Efficient Design of Neural Networks for Solving Third Order Partial Differential Equations

Article *in* Journal of Physics Conference Series · May 2020

DOI: 10.1088/1742-6596/1530/1/012067

2 authors:

Luma Naji Mohammed Tawfiq
University of Baghdad
138 PUBLICATIONS **359** CITATIONS
SEE PROFILE

Muna H. Ali
University of Anbar
8 PUBLICATIONS **27** CITATIONS
SEE PROFILE

Journal of Physics: Conference Series

Efficient Design of Neural Networks for Solving Third Order Partial Differential Equations

Efficient Design of Neural Networks for Solving Third Order Partial Differential Equations

Luma N M Tawfiq[1] , Muna H Ali[2]

[1]Department of Mathematics, College of Education for pure science Ibn Al-Haitham, University of Baghdad, , Baghdad, Iraq
E-mail: luma.n.m@ihcoedu.uobaghdad.edu.iq
[2]Department of Mathematics ،College of Education for Pure Sciences, University Of Anbar, Anbar, Iraq
E-mail: eps.munahussain.ali@uoanbar.edu.iq

Abstract. The aim of the research is design efficient neural networks to solve important type of partial differential equations represent many important models. The efficiency of design based on chooses a suitable activation function, suitable training algorithms with study how to increase the speed of this algorithm. Then we used suggested design to solve third order PDEs.

1. Introduction

A neural network (ANN) is a mathematical branch of the field of "artificial intelligence". Artificial intelligence attempts to model complex processes using learning means rather than predefined or sequential algorithms. Problems like pattern recognition, many forms of graphical analysis, and analyses of complex data interactions are best performed with learning-type methods [1]. Artificial Neural networks effectively are a mapping which creates a function of several variables from a number of sums and products of functions of one variable. It may contain several layers which can be used for calculations, corresponding to the layers of the ANN's and it may modify the parameters in the resulting form of the approximation by a learning or training procedure which passes through the ANN's repeatedly or which moves forward and back ward through the ANN's.
There are many application for ANNs in different branches for more details see ([2]-[7]) here we use this technique to solve type of differential equations that cannot be solved by other methods or solved but here easy implemented, faster, more accuracy. The idea of this research based on many important results got from previous researches such ([8]-[12]).

2. SUGGESTED DESIGN of ANN

The best architecture can be designed depends on the problems to be represented and its type, how the layers are connected to each other, how many layers a ANN has, number of neurons in each layer, and activation function for each layer's.

Here we suggest 3 layers feed forward (FFNN) feeding network. The 1st layer consist input neuron required to feeding the ANN by the data. Then hidden layer consist processing neurons and 3rd layer is output layer. Each layer connected to others by interconnected as synaptic weights, are used to store the data and knowledge. These weights taken the initial value randomly then adjusted in training process by choosing suitable training algorithm. Most of ANNs takes back-propagation training algorithm based on gradient descent (as in the Widrow-Hoff learning rule), which use descent,

downhill, direction. But this is often too slow for practical problems [13], then we can use Newton's or quasi-Newton's method to perform faster convergence, but the implementation of such methods is cumbersome and this is due to the need for computation of the Hessian matrix or its inverse [14]. However, we will consider the CG algorithm which is a search performed along conjugate directions and such process, generally, converge faster than the gradient descent algorithms (GD). Here, we suggest high performance algorithm, which can converge from ten to one hundred times faster than the algorithm of GD and GD with momentum.

Thus, we suggest the following algorithm, the initial iteration depend on the negative of gradient. But the others depend on product search direction with learning rate, which can be express formally as follows:

$$W_{k+1} = W_k + \alpha_k \rho_k \qquad (1)$$

Where; W_{k+1}: is the weight in the k+1 iteration, ρ_k : is the search direction in the k iteration, α_k : is the learning rate in the k iteration.

Suggested update Algorithm

Given x_0;

Step $\rho_0 \leftarrow -g_0$, $k \leftarrow 0$, $r_k \leftarrow XW_k - Y$

While $r_k \neq 0$

$$\alpha_k \leftarrow -\frac{r_k^T \rho_k}{\rho_k^T W \rho_k};$$

$$W_{k+1} \leftarrow W_k + \alpha_k \rho_k;$$

$$\beta_{k+1} \leftarrow \frac{g_{k+1} \rho_{k-1}^T W}{\rho_k^T W \rho_k};$$

$$\rho_{k+1} \leftarrow -g_{k+1} + \beta_{k+1} \rho_k;$$

$$k \leftarrow k+1;$$

end (while)

Most line search algorithms require ρk to be a descent direction because this property guarantees that the performance function can be reduced along this direction. Moreover, we choose the search direction has the form:

$$\rho_{0=-g_0}, \quad \rho_k = -g_k + \beta_k \rho_{k-1}; \qquad (2)$$

where gk is the gradient of the performance function with respect to the weights at the iteration k.

3. Activation function

For each hidden neurons in the hidden layer we choose sigmoid activation function. The term sigmoidal is used for the class of activation functions atisfying $\lim_{m\to\infty} \sigma(x) = 0$ and, $\lim_{m\to\infty} \sigma(x) = 1$, also σ is continuous and monotonic on all R ([15], [16]).

So, the logistic sigmoid is often used because it is well suited to the demands of back- propagation. It is continuous and bounded function and ease to find its derivative, that is, $\sigma'(x) = \sigma(x)[1 - \sigma(x)]$ and $\sigma''(x)=\sigma(x)[1 - \sigma(x)]\ [1 -2\sigma(x)]$.

The linear activation function will be choosing for output neuron.

4. Variable Learning Rate

To increase the speed of suggested ANN we suggest rule to determine the optimal setting for the learning rate before training and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface. That is

$$\alpha_{0pt} = \frac{\left|\left(W^* - W(k)\right)^T X(k)\right|}{\|X(k)\|^2}$$

where w* is the optimal weight.

The above αopt is not practical (because w* is not known), we can use the optimal learning rate for quadratic error surface (E) as:

$$\alpha_k = \frac{E(k)^T E'(k)}{\rho_k^T E''(k)\rho_k}$$

where ρk is the search direction.

The performance of the training algorithm can be improved if during the training process we allow the learning rate to change.

ANN offers an appealing tool for optimization applications, which usually involve finding a global minimum of an energy function (error function (E)). The energy function always decreases when the state of any processing element changes. Suppose the state changed from +1 to −1 (or vice versa). The change in energy ΔE is given as:

$\Delta E = E(x^{new}) - E(x^{old})$

$$= -\frac{1}{2}\sum_{i=1}^{N} X_i^{new}\sum_{j=1}^{N} W_{ij}X_j^{new} - \sum_{i=1}^{N} X_i^{new}b_i + \frac{1}{2}\sum_{i=1}^{N} X_i^{old}\sum_{j=1}^{N} W_{ij}X_j^{old} + \sum_{i=1}^{N} X_i^{old}b_i$$

$$= -X_k^{new}\sum_{j=1}^{N} W_{kj}X_j^{new} - X_k^{new}b_k + X_k^{old}\sum_{j=1}^{N} W_{kj}X_j^{old} + X_k^{old}b_k$$

The above result hold because except for the kth processing element state xk, none of the other states have changed, that is:

$$X_i^{new} = X_i^{old}, \text{for } i \neq k$$

and use wij = wji, wii = 0, then:

$$\Delta E = (X_k^{old} - X_k^{new})\left[\sum_{j=1}^{N} W_{kj}X_k^{old} + b_k\right]$$

(1) If X_k has changed from $X_k^{old} = -1$ to $X_k^{new} = +1$, then, $X_k^{old} - X_k^{new} = -2$ and $\sum_{j=1}^{N} W_{kj}X_k^{old} + b_k$ must be positive. Thus ΔE will be negative.

(2) If X_k has changed from $X_k^{old} = +1$ to $X_k^{new} = -1$,, then, $X_k^{old} - X_k^{new} = 2$ and $\sum_{j=1}^{N} W_{kj}X_k^{old} + b_k$ must be positive. Thus ΔE again negative.

(3) If X_k has not change, then, $X_k^{old} - X_k^{new} = 0$ Then ΔE = 0. Thus ΔE ≤ 0, whenever a processing element changes state.

5. Application problem

Consider the following 3rd order PDEs:

$(u_{tt} - u_{xx})t + u_{tt} - u_{xx} = 0,\quad 0 < t < 2,\ 0 < x < 1,$

With ICs: $u(x, 0) = \exp(-\frac{(x-0.5)^2}{0.02})$, $u_t(x, 0) = 0$, $u_{tt}(x, 0) = (\frac{(x-0.5)^2}{0.01^2} - 100)\, u(x, 0)$,

Here we take 2 input neurons in the input layer for x and t, also, 15 sigmoid neurons in the hidden layers, with 1 output linear neuron for y. the accuracy of practical results calculated as: Error between exact ($\bar{u}$) and neural solutions (u):

$$E = \max_{i,k} |u(x_i, t_k) - \bar{u}(x_i, t_k)|$$
$$= 6.22 \times 10{-}7$$

'Figure 1' present Simulink suggested design. 'Figure 2' present the results after testing but 'Figure 3' present the results after training suggested design. 'Figure 4' present the deviation E = Δu(x, t) of the obtained neural solution at the 100 grid points that were selected for training. 'Table 1' illustrates the weights of ANN in initial training and after training.

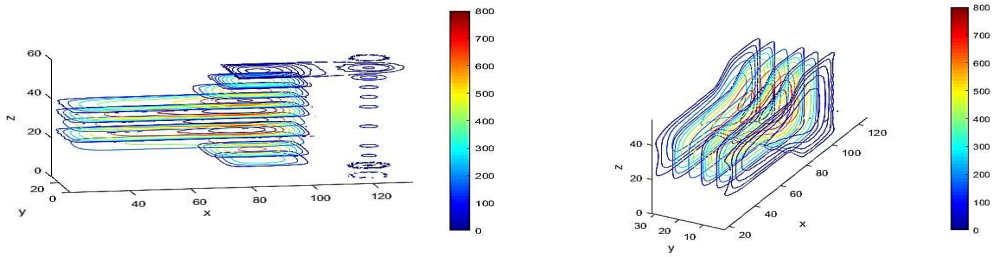

Figure 1. suggested design

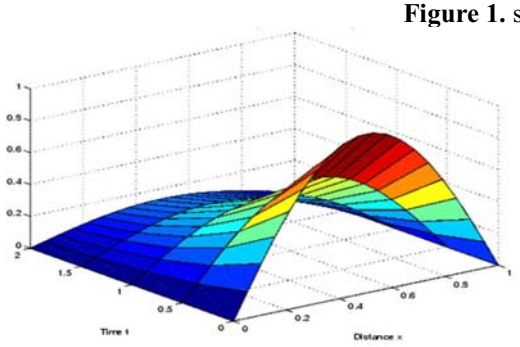

Figure 2. Results of FFNN after testing

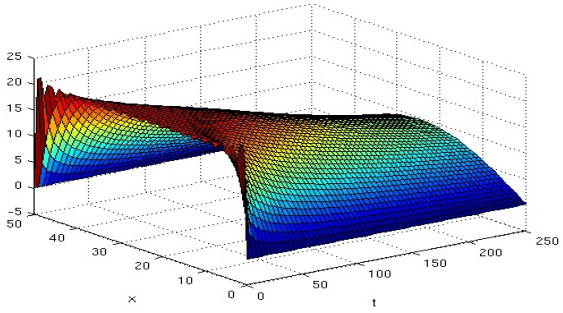

Figure 3. Results of FFNN after training

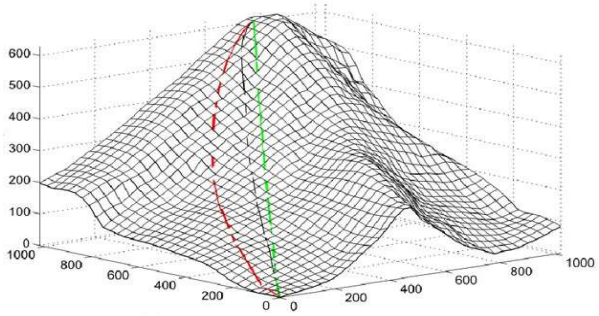

Figure 4. Accuracy of suggested desig

Table 1. weights of ANN in initial (W_0, V_0) and after (W_{opt}, V_{opt}) training

$$W_0 = \begin{pmatrix} 0.24144 & 0.47234 \\ 1.88921 & -1.85291 \\ 0.88459 & 1.53971 \\ -1.77856 & 1.45301 \\ -1.87231 & -1.10013 \\ 1.59298 & -1.90811 \\ -1.88869 & -1.67431 \\ 1.79345 & 1.62125 \\ -1.45391 & 1.34212 \\ 1.29801 & -2.50801 \\ 1.7615 & 2.56012 \\ -1.89021 & -1.40926 \\ 1.43100 & 0.83421 \\ -1.93412 & -1.76451 \\ -1.83512 & 1.86711 \end{pmatrix}, V_0 = \begin{pmatrix} 0.49187 \\ 0.99991 \\ -1.30891 \\ 0.41723 \\ -1.4783 \\ -1.94768 \\ 1.78991 \\ 1.09671 \\ 1.46172 \\ -1.09876 \\ -1.99231 \\ -1.00015 \\ 1.54891 \\ 1.78453 \\ 0.40912 \end{pmatrix}, W_{opt} = \begin{pmatrix} 0.26946 & 0.70201 \\ 0.10893 & -0.41089 \\ -0.41721 & -0.79567 \\ -0.29861 & 0.74301 \\ 0.70071 & 0.82391 \\ 0.56251 & -0.201987 \\ -0.780091 & 0.19811 \\ 0.08734 & -0.69871 \\ -0.70111 & 0.32865 \\ 0.68834 & 0.59510 \\ 0.80124 & -0.62019 \\ 0.71563 & 0.78961 \\ -0.00564 & 0.74321 \\ -0.65238 & -0.69331 \\ 0.15781 & -0.79340 \end{pmatrix}, V_{opt} = \begin{pmatrix} 0.01011 \\ -0.68420 \\ 0.67452 \\ 0.24921 \\ -0.64098 \\ -0.59761 \\ 0.62085 \\ 0.23018 \\ 0.50178 \\ 0.72643 \\ -0.10926 \\ 0.67054 \\ 0.32091 \\ -0.10756 \\ 0.20197 \end{pmatrix},$$

Now, we show the convergence of neural results to exact solution. The term convergence refers to the adjust behavior of the weights during training process, which lead to minimization of error between the desired and actual outputs of ANN. we call it the performance function. Therefore, we give the following convergence theorem.

Theorem 1

The following three statements are equivalent:

(a) W is convergent ;

(b) $\lim_{m\to\infty} \|W^m\| = 0$;

(c) $\rho(W) < 1$.

Proof

We first show that (a) and (b) are equivalent.

Since ||.|| is continuous, and ||O|| = 0, then (a) implies (b). But if (b) holds (for some norm), then there exists an positive constants M such that:

$$\|W^m\|_\infty \le M\|W^m\|_\infty \to 0$$

Hence, (a) holds.

Next we show that (b) and (c) are equivalent.

Now, by the fact that $\lambda(Wm) = \lambda m(W)$, we have:

$$\|W^m\| \ge \rho(W^m) = \rho^m(W)$$

So that (b) implies (c). On the other hand, if (c) holds, then we can find an $\varepsilon > 0$ and a natural norm, such that:

$$\|W\| \le \rho(W) + \varepsilon \equiv \theta < 1$$

Now, if we use the inequalty ||AB|| ≤ ||A|| ||B||, we get:

$$\|W^m\| \le (\|W\|)^m \le \theta^m$$

So that $\lim_{m\to\infty} \|W^m\| = 0$ (since $\theta < 1$) and hence (b) holds.

A test for convergence of weight matrices which is frequently used can follow from the following corollary:

Corollary 1

W is convergent to optimal weight if for some matrix norm, ||W|| < 1.

Proof:

By property of matrix norms ||AB|| < ||A|| ||B||, we have

$$\|W^m\| \le \|W\|^m$$

then $\lim_{m\to\infty} \|W\|^m = 0$, since ||W|| < 1

but $\lim_{m\to\infty} \|W^m\| \le \lim_{m\to\infty} \|W\|^m = 0$, then $\lim_{m\to\infty} \|W^m\| = 0$

So that condition (b) of theorem 1 holds.

6. SENSITIVITY of the LEAST SQUARES SOLUTIONS to the NUMER of LAYERS

The stability of the numerical solution of an over determined system, XW=Y depends on the condition number of the matrix and the choice of the activation functions. The most used function is ridge function and thus we need only to discuss the number of layers to achieve an accurate solution without having an ill-condition matrix.

Suppose the vector $\underline{X}$ minimizes $\left\|X\underline{W} - \underline{Y}\right\|^2$, due to round off error perturbation, the vector $\underline{Y}$ minimizes $\left\|(X + \delta X)W - (\underline{Y} - \delta\underline{Y})\right\|^2$, where $\|\delta X\|^2 = \varepsilon\|X\|^2$, $\left\|\delta\underline{Y}\right\|^2 = \sigma\left\|\underline{Y}\right\|^2$ and ε is the machine precession. Rounding errors may actually change the rank of the matrix, but we always assume that the rank of X is the same as the rank of (X + δX).

Golub in (1969) shows that, if $r = X\underline{W} - \underline{Y}$, then we have roughly:

$$\left\|\underline{Y} - \underline{X}\right\|^2 \le \varepsilon k(X)(\|X\|^2 + \|Y\|^2) + \varepsilon\|r\|^2 k^2(x) + O(\varepsilon^2).$$

We see that the upper bound includes terms $\varepsilon\|\underline{r}\|_2\ k^2(X)$ which is a measure of the inherent sensitivity of the least square problem; nevertheless we avoids some of the ill effects inherent in the use of the normal equations. Moreover, in practice and to be on the safe side, in the absence of other information, one should normalize each column of the neural matrix so that its lengths are one.

The above bound shows that if the norm of the residual vector $\underline{r}$ is small it is not necessarily true that accurate solution has been determined, at the local minima $\underline{r}$ can be small, and as the condition number increases we may get an inaccurate least squares solution.

To show the dependence of the k(X) on the number of layers of the ANN's, i.e., on the matrix arisen from ANN's, and the effect of these on residual, we give the following theorem:

Theorem 2

Let $x \in R^{m\times n}$, $\underline{V} \in R^{N\times 1}$, $\underline{y} \in R^{M\times 1}$ and $a \in R$, if

$\overline{X} = \begin{pmatrix} X \\ \underline{V}^T \end{pmatrix}$ and $\overline{\underline{y}} = \begin{pmatrix} \underline{y} \\ a \end{pmatrix}$, then

1. $k(\overline{X}) \le k(X) \times \left(1 + \frac{\|\underline{V}\|^2}{\|X\|^2}\right)$

2. $\underset{\underline{W}}{Min} \left\|\overline{X}\underline{W} - \overline{\underline{y}}\right\| \ge \underset{\underline{W}}{Min} \left\|X\underline{W} - \underline{y}\right\|_2$

Proof:

1. Suppose the SVD of $\overline{X}$ and X are $\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_N$ and $\bar{\sigma}_1 \ge \bar{\sigma}_2 \ge \ldots \ge \bar{\sigma}_N$ respectively.

$\left\|\overline{X}\underline{W}\right\|_2^2 = \left\|X\underline{W}\right\|^2 + (\underline{V}^T\underline{X})^2$

From the above equality and the definition of SVD we have

$\bar{\sigma}_N^2 \ge \sigma_N^2$ and $\bar{\sigma}_1^2 \le \sigma_1^2 + \left\|\underline{V}\right\|^2$ then:

$K^2(\overline{X}) = \frac{\bar{\sigma}_1^2}{\bar{\sigma}_N^2} \le \frac{\sigma_1^2}{\sigma_N^2}\left(1 + \frac{\|\underline{V}\|_2^2}{\sigma_1^2}\right) = K^2(X)\left(1 + \frac{\|\underline{V}\|_2^2}{\|X\|_2^2}\right)$.

Thus $K(\overline{X}) \le K(X)\left(1 + \frac{\|\underline{V}\|_2^2}{\|X\|_2^2}\right)^{1/2}$

2. Let $\overline{\underline{W}}$ minimize $\left\|\overline{X}\underline{W} - \overline{\underline{y}}\right\|^2$

Then $\underset{\underline{W}}{Min} \left\|\overline{X}\underline{W} - \overline{\underline{y}}\right\|_2^2 = \left\|\overline{\underline{X}}\,\underline{W} - \overline{\underline{y}}\right\|_2^2 = \left\|\begin{pmatrix} X\overline{\underline{W}} - \underline{y} \\ \underline{V}^T\overline{\underline{W}} - a \end{pmatrix}\right\|_2^2 \ge \left\|X\,\overline{\underline{W}} - \underline{y}\right\|_2^2 \ge \underset{\underline{W}}{Min} \left\|X\,\underline{W} - \underline{y}\right\|_2^2$

Thus $\underset{\underline{W}}{Min} \left\|\overline{X}\underline{W} - \overline{\underline{y}}\right\|^2 \ge \underset{\underline{W}}{Min} \left\|X\underline{W} - \underline{y}\right\|^2$, thus the results hold.

Note that, since $\|\underline{W}\|^2 / \|X\|^2$ is almost of order unity, then from the above theorem one can conclude that as we fix the number of activation functions, number of column N and increase the number of layers of the ANN, the condition number K(X) of our system does not get worse, but the residual is worse than the previous value and thus one should be careful on choosing the number of layers to get an accurate numerical solution and thus 3 layers is sufficient to approximate any continuous function to any required accuracy.

7. Conclusions

In this paper, we designed efficient ANN's by choosing suitable: training algorithm, learning rate and activation function. The use of ANN's based on forward models, for some problems, offers several advantages over numerical models in terms of both implementation and calculation of the gradient also in the updates of the weights and overall computational cost. Our numerical results shows that, in approximation of the numerical solution of 3rd order PDE using 3 layers ANN gives better accuracy than using other numerical methods for 3rd order problems.

References

[1] Alexander Poularikas, 2002 *Handbook of Neural Network Signal Processing*, CRC Press LLC,.

[2] Baymani, M., Kerayechian, A. and Effati, S., 2010 *Artificial Neural Networks Approach For Solving Stokes Problem, Applied Mathematics*, 1(04): 288,.
[3] Mall, S. and Chakraverty, S., *2013Comparison of artificial neural network architecture in solving ordinary differential equations, Advances in Artificial Neural Systems*, 2013: 12,.
[4] Tawfiq, L. N.M and Ahmed, S. A. 2016 Using Artificial Neural Network Technique to Estimation the Concentration of Copper for Contaminated Soils, *Global Journal of Engineering Science And Researches*, 3(1): 77-80,.
[5] Tawfiq, L. N. M. and Ghazi, F. F., 2015 *Using Artificial Neural Network Technique For The Estimation of CD Concentration in Contaminated Soils*, International Journal of Innovations in Scientific Engineering, Volume 11, No. 2, pp: 58-68.
[6] Ghazi, F. F., 2018 *Estimation of Heavy Metals Contamination in soil of Zaafaraniya City Using The Neural Network*, Journal of physics conference series, 1003, 12058.
[7] Tawfiq, L. N. M. and Hussein, W. R. 2016 *Design Suitable Neural Network for Processing Face Recognition*, Global Journal Of Engineering Science And Researches, Volume 3, Issue 3, , 58-64.
[8] Tawfiq, L. N. M., 2005 *On Training of Artificial Neural Networks*, Al-Fatih journal, 1(23): 130-139,.
[9] Tawfiq, L. N. M and Oraibi, Y. A., 2017 *Fast Training Algorithms for Feed Forward Neural Networks*, Ibn Al-Haitham Journal for Pure and Applied Science, 26 (1): 275-280.
[10] Tawfiq, L.N.M., and Naoum, R.S., 2007 *Density and approximation by using feed forward Artificial neural*, Ibn Al-Haitham Journal for Pure & Applied Sciences, 20, 1, 67-81.
[11] Tawfiq, L.N.M., 2016 *Using Collocation Neural Network to Solve Eignvalue Problems*, MJ Journal on Applied Mathematics, 1, 1, 1-8.
[12] Tawfiq, L.N.M., and Eqhaar, Q.H., 2009 *On Multilayer Neural Networks and Its Application for Approximation Problem*, 3rd scientific conference of the College of Science. University of Baghdad.
[13] Alexander I. Galushkin, 2007 *Neural Networks Theory*, Springer, Berlin.
[14] L.N. M. Tawfiq and O.M. Salih, 2019 *Design Suitable Feed Forward Neural Network to Solve Troesch's Problem*, Sci.Int.(Lahore), 31 (1), 41-48.
[15] L.N.M. Tawfiq and O.M. Salih, 2019 *Design neural network based upon decomposition approach for solving reaction diffusion equation*, Journal of Physics: Conf. Series, IOP Publishing, 1234 ,012104, 1-7.
[16] Ali MH, Tawfiq LNM, Thirthar AA, 2019 Designing Coupled Feed Forward Neural Network to Solve Fourth Order Singular Boundary Value Problem. *Revista Aus* 26(2) pp.140–6. DOI: 10.4206/aus.2019.n26.2.20.

.