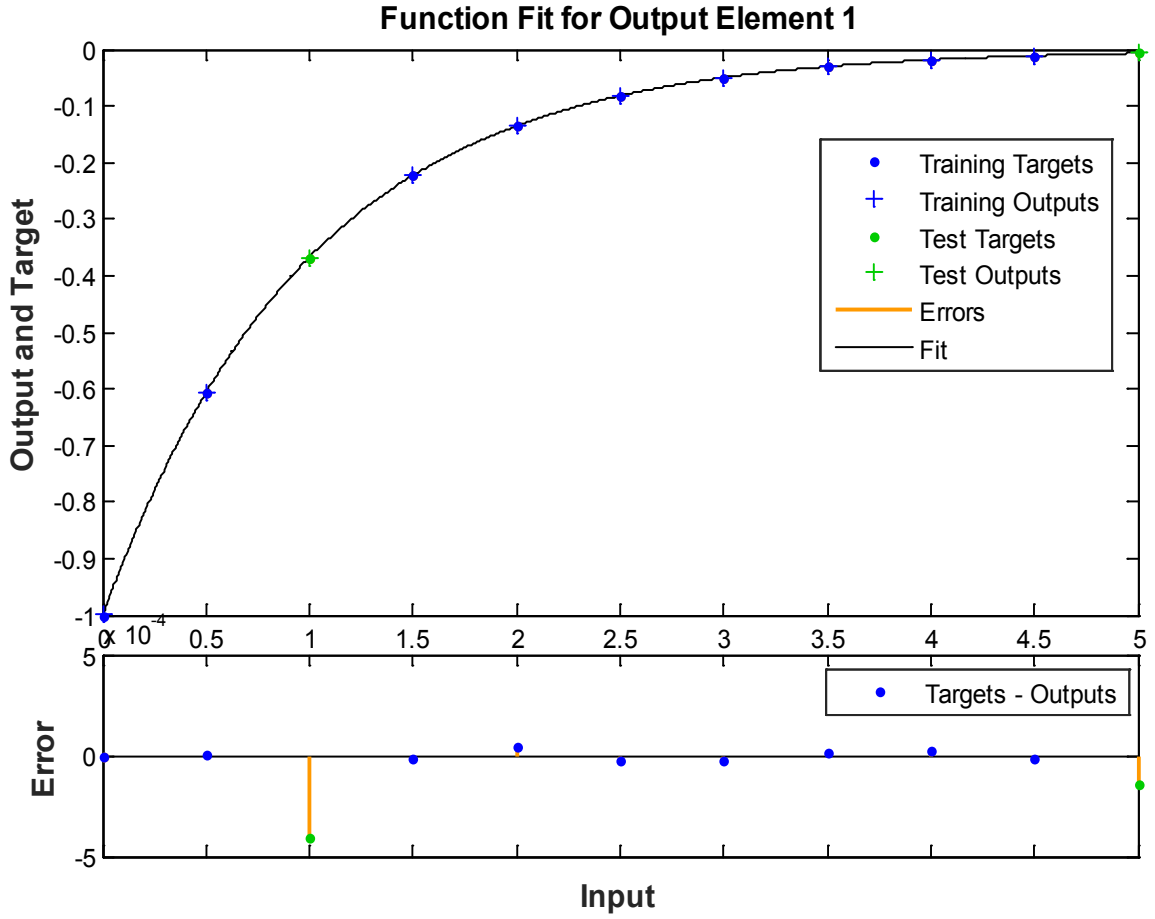

Figure 11. FFNN Bayesian Solution of equation (8) for y_2 with h=0.5

Table 3(a). FFNN Solution of equation (8) for y_1 and y_2 with h=0.5

Input	Analytic Solutions		FFNN (BAYESIAN)			
x	y1	y2	y1	y2	Error y_1	Error y_2
0	1	-1	1	-1	1.39E-06	-4.19E-07
0.5	0.607	-0.61	0.607	-0.61	-1.03E-05	2.13E-06
1	0.368	-0.37	0.368	-0.37	3.00E-05	-0.000403
1.5	0.223	-0.22	0.223	-0.22	-3.37E-05	-1.94E-05
2	0.135	-0.14	0.135	-0.14	-1.43E-05	4.36E-05
2.5	0.082	-0.08	0.082	-0.08	1.31E-05	-2.06E-05
3	0.05	-0.05	0.05	-0.05	1.27E-05	-2.79E-05
3.5	0.03	-0.03	0.03	-0.03	-3.39E-06	1.10E-05
4	0.018	-0.02	0.018	-0.02	-1.52E-05	2.94E-05
4.5	0.011	-0.01	0.011	-0.01	-1.09E-05	-1.80E-05
5	0.007	-0.01	0.007	-0.01	1.30E-05	-0.000143

Table 3(b). Other Solutions of equation (8) for y_1 and y_2 with h=0.5

FFNN (Levenberg Marquardt)				FFNN (Scaled Conjugate gradient)			
y_1	y_2	Error y_1	Error y_2	y_1	y_2	Error y_1	Error y_2
1.319	-1	-0.31994	-1.23E-1	0.925	-1.000	0.074	0.000817
-0.269	-0.606	0.875606	7.20E-13	0.384	0.925	0.222	-1.53166
0.368	-0.367	-0.00025	-5.67E-1	0.762	0.433	-0.394	-0.80156
0.074	-0.223	0.148691	3.03E-12	0.456	0.043	-0.233	-0.26672
0.135	-0.135	-0.00053	-3.33E-1	0.242	-0.139	-0.106	0.004124
0.074	-0.082	0.007774	1.42E-10	0.000	-0.069	0.081	-0.01240
0.066	-0.058	-0.01686	0.009099	-0.038	-0.054	0.088	0.005163
0.066	-0.030	-0.03618	-2.46E-1	-0.043	-0.039	0.073	0.009646
0.067	0.000	-0.04893	-0.01883	-0.043	-0.024	0.061	0.005989
0.068	0.024	-0.05713	-0.03585	-0.042	-0.012	0.053	0.001272
0.069	0.041	-0.06232	-0.04783	-0.042	-0.004	0.049	-0.00225

3. Conclusion

In the present work attempts are made to solve first order stiff ode and stiff system of odes using FFNN model clubbing training algorithms like Bayesian Regularization Algorithm, Levenberg-Marquardt Algorithm, and Scaled Conjugate Gradient Algorithm. The aim of the study was to obtain the good approximated solution with fewer calculations, memory space and time interval requirements. In FFNN MLP model Bayesian algorithm comparatively requires more time than Levenberg-Maraquadrt and Scaled Conjugate Gradient algorithms but avails good results with less memory and calculations. In many ways changing various factors like number of nodes in hidden layers, step size, data size, algorithms results can be derived and comparison are made in calculations to get comparatively good results which can be obtained by using FFNN and Bayesian Regularization algorithm. Because of the good results obtained by the proposed algorithm we propose to use this method for solving higher order stiff ordinary differential equations and applications of stiff differential equations in near future too.

REFERENCES

[1] Suyong K., Weiqi J., Sili D., Yingbo M., Christopher R., "Stiff Neural Ordinary Differential Equations," Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 31, no. 9, pp. 093122, 2021. DOI: 10.1063/5.0060697

[2] Weiqi J., Weilun Q., Zhiyu S., Shaowu P., Sili D., "Stiff-PINN: Physics-Informed Neural Network for Stiff Chemical Kinetics," Journal of Physical Chemistry, vol. 125, no. 36, pp. 8098-8106, 2021. DOI: 10.1021/acs.jpca.1c05102

[3] Richard L. Burden J., Douglas F, "Numerical Analysis", Cengage Learning, USA, 2011.

[4] Miranker W.L., "The Computational Theory of Stiff Differential Equations", Publicazioni IAC Roma Ser. III N. 102, 1975.

[5] Malhotra S., "Simulation of Navier Stokes equation represents the transient model of single tube heat exchanger with vapor-liquid two phase flow inside", International Journal of Emerging Technologies in Computational and Applied Sciences, vol. 5, no. 5, pp. 540-547, 2013.

[6] M. Minsky and S. Papert. Perceptrons. MIT Press, Cambridge, MA, 1969.

[7] T. Khanna. Foundations of Neural Networks. Addison-Wesley Press, Boston, USA, 1990.

[8] Neha Y., Anupam Y., Manoj K., "An introduction of neural network methods for differential equations" Springer Briefs in Applied Sciences and Technology, 2015.

[9] Kumar M, Yadav N "Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey," Computers and Mathematics with Applications, vol. 62, no. 10, pp. 3796–3811, 2011. DOI: 10.1016/j.camwa.2011.09.028.

[10] Kumar M, Yadav N "Numerical Solution of Bratu's Problem Using Multilayer Perceptron Neural Network Method" National Academy Science Letters. vol. 38, no. 5, 425–428. 2015 DOI: 10.1007/s40009-015-0359-3.

[11] Marquardt, D., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," SIAM Journal on Applied Mathematics, vol. 11, no. 2, pp. 431–441, 1963. DOI: 10.1137/0111030

[12] Hagan, M.T., and M. Menhaj, "Training feed-forward networks with the Marquardt algorithm," IEEE Transactions on Neural Networks, vol. 5, no. 6, pp. 989–993, 1994. DOI: 10.1109/72.329697

[13] MacKay, David J. C. "Bayesian interpolation." Neural computation. vol. 4, no. 3, pp. 415–447, 1992. DOI: 10.1162/neco.1992.4.3.415

[14] Foresee, F. Dan, and Martin T. Hagan. "Gauss-Newton

approximation to Bayesian learning." Proceedings of the International Joint Conference on Neural Networks, vol. 3, no. 3, pp. 1930-1935, 1997. DOI: 10.1109/ICNN.1997.614194

[15] Moller M.F., “A scaled conjugate gradient algorithm for fast supervised learning” Neural Networks, Elsevier, vol. 6, no. 4, pp. 525–533, 1993. DOI: 10.1016/S0893-6080(05)80056-5.