Solution of Definite Integrals using Functional Link Artificial Neural Networks

Satyasaran Changdar[1] and Snehangshu Bhattacharjee[1]

[1]Department of Information Technology, Institute of Engineering and Management, Salt Lake, Kolkata

Abstract

This paper discusses a new method to solve definite integrals using feedforward artificial neural networks. The objective is to build a neural network that would be a novel alternative to pre-established numerical methods and with the help of a learning algorithm, be able to solve definite integrals, by minimising a well constructed error function. The proposed algorithm, with respect to existing numerical methods, is effective and precise and well-suited for purposes which require integration of higher order polynomials. The observations have been recorded and illustrated in tabular and graphical form.

1 Introduction

Through centuries, men have been captivated by the power of the human brain. Our very ability to capture, comprehend, analyse and draw conclusions from seemingly innocent pieces of information, is a marvel which deserves admiration. Only recently has science been able to gauge intelligence and its parameters, and yet it is far from fathoming the intricate machinery of sense and logic completely.

With several scientific attempts being made to study activities and patterns of the brain since the nineteenth century, it was only in 1943 that McCulloch and Pitts introduced a simple model of a neural network [16], which had weighted inputs summed up to give the output. With the advent of computers in the 1950s, it was finally possible to implement the theories. The concept of a "thinking machine" has been much argued over, and the Dartmouth Project in 1956 followed by Frank Rosenblatt's work in development of a Perceptron [20] was a major impetus to the idea. Mark I Perceptron was built in the year 1958, at Cornell University, with its main function being character recognition. Soon after, Bernard Widrow and Marcian Hoff of Stanford successfully launched neural network models ADALINE and MADALINE [23] for real-world purposes.

In 1982, John Hopfield of Caltech, with his charismatic paper presented to the National Academy of Sciences [6], started rekindling interest in what was, by far, an overlooked field in science. The US-Japan Joint Conference on Cooperative/Competitive Neural Networks propelled efforts considerably and by 1985, the Neural Networks for Computing conference was held, which was to become an annual affair. Also, IEEE's first International Conference on Neural Networks became a success in 1987. The last decade of the twentieth century was crucial in bringing the study of neural networks to the fore with many sectors in innovation and research recognising and targeting it. ANN is being extensively used as a tool in various disciplines for applications which include modelling complex mathematical structures and obtaining solutions to mathematical problems. Mathematical models, with a framework of ANN, can provide sustainable and effective solutions in commercially applicable fields [7] [2] [3] [4]. Functions such as those discussed below are frequently encountered in real-life problems of engineering. Calculus is extensively used in manufacturing and other core sectors for applications like improving architecture and determining the infrastructure of bridges. It plays an important role in applications that deal with massive data and require precision and calculus equations to make predictions like in meteorology and in the finance sector. Even in the case of search engines which demand a number of variables to be taken into account, calculus comes into play. The next logical step would be incorporation of neural networks in practical applications in the near future.

In the present paper, we attempt to discuss a way of solving Definite Integrals by applying a direct, clinical and robust approach using the effectiveness of neural networks instead of more cumbersome numerical methods involving analytical calculations.

2 Motivation

Determining an Indefinite Integral refers to the process of determining a function from its derivative $f(x)$ [22]. The functions that could possibly have $f(x)$ as a derivative are the so-called antiderivatives of $f(x)$, and the formula that leads to them is called the indefinite integral of $f(x)$.
Mathematically:

$$\int f(x)dx = g(x) + C \tag{1}$$

In this equation, $g(x)$ is one of the antiderivatives of $f(x)$, and as the constant C changes, the equation gives a family of antiderivatives.
Let $g(x)$ be the antiderivative of the continuous function $f(x)$ defined on [a,b]. Then the definite integral of $f(x)$ over [a,b] is denoted by

$$\int_a^b f(x)dx = g(b) - g(a) \tag{2}$$

In the process of solving the above equation, we come across a number of problems:
1. The integrand $f(x)$ may be known only at certain points.
2. A formula for the integrand $f(x)$ may be known, but it may be difficult or impossible to find an antiderivative, that is, an elementary function.
3. It may be possible to find an antiderivative symbolically, but it may be easier to compute a numerical approximation than to compute the antiderivative.
In the proposed method, we have discussed a solution in accordance to the above points (2) and (3).

Usually, in cases where analytical solutions may not be arrived at easily, we rely on numerical methods for solving the equation. This again leads to specialised methods, specific to the purpose. A more appropriate approach would be to design a solution that aptly fits in the places of these varied methods and gives a decently precise and accurate solution. This leads us to consider Artificial Neural Networks as an alternative [12] [13] [14]. The efficiency of Artificial Neural Networks is due to its nonlinear learning capability [8]. This powerful tool finds application in functional approximation and optimisation. Due to application of FLANN (Functional Link Artificial Neural Network) [15] [21], we no longer need hidden layers. This makes the learning algorithm simpler. The FLANN architecture effectively expands the input vector giving more flexibility and decision-making capability to the network, due to a more expansive input space.

3   Mathematical modelling

We have already established that the definite integral of a function can be calculated as shown in equation 2

$$\int_a^b f(x)dx = g(b) - g(a) \tag{3}$$

In the proposed model we would replace the $g(b)$ and $g(a)$ by $N(b)$ and $N(a)$ respectively. Here, $N(x)$ represents a function of x which is obtained as the output of the neural network. Therefore our former equation reduces to

$$\int_a^b f(x)dx = N(b) - N(a) \tag{4}$$

3.1   Model of the Neural Network

Output of the Neural Network

$$N(x) = w_1\Phi_1(x) + w_2\Phi_2(x) + w_3\Phi_3(x) + ... + w_n\Phi_n(x) \tag{5}$$

In general, a model of a neural network comprises simple computational units (nodes) which may be input nodes, output nodes or hidden nodes. The units, with appropriate interconnections, is represented by a graph with the nodes as the vertices and each edge between two neurons is labelled with a synaptic weight. The network may or may not have multiple layers, depending on the complexity of problems. As layers and number of neurons increase, the training process becomes more complicated and requires intense computation. An alternative to this cumbersome approach is the FLANN architecture [15] [18] [17]. In our present study, we have employed the classification method of Functional Link Artificial Neural Network (FLANN). FLANN architecture uses a single layer feed-forward network, which can be considered as an alternative to multi-layer neural network for solving problems. The drawback with multi-layer neural network is its linear nature which becomes a hindrance for mapping complex non-linear problems. Since the practical realm of data deals mostly with

non-linearity, therefore, choosing multi-layer networks for this is not wise. Yet again, employing a multi-layer neural network for this purpose further complicates the task. But in FLANN architecture, the input vector is functionally expanded to overcome such problems, such that even though it is a single layer neural network, it is well-suited to the task. The model which has been used is shown in Fig.1.
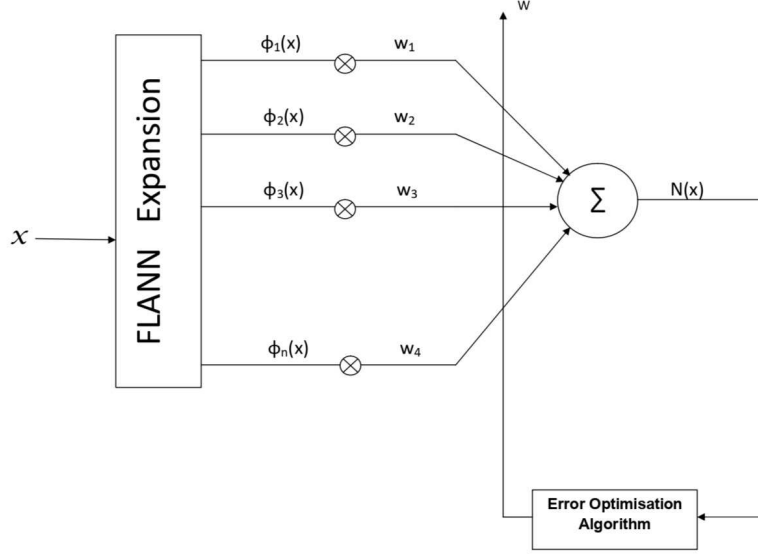

Figure 1: Schematic diagram of the neural network model

If the input vector is x, then after expansion, the input pattern is a set of elements :

$$\Phi_1(x), \Phi_2(x), ...\Phi_n(x) \tag{6}$$

which are appropriate functions that will serve in the approximation process. The set of functions for expansion may be set in accordance to the complexity of the problem and the restraints posed on it.

3.2 Modelling the Error Function

As discussed above in equation 4

$$\int_a^b f(x)dx = N(b) - N(a) \tag{7}$$

Our objective is to build a function $N(x)$ using the neural network [10] [11] [24] [9] which will approximate the function $g(x)$ upto a desired precision, so that this function $N(x)$ can be later on employed to compute the values of the integral at a and b. Therefore,

$$g(x) \approx N(x) \tag{8}$$

4

In the ideal scenario, $g(x)$ equals $N(x)$

$$g(x) - N(x) = 0 \qquad (9)$$

Differentiating equation 9 with respect to x gives,

$$g'(x) - N'(x) = 0 \qquad (10)$$

or,

$$f(x) - N'(x) = 0 \qquad (11)$$

where,

$$N(x) = w_1\Phi_1(x) + w_2\Phi_2(x) + w_3\Phi_3(x) + ... + w_n\Phi_n(x) \qquad (12)$$
$$N'(x) = w_1\Phi'_1(x) + w_2\Phi'_2(x) + w_3\Phi'_3(x) + ... + w_n\Phi'_n(x) \qquad (13)$$

Therefore equation 11 becomes

$$f(x) - (w_1\Phi'_1(x) + w_2\Phi'_2(x) + w_3\Phi'_3(x) + ... + w_n\Phi'_n(x)) = 0 \qquad (14)$$

The error function can now be summarised as:

$$E(x) = \frac{1}{2}\sum_{i=1}^{k}\left(f(x_i) - N'(x_i)\right)^2 \qquad (15)$$

where x_i refers to each training sample point; k points belonging to the interval (a,b) have been considered.

3.3   Minimisation of Error

In this study, we have already constructed an error function, minimisation thereof would lead to an appropriate solution to the given definite integral. The error function uses a parameter which corresponds to the result derived from the neural network. This parameter is modified with every modification of the network (modification here corresponds to updation of the weights in the network). The error optimisation algorithm used is gradient descent algorithm [1] [19]. This algorithm uses first-order iterative optimisation to find the minimum of a function. The algorithm is designed to proceed in steps proportional to the gradient of the function, in the negative direction of the current point. In the gradient descent algorithm, initially, the gradient of the function is to be calculated. This is done by differentiating the error function with respect to each of the weights one by one. These are then used to update the weights. The process leads to reduction of the error gradually until we reach a desired precision.

Differentiating 15 with respect to each weight:

$$\frac{dE}{dw_1} = -\sum_{i=1}^{k} e_i\Phi'_1(x_i) \qquad (16)$$

$$\frac{dE}{dw_2} = -\sum_{i=1}^{k} e_i\Phi'_2(x_i) \qquad (17)$$

$$\frac{dE}{dw_3} = -\sum_{i=1}^{k} e_i\Phi'_3(x_i) \qquad (18)$$

and likewise, till

$$\frac{dE}{dw_n} = -\sum_{i=1}^{k} e_i \Phi'_n(x_i) \tag{19}$$

where,

$$e_i = f(x_i) - N'(x_i) \tag{20}$$

The weights are then updated as:

$$w_1^+ = w_1 - \eta \frac{dE}{dw_1} \tag{21}$$

$$w_2^+ = w_2 - \eta \frac{dE}{dw_2} \tag{22}$$

$$w_3^+ = w_3 - \eta \frac{dE}{dw_3} \tag{23}$$

and likewise, till

$$w_n^+ = w_n - \eta \frac{dE}{dw_n} \tag{24}$$

where, η is the *learning rate*.

4   Results and observations

Let us work out an example using the method discussed above. Consider the function,

$$f(x) = \sqrt{1 + (x^2)} \tag{25}$$

We would like to integrate the function from 0 to 2, i.e

$$\int_0^2 \sqrt{1 + (x^2)} dx \tag{26}$$

Using Maclaurin Series, a function $g(x)$ can be expanded as,

$$g(x) = \sum_{n=0}^{\infty} g^{(n)}(0) \frac{x^n}{n!} \tag{27}$$

where $g^{(n)}$ is the nth derivative of $g(x)$.
Expanding the concerned function using Maclaurin's Series,

$$\sqrt{(1 + (x^2))} = 1 + \frac{x^2}{2} - \frac{x^4}{8} + \frac{x^6}{16} - ..... \tag{28}$$

Approximating equation 28 using the neural network would require a good choice of the functional links $\phi_1, \phi_2, \phi_3, ...\phi_n$
In accordance to the Maclaurin Series expansion of $f(x)$, which is a polynomial function, we choose

$$\phi_1(x) = x \tag{29}$$

6

$$\phi_2(x) = x^2 \tag{30}$$

$$\phi_3(x) = x^3 \tag{31}$$

and likewise, till

$$\phi_n(x) = x^n \tag{32}$$

Therefore, equation 5 can be reconstructed as

$$N(x) = w_1 x + w_2 x^2 + w_3 x^3 + \ldots\ldots + w_n x^n \tag{33}$$

Similarly, equation 13 becomes

$$N'(x) = w_1 + 2w_2 x + 3w_3 x^2 + \ldots\ldots + nw_n x^{n-1} \tag{34}$$

The error function for the problem under consideration, becomes

$$E(x) = \frac{1}{2} \sum_{i=1}^{k} \left(f(x) - (w_1 + 2w_2 x + 3w_3 x^2 + .. + nw_n x^{n-1}) \right)^2 \tag{35}$$

The objective is to optimise this error function for which we initialise the network with random weights, a suitable learning rate η is chosen, and with every iteration, the weights are updated using the gradient descent algorithm, as follows:

$$w_1^+ = w_1 - \eta \frac{dE}{dw_1} \tag{36}$$

$$w_1^+ = w_1 - \eta \sum_{i=1}^{k} -e_i \tag{37}$$

$$w_2^+ = w_2 - \eta \frac{dE}{dw_2} \tag{38}$$

$$w_2^+ = w_2 - \eta \sum_{i=1}^{k} -e_i(2x_i) \tag{39}$$

$$w_3^+ = w_3 - \eta \frac{dE}{dw_3} \tag{40}$$

$$w_3^+ = w_3 - \eta \sum_{i=1}^{k} -e_i(3x_i{}^2) \tag{41}$$

and likewise, till

$$w_n^+ = w_n - \eta \frac{dE}{dw_n} \tag{42}$$

$$w_n^+ = w_n - \eta \sum_{i=1}^{k} -e_i(nx_i{}^{n-1}) \tag{43}$$

The above procedure is repeated till the error is minimised to a desired precision. Finally, the weights are locked at the most favourable magnitudes (where the error function is minimised). These weights, when now incorporated in the network (i.e. $N(x)$ which has been approximated to $g(x)$, the assumed anti-derivative of $f(x)$), provides us the result of the integral in any interval within [0,2], i.e, the domain in which

the network has been trained. The results obtained from this neural network versus
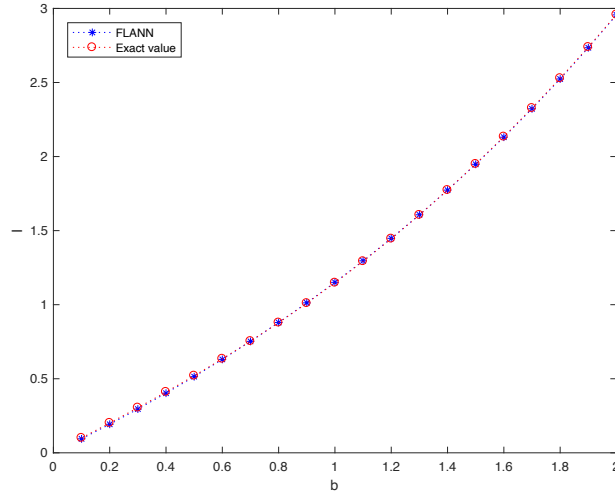the exact values calculated analytically, are illustrated in Fig.2

Figure 2: Graph comparing exact value with observed value using FLANN for
$I = \int_0^b f(x)dx$ where $f(x) = \sqrt{1 + x^2}$, b∈(0,2]

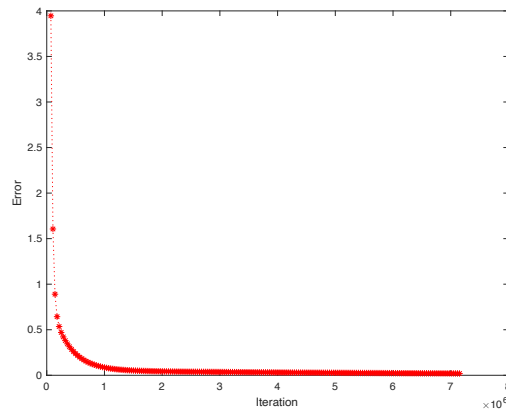On analysing the change in error with the number of iterations, we obtain Fig.3

Figure 3: Error vs. Iteration while integrating $\sqrt{1 + x^2}$

Similarly, the proposed algorithm is applied to integrate another function:

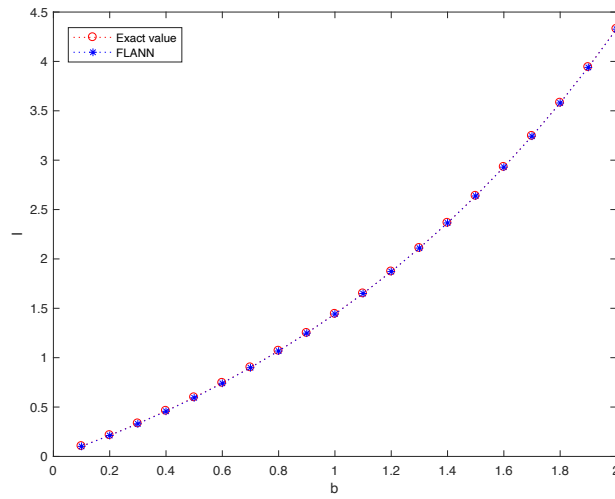$$\int_0^2 2^x dx \tag{44}$$

The Fig.4 is obtained in this case.

Figure 4: Graph comparing exact value with observed value using FLANN for $I = \int_0^b f(x)dx$ where $f(x) = 2^x$, b$\in$(0,2]

In order to illustrate the feasibility and efficiency of the proposed method, a few examples are selected, the method is applied to them, and the results are recorded in Fig.5. The given functions are integrated over [0,2].

Given Function f(x)	x-1	x^2	$\sqrt{1+x^2}$	sin x	ln (x+1)	2^x
Exact Value	0.00	2.66	2.95	1.29	1.41	4.32
FLANN	0.00	2.66	2.95	1.29	1.41	4.32

Figure 5: Comparison of a few functions solved analytically and using the discussed model

9

5 Examples for comparison with other methods

Conventional numerical methods, when implemented in calculating integrals, are not always sufficient or accurate in approximation[5]. For instance, the Trapezoidal rule gives an error:

$$E_n^T(f) \equiv \int_a^b f(x)dx - T_n(f) \equiv -\frac{h^2(b-a)}{12}f^2(c_n)$$

for some c_n in the interval (a,b).

As is evident, the error in Trapezoidal rule increases in the following scenarios:
1. If h increases, where h is the width of each subinterval between $[a, b]$
2. If $(b - a)$ is a large value
3. If the function $f(x)$ is a polynomial of a high degree; which causes $f^2(c_n)$ to yield a large value.

Similarly, Simpson's One-third rule gives an error:

$$E^S(f) \equiv \int_a^b f(x)dx - S(f) \equiv -\frac{1}{180}(b-a)h^4 f^4(c_n)$$

where c_n lies in the interval (a,b).

The error, in this case increases for the following:
1. If h increases, where h is the width of each subinterval between $[a, b]$
2. If $(b - a)$ is a large value
3. If the function $f(x)$ is a polynomial of a very high degree, which causes $f^4(c_n)$ to yield a large value.

In the following example, we have considered the function

$$\int_a^b f(x)dx = \int_0^6 x^6 dx \tag{45}$$

Since the polynomial considered is of a high degree, therefore $f^2(c_n)$ and $f^4(c_n)$ yields a considerably larger value where c_n lies in the interval $(0, 6)$. In cases where $f(x)$ is a higher degree polynomial and $(b-a)$ is also large, both the Trapezoidal and Simpson's Rule would fail to come up with precise results.

To show the promising results of the proposed FLANN architecture, we have compared the results of the equation 45 with those obtained from FLANN, Trapezoidal method, Simpson's rule and the same calculated analytically. While training the neural network for the concerned problem, we have considered 10 training points *(i.e. k=10)* belonging to the interval (a,b). Similarly, using k points in Trapezoidal and Simpson's Rule implies that the interval $[a, b]$ gets divided into *k+1* sub-intervals, thus the width h of each subinterval becomes $\frac{b-a}{k+1}$. These results have been shown in Fig.6 and Fig.7
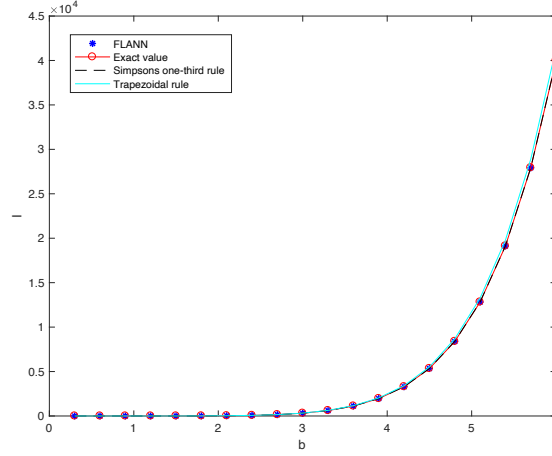
Figure 6: Graph comparing the results of the integral $I = \int_0^b f(x)dx$ where $f(x) = x^6$, b∈(0,6] calculated using FLANN, Trapezoidal method, Simpson's rule and the same calculated analytically
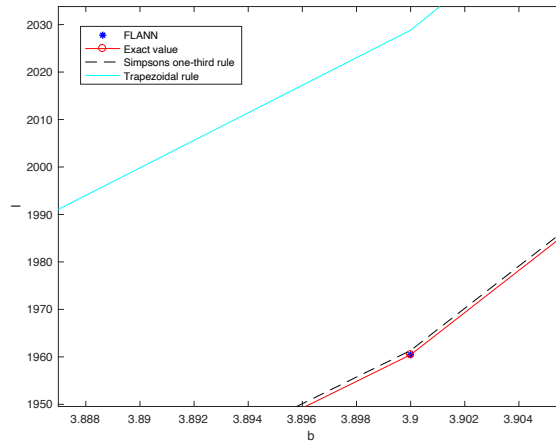

Figure 7: This graph is a magnified version of Fig.6, which clearly shows the precision attained by the proposed FLANN algorithm when compared to other techniques. Note that the exact value has completely coincided with the calculated value using the FLANN technique.

11

6 Practical Applications

While solving real life problems we come across various situations where solving a definite integral analytically becomes difficult or seems impossible. To tackle situations like these we have been using numerical integration techniques. One such definite integral which is difficult to solve analytically is the elliptic integral. Mathematics defines an elliptic integral as any function f(x) which can be expressed in the form

$$f(x) = \int_c^x R(t, \sqrt{P(t)})dt \qquad (46)$$

where R is a rational function of its two arguments, P is a polynomial of degree 3 or 4 with no repeated roots, and c is a constant. In general, integrals in this form cannot be expressed in terms of elementary functions easily. This is where numerical integration techniques proves to be of immense help and can often give precise results. Integrals like elliptic integrals which are often difficult to solve have huge applications in astrophysics and various engineering domains.

7 Conclusion

The main objective of the paper has been to establish a new method using neural networks, which is applicable in solving definite integrals in a more efficient and novel approach as compared to existing methods. We construct a neural network model based on FLANN architecture, which is used to approximate the integrand by training the neural network weights. The solved examples, compared with other existing methods, show that the algorithm is effective and feasible in all cases.
The proposed algorithm has the following advantages:
1. The FLANN architecture used successfully incorporates non-linearity in the model, which is a requirement in most practical scenarios.
2. The neural network, when trained over the interval [a,b], can easily estimate the values in another interval $[a_1,b_1]$ where a $\leq a_1,b_1 \leq$ b.
3. Even when computing integrals of higher order polynomials, the method evaluates to highly precise levels.

The successful implementation of the algorithm can depend on certain improvisations, as follows:
1. Change of optimisation method.
2. Better choice of the functional links.
3. Use of high-speed and more reliable processors.

References

[1] Pierre Baldi. Gradient descent learning algorithm overview: A general dynamical systems perspective. *IEEE Transactions on neural networks*, 6(1):182–195, 1995.

[2] Satyasaran Changdar and Soumen De. Analysis of non-linear pulsatile blood flow in artery through a generalized multiple stenosis. *Arabian Journal of Mathematics*, 5(1):51–61, 2016.

[3] Satyasaran Changdar and Soumen De. Analytical solution of mathematical model of magnetohydrodynamic blood nanofluid flowing through an inclined multiple stenosed artery. *Journal of Nanofluids*, 6(6):1198–1205, 2017.

[4] Satyasaran Changdar and Soumen De. Investigation of nanoparticle as a drug carrier suspended in a blood flowing through an inclined multiple stenosed artery. *Bionanoscience*, 8(1):166–178, 2018.

[5] Joe D Hoffman and Steven Frankel. *Numerical methods for engineers and scientists*. CRC press, 2001.

[6] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[7] ALAN Horwitz and ARYA Ebrahimpour. Engineering applications in differential and integral calculus. *International Journal of Engineering Education*, 18(1):78–88, 2002.

[8] Nicolaos Karayiannis and Anastasios N Venetsanopoulos. *Artificial neural networks: learning algorithms, performance evaluation, and applications*, volume 209. Springer Science & Business Media, 2013.

[9] Manoj Kumar and Neha Yadav. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey. *Computers & Mathematics with Applications*, 62(10):3796–3811, 2011.

[10] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[11] Isaac E Lagaris, Aristidis C Likas, and Dimitris G Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Transactions on Neural Networks*, 11(5):1041–1049, 2000.

[12] Susmita Mall and Snehashish Chakraverty. Regression-based neural network training for the solution of ordinary differential equations. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):136–149, 2013.

[13] Susmita Mall and Snehashish Chakraverty. Chebyshev neural network based model for solving lane–emden type equations. *Applied Mathematics and Computation*, 247:100–114, 2014.

[14] Susmita Mall and Snehashish Chakraverty. Application of legendre neural network for solving ordinary differential equations. *Applied Soft Computing*, 43:347–356, 2016.

[15] Susmita Mall and Snehashish Chakraverty. Single layer chebyshev neural network model for solving elliptic partial differential equations. *Neural Processing Letters*, 45(3):825–840, 2017.

[16] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[17] Santosh Kumar Nanda and Debi Prasad Tripathy. Application of functional link artificial neural network for prediction of machinery noise in opencast mines. *Advances in Fuzzy Systems*, 2011:4, 2011.

[18] Santosh Kumar Nanda, Debi Prasad Tripathy, Simanta Kumar Nayak, and Subhasis Mohapatra. Prediction of rainfall in india using artificial neural network (ann) models. *International Journal of Intelligent Systems and Applications*, 5(12):1, 2013.

[19] Arunselvan Ramaswamy and Shalabh Bhatnagar. Analysis of gradient descent methods with nondiminishing bounded errors. *IEEE Transactions on Automatic Control*, 63(5):1465–1471, 2018.

[20] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[21] Deepti Moyi Sahoo and Snehashish Chakraverty. Functional link neural network learning for response prediction of tall shear buildings with respect to earthquake data. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(1):1–10, 2018.

[22] James Stewart. *Essential calculus: Early transcendentals*. Cengage Learning, 2012.

[23] Bernard Widrow and Michael A Lehr. Artificial neural networks of the perceptron, madaline, and backpropagation family. In *Neurobionics*, pages 133–205. Elsevier, 1993.

[24] Neha Yadav, Anupam Yadav, and Manoj Kumar. Neural network methods for solving differential equations. In *An Introduction to Neural Network Methods for Differential Equations*, pages 43–100. Springer, 2015.