Design of Mexican Hat Wavelet Artificial Neural Network for Solving Lorenz Model

By

Anika Fayyaz

NATIONAL UNIVERSITY OF MODERN LANGUAGES
ISLAMABAD

November, 2025

Design of Mexican Hat Wavelet Artificial Neural Network for Solving Lorenz Model

By

Anika Fayyaz

MS-Math, National University of Modern Languages, Islamabad, 2025

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In **Mathematics**

To

FACULTY OF ENGINEERING & COMPUTING

NATIONAL UNIVERSITY OF MODERN LANGUAGES ISLAMABAD

NATIONAL UNIUVERSITY OF MODERN LANGUAGES

FACULTY OF ENGINEERING AND COMPUTING

THESIS AND DEFENSE APPROVAL FORM

The undersigned certify that they have read the following thesis, examined the defense, are satisfied with overall exam performance, and recommend the thesis to the Faculty of Engineering and Computing for acceptance.

Thesis Title: Design of Mexican Hat Wavelet Artificial Neural Network for Solving Lorenz Model

Submitted By: Anika Fayyaz **Registration #:** 93 MS/MATH/F23

Master of Science in Mathematics

Title of the Degree

Mathematics

Name of Discipline

Dr. Muhammad Rizwan	
Name of Research Supervisor	Signature of Research Supervisor
Dr.Farhad Muhammad Riaz	
Name of Research Co-Supervisor	Signature of Research Co-Supervisor
Dr. Anum Naseem	
Name of HOD	Signature of HOD
Dr. Noman Malik	
Name of Dean (FEC)	Signature of Dean (FEC)

November, 2025

AUTHOR'S DECLARATION

I Anika Fayyaz
Daughter of Muhammad Fayyaz
Discipline Mathematics

Candidate of Master of Science in Mathematics at the National University of Modern Languages do hereby declare that the thesis Design of Mexican Hat Wavelet Artificial Neural Network for Solving the Lorenz Mode submitted by me in partial fulfillment of MS degree, is my original work and has not been submitted or published earlier. I also solemnly declare that it shall not, in the future, be submitted by me for obtaining any other degree from this or any other university or institution. I also understand that if evidence of plagiarism is found in my thesis/dissertation at any stage, even after the award of a degree, the work may be canceled and the degree revoked.

Signature of Candidate

Anika Fayyaz
Name of Candidate

25, November, 2025
Date

ABSTRACT

Title: Design of Mexican Hat Wavelet Artificial Neural Network for solving Lorenz Model

Artificial Neural Networks (ANNs) have become a highly effective technique for addressing complex mathematical and scientific challenges, thanks to their remarkable learning and approximation abilities. One essential area of study within computational mathematics and dynamical systems is the Lorenz model, which consists of nonlinear differential equations. This thesis introduces a novel computational strategy that integrates artificial neural networks with a hybrid optimization framework to explore the chaotic dynamics characterized by the Lorenz model. In order to precisely approximate the solutions of the Lorenz model, a hybrid optimization technique that combines Particle Swarm Optimization (PSO) and Active Set Algorithm (ASA) is devised in this study. The model's learning efficiency is increased by using a feedforward ANN structure with a Mexican Hat activation function. The proposed ANN-PSO-ASA approach, with its ability to manage the complexity of chaotic systems, appears to be a promising tool for modeling complex dynamical behaviors. For every Lorenz model test case, 50 experimental runs were conducted in order to evaluate the suggested ANN-based framework's consistency, accuracy, and robustness. Superior convergence dependability, numerical stability, and prediction accuracy were demonstrated by the hybrid PSO-ASA optimized ANN model, which continuously outperformed other hybrid optimization techniques as well as conventional numerical methods. Statistical measures including Mean Squared Error (MSE) analysis and Mean Absolute Deviation (MAD) were used to validate these results. All things considered, this study demonstrates how well new revolutionary methods work to solve nonlinear differential equations related to chaotic systems. A flexible, adaptive, and computationally effective method for simulating intricate dynamical events is the PSO-ASA optimized ANN framework, which incorporates the Mexican Hat activation function.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence
ANN	-	Artificial Neural Network
NN	-	Neural Network
FFNN	-	Feed Forward Neural Network
DE	-	Differential Equation
PI-NN	-	Physics-based unsupervised neural network
MHW	-	Mexican Hat Wavelet
GA	-	Genetic Algorithm
SQP	-	Sequential Quadratic Programming
LAF	-	Linear Activation Function
RNN	-	Recurrent Neural Network
RBF	-	Radial Basis Function
RK	-	Runge Kutta

ACKNOWLEDGMENT

First and foremost, I offer my heartfelt gratitude to Almighty Allah for granting me the strength, patience, and wisdom to complete this research successfully. This accomplishment would not have been possible without the sincere support and encouragement from various individuals, to whom I am deeply thankful. I am especially indebted to my research supervisor, Dr. Muhammad Rizwan and Co-Supervisior Dr. Farhad Riaz, whose unwavering guidance, encouragement, and valuable insights played a crucial role throughout my research journey. My sincere thanks extend to our esteemed faculty members and all other teachers whose knowledge and support greatly contributed to my academic and research growth. I would also like to acknowledge the administrative staff of the Department of Mathematics for their continuous cooperation and assistance, which made many challenges easier to overcome. Lastly, I extend my gratitude to all those whose contributions may not be named individually, but whose support has been no less valuable. Thank you all for being part of this journey.

DEDICATION

This thesis work is dedicated to my parents, family, and my teachers throughout my education career who have not only loved me unconditionally but whose good examples have taught me to work hard for the things that I aspire to achieve.

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

Differential equations are crucial for simulating complex processes in many fields of science and engineering. Nonlinear differential equations often exhibit complex and chaotic behavior, making them potentially challenging to solve using traditional numerical techniques. Artificial Neural Networks (ANNs), which capitalize on their ability to recognize subtle patterns and approximate complex functions, offer a useful alternative. Wavelet-based activation functions, such as the Mexican Hat wavelet, are still underutilized despite the growing popularity of neural network-based solutions, particularly when dealing with chaotic and nonlinear systems. By analyzing how well a Mexican Hat wavelet-based ANN solves the Lorenz Model, the work seeks to bridge this gap. The Lorenz Model is a set of differential equations that describes a deterministic, chaotic model of atmospheric convection. Edward Lorenz was the first, who introduced Lorenz Model in 1963 [1] to study weather patterns and has since become a cornerstone in chaos theory. This chapter reviews the various research papers that are relevant to our field of study. Additionally, highlight numerous articles and highlight the literature that is relevant to the research issue. This chapter will assist us in strengthening our foundational understanding of our field.
Kudryashov [2] studied analytical solutions to the Lorenz system, identifying and categorizing all precise solutions, making these non-linear differential equations (DE) a standard model for chaotic systems. Bougofa et al. [3] studied the Lorenz system, a deterministic chaos model using a numerical analysis, Algaba et al [4] were able to determine the Lorenz system's bifurcations. Eusebius Doedel et al. [5] showed that there exists preturbulence in the Lorenz system's three-dimensional phase space. To analyze the dynamics of a chaotic system, it is

essential to study its behavior in detail, Wu et al. [6] looked at the irregular patterns in nonlinear dynamics using Reservoir-based neural processing. In order to solve the Lorenz differential equation, Chowdhury et al. [7] used the semi-analytical solution method and compared it to the multistage semi-analytical technique and the fourth order Runge–Kutta (RK4). Te Krishnan et al. [8] studied heat transfer in fluids and showed that a simplified ODE model, similar to the Lorenz system, can describe temperature changes and fluid flow. Zlatanovska and Piperevski [9] solved the Lorenz model using the Laplace homotopy analysis method, where solvable classes of differential equations were applied to study solutions of the reduced third-order Lorenz model. In their study of chaotic dynamical systems, Klöwer et al. [10] found that aperiodic system modeling employing predictable fnite-accuracy of result numbers always produce orbital paths that eventually turn periodic. Muhammad Naeem Aslam et al. [11] utilized ANN techniques and hybrid optimization technique PSO-NNA to solve the Lorenz model using log sigmoid as an activation function.

Differential equations can be approximated and solved by ANN. NN, both supervised and unsupervised, are utilized to compute DE. Yang et al. [12] proposed PI-GAN as a solution to solve DE, by employing unsupervised NN combined with optimization techniques. Farhad et al. [13], examine a quantitatively infected CD4 + T cell system using the unsupervised feedforward artificial neural network (FFANN) for the model, with adjustable parameters optimized using PSO, NMSM and hybrid methods. Using a Morlet wavelet neural network and a genetic algorithm, Shahid et al. [14] solve a second-order delay differential perturbed singular model that addressed three issues with the numerical performance of DD-PSM. Mariam et al. [15] proposed a hybrid PSO-ANN approach to optimize wind farm layouts by taking into account local wind characteristics and topographical constraints to maximize energy extraction. David et al. [16] utilized a PI-NN to solve differential equations representing dynamic system movement, improving model predictability and achieving similar numerical errors. Awatef et al. [17] used artificial neural network techniques to predict heat efficiency in a vacuum tube solar collector, achieving high accuracy with minimal margin of error.

Liu et al. [18] developed a novel category of PI-GANs to address stochastic issues with distributed measurements, incorporating stochastic differential equations for improved stability. M.Raissi et al. [19] PI-NN were employed to compute nonlinear differential equations, using discrete and continuous time models, demonstrating the effectiveness of these techniques. Hagge et al. [20] utilized recurrent neural networks to solve differential equations with unknown sink terms, utilizing deep learning approaches for a fedbatch bioreactor simulation challenge. Investigation of dengue fever Model was done by Riaz et al. [21], using ANN-PSO-SQP and MHW as an activation function. Shoaib et al. [22], Analyze the dynamics of hepatitis (B) virus using Log sigmoid function and ANN-GA-SQP techniques. Bhat et al. [23], to get the results of nonlinear Liénard DE by employing an Artificial NN combined with GA and SQP, using the sigmoid activation function.

The PSO has been used successfully in a variety of domains, including robotics [24, 25], electric system [26] and sport sciences [27]. Pires et al. [28] recently controlled the PSO's rate of convergence using fractional calculus. Since evolutionary techniques [29] yield the most successful PSO variations, this study aims to control the convergence rate of an advanced PSO variant, inspired by the work of Pires et al., was analyzed.

Powerful techniques for examining epidemic models with nonlinear dynamics have been made available by recent advancements in stochastic intelligence systems, particularly feed-forward ANNs. The nonlinear system of Leptospirosis disease (LD) is solved in this paper by Farhad et al.[30] using a swarming-optimized neuro-heuristic scheme. The technique uses Sequential Quadratic Programming (SQP) for local refining and the Mexican Hat Wavelet-based ANN (MHW-ANN) with Particle Swarm Optimization (PSO) for global search. Farhad et al.[31] uses Mexican Hat activation function to study hybrid MHW-ANN-PSO-SQP strategy to address the Zika virus's nonlinear SEIR model. The Runge–Kutta solver and ANN-based GA-ASA are used to validate the approach, which optimizes an error-based fitness function. Its precision, stability, dependability, and efficiency are confirmed by the results, and Mean Execution Time is used to measure complexity.

A comprehensive literature review is presented, discussing previous research on getting the solutions of DE using ANN and studies related to it is the focus of our thesis.

CHAPTER 2

BASIC CONCEPTS AND DEFINITIONS

In Chapter 2, a list of some significant subjects pertaining to our field of study is provided. The subjects that aid and direct us in comprehending the fundamentals and phenomena of our field of study. These subjects include differential equations, AI, NN, and numerous kinds of NN, their varieties, as well as several approaches to solving differential equations.

2.1 Artificial Intelligence (AI)

AI is a branch of science and engineering focused on evolution of system theory that demonstrate behaviors typically associated with human intelligence [32]. Artificial intelligence (AI) has made recent advancements in research studies and real-world achievements possible. Numerous mathematical fields were significantly impacted by the development of artificial intelligence. A biological subfield of artificial intelligence is ANN.

2.2 Artificial Neural Networks (ANNs)

ANN is a class of deep learning algorithms influenced by human brain processes. Similar to the human brain, an NN contains a network of neurons. The neurons in NN are organized into

multiple layers, including input, hidden, and output layers, to enable complex data processing. Artificial Neural Network (ANN) techniques are also utilized to obtain approximate solutions of ODEs and PDEs. These equations can be solved using both supervised and unsupervised neural network models. Neural network techniques are capable of solving ODEs and PDEs by generating approximate direct solution form, leveraging the function approximation ability of FF-NN. In this approach, a feed-forward neural network acts as the core estimation tool, with its parameters optimized to reduce the corresponding error function.
When compared to other computational techniques, the ANN technique's main advantages are its simplicity, speed, and capacity to handle complex and nonlinear relationships between variables and extracted data [33].

2.2.1 History of NNs

This part outlines the origin of neural networks. Neural network technology was introduced in the 1940s [34]. McCulloch and Pitts' 1943 study, which showed that neural networks could perform arithmetic and logical operations, is where neurocomputing got its start [35]. Wiener and von Neumann, among other experts, proposed computer design that was influenced by the human brain [36, 37]. Marvin Minsky's 1951 Snark neurocomputer was a typical era invention, although it did not perform any significant information processing functions [34]. The earliest successful neurocomputer, known as the Mark I Perceptron, was created by Frank Rosenblatt, Charles Wightman, and their team during 1957–1958 [38]. Marvin Minsky and Seymour Papert spearheaded an effort to denigrate neural network research and shift funds to artificial intelligence in the middle of the 1960s [39]. In the 1970s, many of the present leaders in the subject started publishing their work. Amari [40], Fukushima [41], Grossberg [42], Klopf, and Gose [43] are a few examples. In the early 1980s, neurocomputing researchers began exploring Neuro-inspired computing and NN applications. John Hopfield [44], a renowned physicist, wrote two influential papers on neural networks and gave numerous lectures worldwide. The domain of NNs experienced significant growth in 1986 following the scholarly work of the Parallel Distributed Processing books [45]. The IEEE International Professional workshop on NNs in 1987 established the International NN Society, leading to the founding of journals. Subsequently, the journal Neural Computation was launched in 1989, followed by the introduction of IEEE Transactions on Neural Networks in 1990 [34].

2.3 Mathematical Modelling of ANNs

The arriving signals called inputs $S = xj|j = 1,2,...,n$ will taken by the input layer. After the incoming signals, known as inputs, are multiplied by the adjusted connection weights, they are first added up, or combined, and then they are run through a activation function to generate the particular neuron's output. The weighed total of the neuron's inputs is the activation function [46]. Eq 2.1 provides a straightforward equation that summarizes the ANN operations [47]

$$Y_i = f\left(b + \sum_{j=1}^{n} w_{ij}x_j\right), \tag{2.1}$$

where n represents the count of neurons in the hidden layer, w_{ij} corresponds the weight of the connection linking the input variable to the hidden layer, and b is the bias at the hidden layer. The input variable is called xj, the output variable is called Yi, and the activation function is called f. Fig. 2.5 provides a detailed illustration of ANNs.

2.4 Activation Functions

As mentioned in the section above, the neuron's numerous weights w_{ij} and inputs x_j determine the output. Although sigmoid, step, linear, non-linear, and sign functions are widely used, a threshold function was initially proposed as the activation mechanism for a neuron's output. The neuron produces a result, say y, by acting in an activation-function like f(net), which is expressed as:

$$Y = f(net) = f\left(b + \sum_{j=1}^{n} w_{ij}x_j\right) \tag{2.2}$$

Here are a few examples of activation functions used in NNs.

2.4.1 Linear Activation Function

A neuron's linear transformation function is known as the LAF, and it is shown in Fig 2.1. The mathematical expression is given by:

$$Y = f(net) = f\left(\theta + \sum_{j=1}^{n} w_{ij}t_j\right) = net \tag{2.3}$$

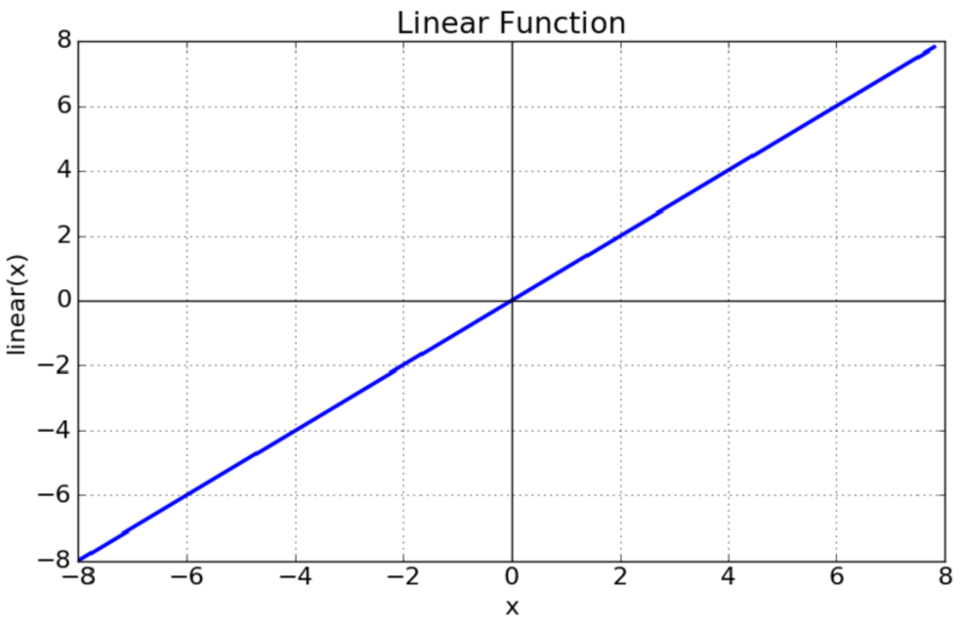


Figure 2.1: Linear Activation Functions

2.4.2 Sigmoid Activation Function

Sigmoid AF is a nonlinear S-shaped function that produces outputs ranging from 0 to 1, converges to +1 when input value approaches infinity, as illustrated in Fig 2.2.

$$y = \frac{1}{1+e^{-\lambda t}} \tag{2.4}$$

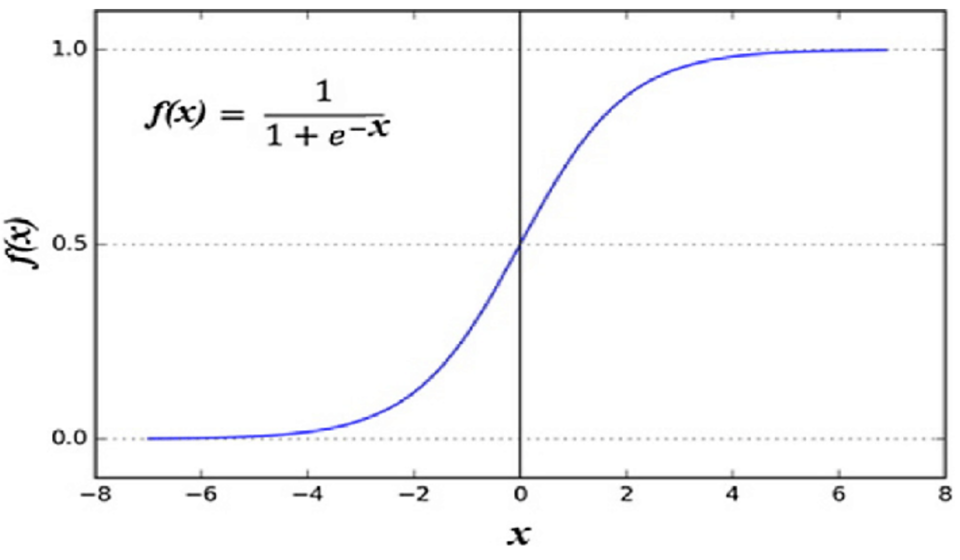


Figure 2.2: Sigmoid Activation Functions

2.4.3 Mexican Hat Activation Function

The Mexican Hat activation function is obtained by taking the second derivative of a Gaussian function, a non-linear activation function with a wave-like form. The graphical expression of Mexican Hat Activation function is given in Fig 2.3. The Mexican Hat Activation Function is mathematically represented as:

$$f(x) = (1 - x^2)e^{-x^2/2} \tag{2.5}$$

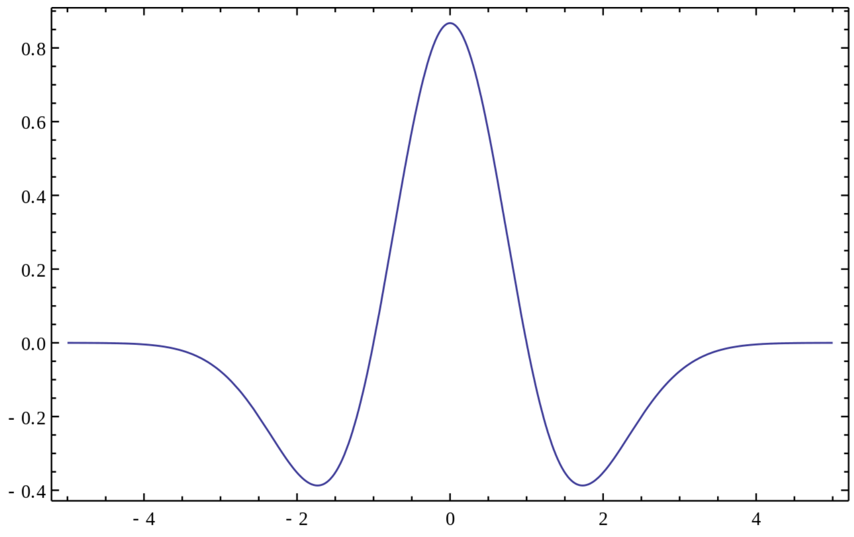


Figure 2.3: Mexican Hat Activation Functions

2.4.4 Morlet Wavelet Activation Function

A vital tool in wavelet transforms and neural networks for time-frequency analysis is the Morlet wavelet activation function, which combines a complex sinusoid and a Gaussian window. The graphical expression of Morlet Wavelet function is given in Fig 2.4. The mathematical form of Morlet Wavelet Activation Function is given as:

$$\psi(t) = e^{i\omega_0 t} e^{-\frac{t^2}{2\sigma^2}} \tag{2.6}$$

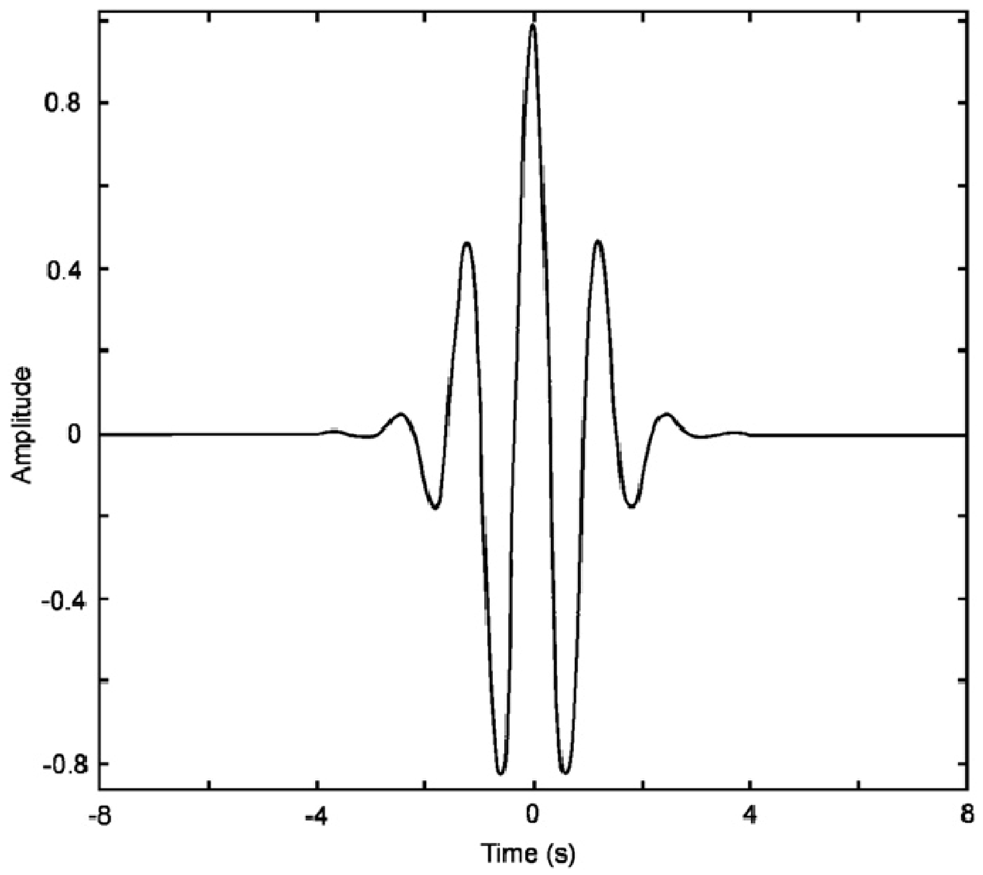


Figure 2.4: Morlet Wavelet Activation Functions

2.5 Neural Network Architecture

One node alone is inadequate for practical, real-world applications, therefore networks with many nodes are frequently used. One of the key early considerations for neural network developers is how nodes are connected, which affects how calculations are carried out. In nature, a neural network is a system for processing data composed of numerous simple, closely connected processing units known as neurons. The term ‘nodes’ refers to neurons and the lengths of synaptic connections [48]. The Architecture of ANN is given in Fig 2.5.

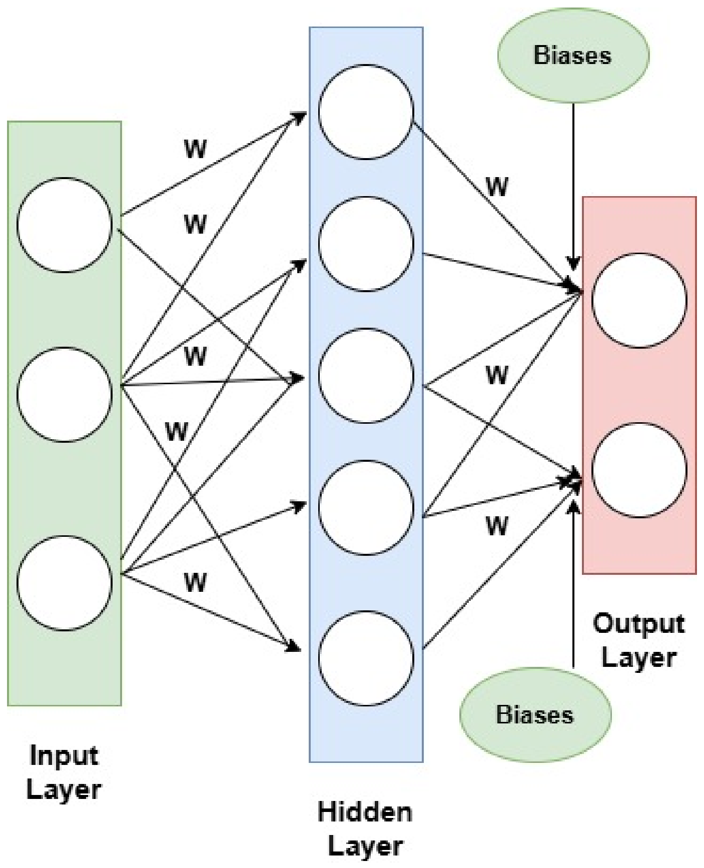


Figure 2.5: Architecture of ANN

2.5.1 Recurrent Neural Networks

A RNN permits flexible links among its nodes, acting as a memory that adapts its internal state to input data sets, making it beneficial for solving problems involving multiple inputs. While learning, In a recurrent network, data flows both from inputs to outputs and back from outputs to inputs. This process is repeated until the output values remain constant. The network is regarded as being in a stable or equilibrium condition at this time.

2.5.2 Feed Forward Neural Networks (FF-NN)

The neurons of a feed-forward neural network (FF-NN) are arranged in layers. FF only generates a single output configuration derived from an input instance rather than a collection of values. The FF-NN is the most basic ANN variant or device. In a network like this, Signals or data are transmitted in one direction, moving from the input layer through the hidden layers to the output layer [49].

2.5.3 Radial Basis Function Neural Network (RBF-NN)

One kind of artificial neural network that use RBF as activation functions is known as RBF-NN. It works especially well for function approximation, interpolation, and pattern recognition. This one is composed of three layers. recognizing the first layer as the input layer, the second as the BBF layer, and the third as the output layer.

2.6 Paradigms of Learning

A key feature of an ANN is its ability to generalize knowledge after training on a dataset. The two main types of learning are:

2.6.1 Supervised Learning

A common machine learning technique is supervised learning, in which the algorithm is trained with data that contains both inputs and accurate outputs. With an emphasis on regression and classification, the model learns the relationship between inputs and outputs to produce accurate predictions on new data. The success of supervised learning depends on having enough high-quality labeled data and choosing the right model for the task. The best results also depend on the data being prepared and assessed correctly.The caliber of the labeled data used for training has a significant impact on a supervised learning model's performance. Generally speaking, using a larger and more representative dataset results in a more accurate model. The workflow for Supervised learning is given in Fig 2.6.

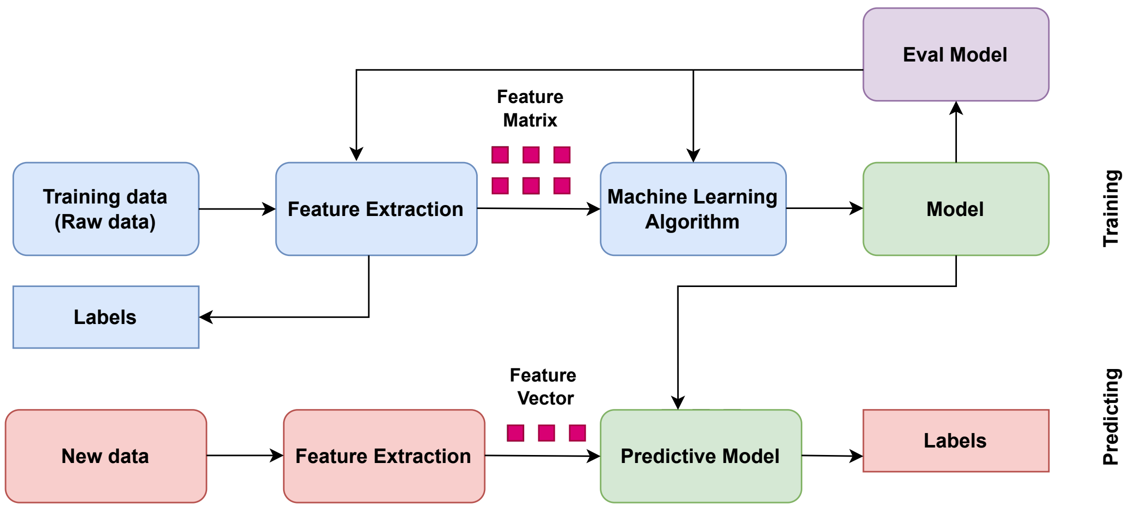


Figure 2.6: Supervised Learning workflow

2.6.2 Unsupervised Learning

In unsupervised learning, the model discovers patterns in unlabeled data without direct supervision. Finding underlying patterns in the data is the goal, with no explicit guidance on what to predict. By assembling related items into groups, unsupervised learning seeks to uncover the core data structure. This method concentrates on streamlining the data without the need for labels, in contrast to supervised learning, which relies on labeled data for training. Unsupervised learning is implemented in fields including customer segmentation, anomaly detection, and data compression and is especially helpful when labeled data is hard to come by or unavailable.The workflow for Unsupervised learning is given in Fig 2.7.

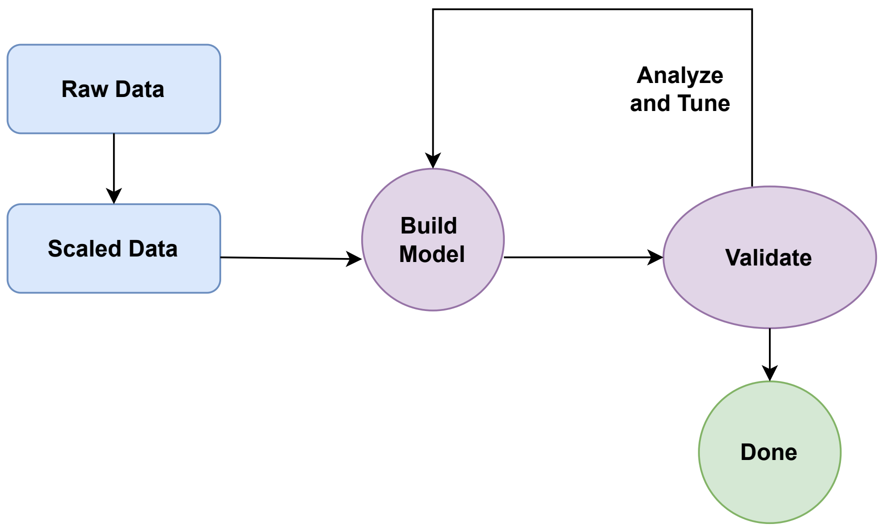


Figure 2.7: Unsupervised Learning workflow

2.7 Differential Equations (DE)

A DE is an equation in mathematics that involves the derivatives of one or more functions [50]. Another interpretation of a differential equation (DE) is the connection between an unsolved function's derivatives and the function itself. Differential equations (DEs) are used to formulate the majority of mathematical models.

2.8 Differential Equations Classification

DEs can be categorized into the following types [51]:

2.8.1 The Ordinary Differential Equation (ODE)

ODEs explain how a function and its derivatives relate to one independent variable. An ODE involves a function and its derivatives, which display the function's change. The order of the ODE is the highest derivative in the equation. Finding a function that, under specific circumstances, satisfies the equation is the goal of solving an ODE. The standard form of an nth-order ordinary differential equation (ODE) is shown below:

$$f(\theta,\varphi,\quad \frac{d\varphi}{d\theta},\quad \frac{d^2\varphi}{d\theta^2},\quad \dots,\quad \frac{d^n\varphi}{d\theta^n}) = 0. \tag{2.7}$$

2.8.2 Partial Differential Equation (PDE)

Multivariable functions and their partial derivatives are included in a PDE. These formulas describe the relationship between a function and its rates of change with respect to multiple variables. A general form of a PDE for a function u(x_1,x_2,...,x_n) involving independent variables x_1, x_2,..., x_n, is as follows:

$$F(x_1,x_2,\dots,x_n,u,\frac{\partial u}{\partial x_1},\frac{\partial u}{\partial x_2},\dots,\frac{\partial^2 u}{\partial x_1^2},\frac{\partial^2 u}{\partial x_2^2},\dots) = 0. \tag{2.8}$$

2.8.3 Delay Differential Equation (DDE)

A derivative of an unknown function based on its historical and current values is represented by a DDE, a type of DE. Stated differently, the rate of change of a function at time t depends on its measurements at the beginning time, as in $t-\tau$, where τ is a variable or constant delay. The general form is given as :

$$\frac{d}{dt}x(t) = f(t,x(t),x(t-\tau)). \tag{2.9}$$

.

2.8.4 Differential Algebraic Equation (DAE)

A Differential Algebraic Equation (DAE) is an equation that combines algebraic and differential equations. Some of the variables in a DAE have derivatives, just like in a normal differential equation, but others simply show up algebraically. The common representation of a DAE is given as:

$$F(t, x(t), \dot{x}(t)) = 0. \tag{2.10}$$

2.8.5 Stochastic Differential Equation (SDE)

A SDE is an equation that reflects the unpredictable or random evolution of a system, while normal differential equations (ODEs) show predictable changes. Noise, like weather, is typically introduced in SDEs. The general form is given as :

$$\frac{d}{dt}x(t) = f(t, x(t)) + \sigma(t, x(t))\, dW(t). \tag{2.11}$$

2.9 Types of Differential Equations Problems

Differential equation problems are generally classified into two main types, as outlined below:

2.9.1 Initial Value Problems (IVPs)

Assigning values to the dependent variable and its derivatives at the same value as the independent variable in the equation is known as an IVP. These complexities usually include a time component.

Example: If time throughout the interval [0, 1] is the independent variable, then the starting value issue could indicate the temperature of a freshly poured cup of coffee at time 0. When a warm cup of coffee is set in a colder room, its original temperature at the time of pouring can be used to model how quickly it cools.

2.9.2 Boundary Value Problems (BVPs)

BVP arises when the values of the dependent variable and its derivatives are defined at the boundaries of the independent variable. In steady-state equilibrium problems, the auxiliary conditions correspond to the boundary conditions imposed along the entire boundary of the solution domain.

Example: If space is the independent variable over the [0, L] interval, then the expression for a boundary value issue would define the values for y(x) for both x = 0 and x = L. Instead of stating the temperature at a single site throughout time, the data may be given at a certain moment in time across entire spatial locations in situations when the problem involves both temperature and time. There are, however, three types of boundary conditions, which are listed below [52]:

Dirichlet Boundary Condition

One kind of boundary condition applied to mathematical problems, especially differential equations, is the Dirichlet Boundary Condition. It indicates the value of the differential equation solution at the domain's edge. In other words, it establishes the dependent variable's value at the borders. The Dirichlet boundary condition defines the precise value of the dependent variable (such as temperature, displacement, or concentration) at the domain boundaries.

Example Consider the partial differential equation that models the temperature distribution u(x,t) along a one-dimensional rod:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial t^2} = g(x,t), \tag{2.12}$$

The BCs are given as:

$$u(0,t) = C \quad \text{and} \quad u(L,t) = D. \tag{2.13}$$

where C and D are constants and L is the rod's length.

Neumann Boundary Condition

The Neumann boundary condition establishes the derivative of the solution at the boundary instead of specifying the value of the function. It is commonly used to depict situations where the rate of change of the function at the boundary is known.

Example

The following is an example of the Neumann boundary condition:

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial t^2} = g(x,t), & \text{domain } 0 < x < L,\ 0 < t < T \\ \frac{\partial u}{\partial x}(0,t) = \alpha, & \text{at } x = 0 \\ \frac{\partial u}{\partial x}(L,t) = \beta, & \text{at } x = L \end{cases} \tag{2.14}$$

Mixed Boundary Condition

Mixed boundary conditions are used in boundary value problems when different types of boundary conditions are applied to different areas of a domain's boundary. It entails combining Dirichlet boundary conditions, which specify the value of the function at the boundary, with Neumann boundary conditions, which define the derivative of the function at the boundary.

Example

Here is an illustration of a mixed boundary condition:

$$\begin{cases} \frac{\partial^2\phi}{\partial\theta^2} + \frac{\partial^2\phi}{\partial t^2} = f(x,y), & \text{(PDE)} \\ \omega_1 \frac{\partial\phi}{\partial n}(0,t) + \omega_2\phi(0,t) = \omega, & \text{(Mixed BC at } \theta = 0) \\ \omega_3 \frac{\partial\phi}{\partial n}(L,t) + \omega_4\phi(L,t) = \omega_i. & \text{(Mixed BC at } \theta = L) \end{cases} \tag{2.15}$$

2.10 Numerical Methods for Solving DEs

Numerical approximation is a crucial principle in applied mathematics and computational science, vital for addressing problems that cannot be solved analytically. This approach involves using numerical techniques to approximate mathematical functions and procedures, aiming to produce solutions that closely align with the exact values. Different scientist use numerical techniques like Runga-Kutta Method, Finite Difference Method, Euler method to solve a wide range of problems [53]. Using numerical techniques has certain limitations. For example, a little error might occasionally result in a large departure from the exact solution, and certain problems may require numerous iterations, which can raise the computing cost. It is difficult to solve singularity, discontinuity problems and problems having irregular geometries by using numerical methods. Some of the numerical methods are given below :

2.10.1 Finite Element Method

The FEM is a numerical technique used to estimate solutions for BVPs involving PDEs. It involves breaking down a complicated issue into lesser, more manageable components (elements), which are subsequently resolved separately and integrated to provide a comprehensive solution. The finite element approach divides the domain under study into a finite set of elements. A piecewise polynomial of low degree is then used to approximate the function. Additionally, they are designed to support only a limited number of features. The fundamental argument for applying an approximation solution on a set of subdomains is that a complex function is easier to represent by a set of simple polynomial terms.

2.10.2 Euler Method

One fundamental numerical method for solving ODEs is the Euler method. This explicit method for initial value problems provides an approximation of the solution at discrete time steps. The general form of the Euler method for solving ODEs is given as [54] :

$$y_{n+1} = y_n + h \cdot f(t_n, y_n), \tag{2.16}$$

where h is the step size, $y(n)$ is the current value of solution, $y(n+1)$ is the next value to find, $t(n)$ is the current time and $f(t(n), y(n))$ is the derivative of given ODE.

2.10.3 Taylor Method

The Taylor method is implemented to numerically solve ODEs. Its basis is the extension of the ODE solution into a Taylor series around a specified point.The Taylor series expansion of a function $y(t)$ around a point t_0 is given by [55]:

$$y(t) = y(t_0) + \frac{1}{1!}y'(t_0)(t-t_0) + \frac{1}{2!}y''(t_0)(t-t_0)^2 + \frac{1}{3!}y^{(3)}(t_0)(t-t_0)^3 + \dots \tag{2.17}$$

2.10.4 Runge-Kutta Method

RK method is widely utilized for solving IVPs in DE. The Runge–Kutta approach allows the construction of highly accurate numerical solutions using only the functions themselves, without needing higher-order derivatives. By offering a framework for intricate numerical algorithms in scientific computing and overcoming impractical or unachievable analytical solutions, Runge-Kutta techniques transformed ODE solution. The Runge-Kutta algorithm is used to find the numerical solution of the ODE y(x) = f (x, y) with initial condition

$$y(x_0) = w_0.$$

In 1900, Runge and Kutta presented the traditional fourth-order iterative method [56]

$$\begin{aligned}
w_{k+1} &= w_k + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)h, \\
K_1 &= f(x_k, w_k), \\
K_2 &= f\left(x_k + \tfrac{1}{2}h,\, w_k + \tfrac{1}{2}K_1 h\right), \\
K_3 &= f\left(x_k + \tfrac{1}{2}h,\, w_k + \tfrac{1}{2}K_2 h\right), \\
K_4 &= f\left(x_k + h,\, w_k + K_3 h\right).
\end{aligned} \tag{2.18}$$

2.11 ANN in solving a Differential Equations

Artificial neural networks are very helpful for solving a differential equation and is the best way as compared to other methods. When DEs are complex, then ANN techniques are very helpful in giving the best solution by using different optimization techniques. In ANN, by using different optimization techniques, we try to minimize the error function and get the best optimal solution. When the DE is very complex, the boundary condition is very complex, and nonlinearity occurs in the equation, the ANN techniques are best to implement. There are different optimization techniques that are widely used nowadays to get stable and accurate solutions. The optimization techniques include Active Set Algorithm (ASA), Particle Swarm Optimization (PSO), Sequential Quadratic Programming (SQP) and Genetic Algorithm (GA). ANN techniques are nowadays widely used to solve nonlinear DE due to their efficiency and stability in getting the solution.

CHAPTER 3

NEURO-COMPUTING SOLUTION FOR LORENZ DIFERENTIAL EQUATIONS THROUGH ARTIFCIAL NEURAL NETWORKS INTEGRATED WITH PSO-NNA META-HEURISTIC ALGORITHMS: A COMPARATIVE STUDY

3.1 Introduction

In this paper, artificial neural network (ANN) techniques are used to solve the nonlinear Lorenz model. Optimization techniques are used to obtain the best optimal solution. In this research, particle swarm optimization (PSO) combined with Neural Network Algorithm (NNA) is used to get the best optimal solution of the Lorenz model, which is difficult and very complex if we solve it using traditional numerical methods. In the end, to check the accuracy, the results of ANN-based techniques are compared with the already existing numerical methods.

3.2 The Equations

The Lorenz model's non-linear differential equations are given as:[57]

$$\frac{dx}{dt} = \sigma(y-x), \tag{3.1}$$

$$\frac{dy}{dt} = -y + \rho z - xz, \tag{3.2}$$

$$\frac{dz}{dt} = -\beta z + xy. \tag{3.3}$$

For the Lorenz Model, the initial conditions are given as:

$$x(0) = x_0, \qquad y(0) = y_0, \qquad z(0) = z_0. \tag{3.4}$$

where σ is the Prandtl number, ρ is the Rayleigh number, and β is the system parameter.

3.3 Methodology

In this paper, the combined optimization technique that includes PSO and NNA is used to solve the Lorenz model. The artificial neural network framework, that is, ANN-PSO-NNA, is created by using a logarithmic sigmoid as the activation function. The purpose is to get the best solution for the Lorenz model. In the hidden layer, the weights and biases of the ANN architecture are improved using PSO-NNA. A fitness function is created to minimize the error. In the end, an error analysis was performed to verify the stability and accuracy of the model.

3.4 Mathematical Modelling

In order to deal with chaotic systems, Physics-Informed Neural Networks (PINNs) are created. The fitness function is an essential component in Artificial Neural Networks because it introduces non-linearity into the model and increases the system's accuracy and stability. Some fitness functions are ReLU, Log sigmoid, and Mexican Hat. In this research, the Log sigmoid is used, and the mathematical form of the Log sigmoid activation function is given as:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.5}$$

The following derivatives are found when an artificial neural network is used to solve the Lorenz differential equations:

$$\hat{x}'(t) = a_{xi} W_{xi} \frac{e^{-(b_{xi}+W_{xi}t)}}{\left(1+e^{-(b_{xi}+W_{xi}t)}\right)^2}, \tag{3.6}$$

$$\hat{y}'(t) = a_{yi}W_{yi}\frac{e^{-(b_{yi}+W_{yi}t)}}{\left(1+e^{-(b_{yi}+W_{yi}t)}\right)^2}, \tag{3.7}$$

$$\hat{z}'(t) = a_{zi}W_{zi}\frac{e^{-(b_{zi}+W_{zi}t)}}{\left(1+e^{-(b_{zi}+W_{zi}t)}\right)^2}. \tag{3.8}$$

3.5 ANN based Fitness Function

The ANN-based fitness function is formed by calculating the residuals of the Lorenz equations after substituting the trial solutions. These residuals, which depend on network parameters and activation functions, are squared and summed to produce an error value. Optimization algorithms then minimize this error to obtain accurate solutions. The following are the fitness function obtained from the artificial neural network (ANN) model:

$$\varepsilon_x = \sum_{j=1}^{k}\left(a_{xi}W_{xi}\frac{e^{-(b_{xi}+W_{xi}t)}}{\left(1+e^{-(b_{xi}+W_{xi}t)}\right)^2} - \sigma\left(\frac{a_{yi}}{1+e^{-(W_{yi}+b_{yi})}} - \frac{a_{xi}}{1+e^{-(W_{xi}+b_{xi})}}\right)\right)^2, \tag{3.9}$$

$$\begin{aligned}\varepsilon_y = \sum_{j=1}^{k}\Bigg(a_{yi}W_{yi}\frac{e^{-(b_{yi}+W_{yi}t)}}{\left(1+e^{-(b_{yi}+W_{yi}t)}\right)^2} - R\frac{a_{xi}}{1+e^{-(W_{xi}+b_{xi})}} + \frac{a_{yi}}{1+e^{-(W_{yi}+b_{yi})}} \\ + \frac{a_{xi}}{1+e^{-(W_{xi}+b_{xi})}}\frac{a_{zi}}{1+e^{-(W_{zi}+b_{zi})}}\Bigg)^2,\end{aligned} \tag{3.10}$$

$$\begin{aligned}\varepsilon_z = \sum_{j=1}^{k}\Bigg(a_{zi}W_{zi}\frac{e^{-(b_{zi}+W_{zi}t)}}{\left(1+e^{-(b_{zi}+W_{zi}t)}\right)^2} - \frac{a_{xi}}{1+e^{-(W_{xi}+b_{xi})}}\frac{a_{yi}}{1+e^{-(W_{yi}+b_{yi})}} \\ -B\frac{a_{zi}}{1+e^{-(W_{zi}+b_{zi})}}\Bigg)^2.\end{aligned} \tag{3.11}$$

and, we can also assume

$$\varepsilon_{ic} = \left((\hat{x}(t)-\hat{x}_0)^2 + (\hat{y}(t)-\hat{y}_0)^2 + (\hat{z}(t)-\hat{z}_0)^2\right), \tag{3.12}$$

$$\varepsilon = \varepsilon_x + \varepsilon_y + \varepsilon_z + \varepsilon_{ic}, \tag{3.13}$$

where ε is the fitness function based on physics-informed neural networks, and $\hat{x_0}$, $\hat{y_0}$, and $\hat{z_0}$ are the initial conditions.

3.6 Meta-heuristic optimization algorithms

Meta-heuristic algorithms are crucial in solving non-linear optimization problems in engineering domains. These methods provide approximate solutions in situations where conventional approaches face difficulties due to complexity or limitations. Over time, researchers have introduced various meta-heuristic algorithms such as GA, PSO, Water Cycle Algorithm, Ant Colony Optimization, and Simulated Annealing.

3.7 Particle Swarm Optimization (PSO) Algorithm

PSO is associated to the field of evolutionary worldwide-search method within big multifaceted resolution gaps. Population-based searches are a useful alternative for large search spaces, but they don't guarantee finding the optimal solution. It is an engineering model of swarm intelligence that draws inspiration from insect swarms, fish schools, and flocks of birds [58] . Kennedy and Ehrhart created the PSO's initial mathematical formulation [59] . The primary goal of PSO is to examine the nonlinear complex optimization issues that are challenging to resolve with conventional method.

The PSO method, like other population-based evolutionary algorithms, uses a large number of randomly generated seeking individuals, called particles. PSO helps adjust particle locations based on both individual and swarm-identified global best through an iterative process. Each particle is assigned with random position and velocities. Each of particle is representing the best solution for the optimization problem, similar to a flock of birds acclimating to their surroundings. Personal best (pbest) implies to the position that performs ideal for each particle, whereas global best (gbest) implies to the location that performs ideal for the entire swarm. Position update and velocity update are the two primary operators in particle swarm. Throughout the optimization process, particles are driven toward their own optimal positions and the optimal location found by the whole swarm. Considering every individual present speed, separation from its previous optimal position, and for every iteration, a new velocity value is calculated based on the distance from the global best location. The updated velocity value is then used to find the particle's next location in the search space. Then, until a minimum error is achieved, or a predefined number of times, this process is repeated [60]. Chapter 4 provides a comprehensive mathematical formulation.

3.8 Neural Network Algorithm (NNA)

NNA is an optimization technique based on biological and artificial neural networks, designed for complex optimization problems. It uses population size and stopping criteria, population updates, weight matrix adjustments, bias operations, and transfer functions. In a study, NNA is hybridized with Particle Swarm Optimization to fine-tune ANN weights for solving the Lorenz system.

3.9 Results and Discussion

For the present analysis, predefined numerical values were assigned to the parameters $\sigma = 0.1$, $\rho = 0.2$, and $\beta = 0.3$ and these values were subsequently employed to evaluate the Lorenz differential equations. Here, the defined numerical values allocated to the parameters σ, ρ, and β have been used to assess the Lorenz diferential equations in the given equation. The problem is solved using artificial neural networks (ANNs) with ten (10) neurons in the hidden layer and ninety (90) correspondence weights, the artificial neural network-constructed fitness function in this instance, where $t \in [0,1]$ with a step size of 0.1. PSO and NNA were hybridized to determine the ideal weights for artificial neural networks.In this work, the ND solver numerical technique has been employed, and the accuracy of the method has been evaluated through error analysis. The results are given in table 3.1, 3.2, 3.3.

t	**Numerical** $x(t)$	**ANN** $x(t)$
0.0	0.000000000	5.40E-06
0.1	0.009468358	0.009515209
0.2	0.017943263	0.017994147
0.3	0.025521751	0.025530937
0.4	0.032291400	0.032278219
0.5	0.038331266	0.038342661
0.6	0.043712682	0.043772499
0.7	0.048500038	0.048586562
0.8	0.052751446	0.052812704
0.9	0.056519362	0.056520039
1.0	0.059851150	0.059839029

Table 3.1: Comparison of $x(t)$ in the Lorenz Model

t	**Numerical** $y(t)$	**ANN** $y(t)$
0.0	1.000000000	0.999997313
0.1	0.904930603	0.904960789
0.2	0.819077383	0.819086889
0.3	0.741542942	0.741558898
0.4	0.671516632	0.671554520
0.5	0.608266533	0.608300919
0.6	0.551132369	0.551137058
0.7	0.499518461	0.499497847
0.8	0.452887662	0.452874396
0.9	0.410755511	0.410778667
1.0	0.372685019	0.372719378

Table 3.2: Comparison of $y(t)$ in the Lorenz Model

t	**Numerical** $z(t)$	**ANN** $z(t)$
0.0	0.000000000	4.85E-06
0.1	0.000446736	0.000476715
0.2	0.001598645	0.001635725
0.3	0.003222261	0.003233088
0.4	0.005138600	0.005132109
0.5	0.007211748	0.007215832
0.6	0.009340018	0.009372364
0.7	0.011448536	0.011502689
0.8	0.013483495	0.013534246
0.9	0.015407483	0.015433005
1.0	0.017195735	0.017209134

Table 3.3: Comparison of $z(t)$ in the Lorenz Model

3.9.1 Comparison of Results

It is clear that the Artificial Neural Network (ANN)-based approach provides a more accurate and optimal answer when compared to the numerical solution produced using conventional approaches. When tackling the identical problem, however, the ANN model shows a much smaller error, suggesting that the ANN technique is better able to represent the complex patterns and chaotic dynamics present in nonlinear systems. One factor contributing to the ANN model's exceptional success is its capacity to learn and generalize intricate correlations between variables. This increase in accuracy not only demonstrates the ANN-based solution's efficacy but also its potential as a dependable and effective approach to solving challenging differential equations.

3.10 Novelty of Our Research Thesis

According to a thorough analysis of the literature, scientists today are primarily interested in problem-solving through hybrid optimization and artificial neural networks. The ANN techniques are used to solve non-linear differential equations because they produce effective and precise results [61, 62, 63] . In this study, the hybrid optimization technique is used, which combines the global and local optimizers to solve the Lorenz model. The PSO is used as a global optimizer, and ASA is used as a local optimizer, while the Mexican Hat is used as the activation function. The ANN-PSO-ASA framework is created to solve the Lorenz model and obtain the best optimal solution. In the end, the results are then compared with the numerical methods, and different performance measures are used to check the stability and accuracy of the Lorenz model.

CHAPTER 4

DESIGN OF MEXICAN HAT WAVELET ARTIFICIAL NEURAL NETWORK FOR SOLVING LORENZ MODEL

4.1 Overview

In the field of chaos theory, the most important and discussed chaotic system is the Lorenz model. The chaotic system is highly sensitive with respect to its initial changes, where small changes can cause significantly different outcomes. In this study, the Lorenz system is solved by using ANN-based hybrid optimization techniques to get more precise and accurate solutions. The traditional numerical methods face issues of instability and high computational cost in solving complex non-linear models. To overcome these issues, this study uses ANN-based optimization techniques to obtain more optimal results.

The significant component of the study is the use of both local and global optimizers. The PSO is used as a global optimizer, and ASA is used as a local optimizer. This combined hybrid approach helps to obtain a more accurate solution. These two methods work together to make the neural network more effective by preventing it from getting stuck in local solutions. They strike a good balance between searching carefully in one area and exploring the wider space for better results. The Mexican Hat is used as an activation function because it helps to recognize complex non-linear patterns and then helps to obtain a more optimal solution.

The main purpose of the study is to increase the stability and accuracy of the model by using hybrid optimization techniques, namely PSO and ASA, and by utilizing the effectiveness of the

Mexican Hat activation function. In the end, the ANN-based optimal results are compared with the traditional numerical-based results to check whether the ANN-based hybrid optimization technique is more effective and stable.

The present study intends to support the increasing application of neural networks for numerical problem solving by showing how hybrid optimization techniques can improve the accuracy and stability of ANN-based solutions for complicated nonlinear systems.

4.2 Problem Statement

The Lorenz model is commonly used to study chaotic behavior, illustrating how small changes in initial conditions can lead to significantly different outcomes. It is mostly demonstrated using the coupled nonlinear differential equation-based mathematical model. These differential equations must be accurately solved in order to comprehend the dynamics of the system. Numerous numerical solutions, such as Runge-Kutta and Euler Method [64], were proposed by researchers to solve the Lorenz model. However, these approaches are not generalized, have lower accuracy, and have huge computational cost. Lately, artificial neural network (ANN)-based numerical solvers have become effective instruments for approximating solutions of numerous systems that rely on nonlinear differential equations [65]. This study designs and evaluates a Mexican Hat-based artificial neural network (ANN) optimized with hybrid PSO-ASA for the numerical treatment of the Lorenz Model, to enhance the stability and accuracy of Model.

4.3 Research Gap

It is very difficult and challenging to solve the Lorenz model due to its chaotic nature, that is, small changes in initial conditions lead to very different outcomes. Different numerical methods have been developed to find its solution, but there are still some limitations that need to be addressed.

In the previous studies, the Lorenz model is solved by using ANN techniques combining Particle swarm optimization (PSO) and Neural Network Algorithm (NNA) to find the optimal solution, but there are still some limitations that exist.

- The optimization techniques used PSO and NNA to solve the Lorenz model, but they could not effectively balance local and global searches.
- Some activation functions, which are specially made to deal with non-linear models, are not used in previous research.
- There are many other optimization techniques that produce more effective and accurate results but are not used in this study.

4.4 Addressing the Research Gap

In order to overcome the limitations, in my research, Particle Swarm Optimization (PSO) is used as a global optimizer and Active Set Algorithm (ASA) is used as a local optimizer to solve the Lorenz model and get a more optimal solution. To get more effective results, the Mexican Hat activation function is used, which deals with the non-linearity in the model.

- The Mexican Hat wavelet activation function improves the ANN's potential to identify chaotic action via local feature extraction.
- PSO-ASA produces more effective and accurate results by dealing with both global and local searches.
- A thorough contrast with other numerical methods indicates that the proposed approach is more stable and manages chaotic systems effectively with fewer errors.

4.5 Research Objectives

Major objectives are as follows:

1. To create a new Mexican Hat Wavelet based artificial neural network (MHW-ANN) process that solves differential equations based on the Lorenz model by combining the hybrid optimization of PSO-ASA with the processing efficiency of Mexican Hat.

2. To evaluate the accuracy and stability of the proposed scheme against current numerical methods for solving the Lorenz model.

4.6 Methodology

Machine learning techniques have made significant progress in solving differential equations (DEs), but their application has been challenging. The most encouraging outcomes have been achieved by improving learning of problem-solving methods or the solution itself.

The Lorenz Model is solved using MHW-ANNs, which are used to construct the fitness function, and PSO-ASA is used for optimization. The MHW-ANNs-PSO-ASA scheme is shown in Fig 4.1

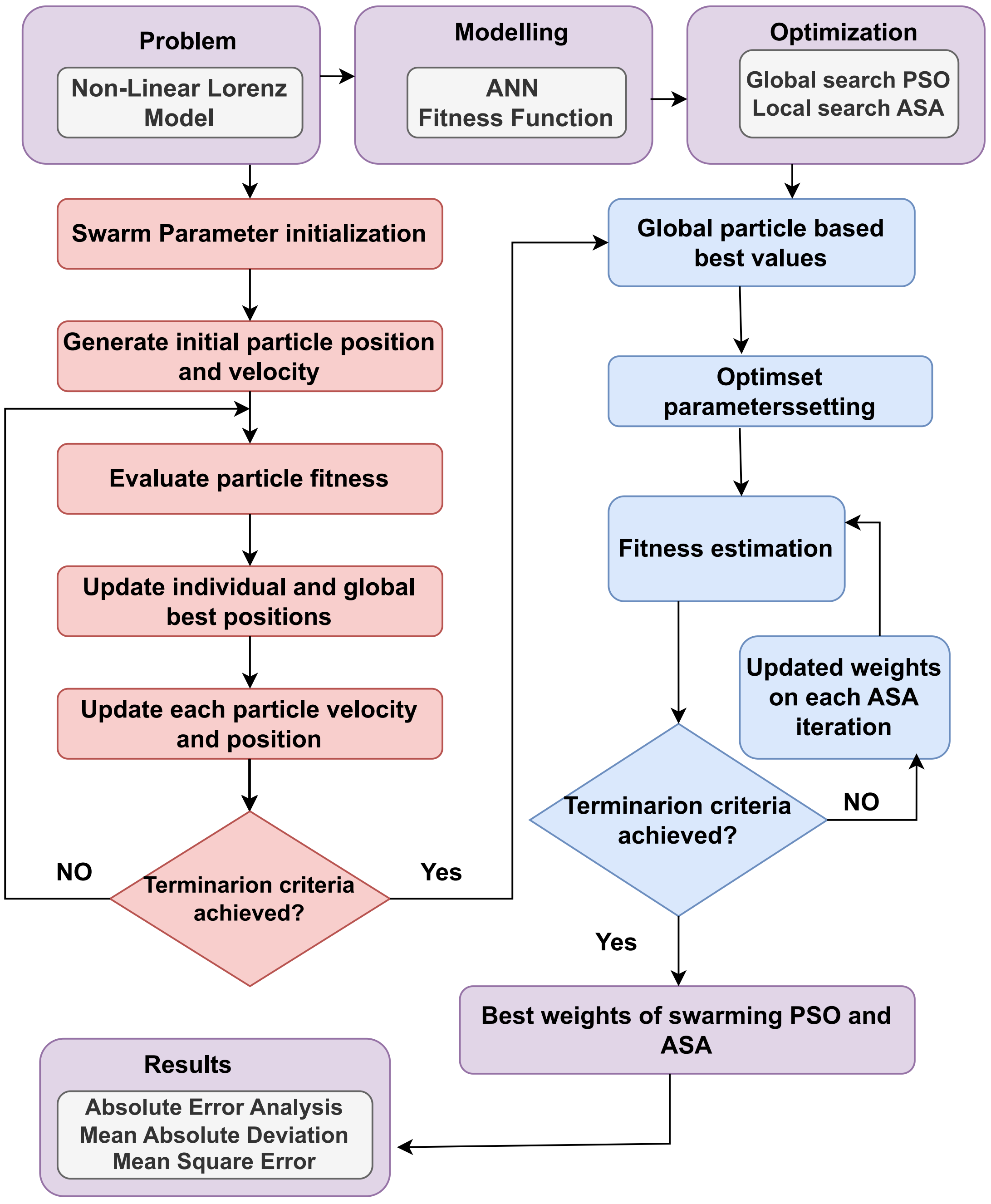

Problem
Non-Linear Lorenz Model
Modelling
ANN
Fitness Function
Optimization
Global search PSO
Local search ASA
Swarm Parameter initialization
Generate initial particle position and velocity
Evaluate particle fitness
Update individual and global best positions
Update each particle velocity and position
Terminarion criteria achieved?
NO
Yes
Global particle based best values
Optimset parameterssetting
Fitness estimation
Updated weights on each ASA iteration
Terminarion criteria achieved?
NO
Yes
Best weights of swarming PSO and ASA
Results
Absolute Error Analysis
Mean Absolute Deviation
Mean Square Error

Figure 4.1: Methodology of ANN

4.6.1 Mathematical Modelling of Neural Networks (NNs)

The effectiveness of approximation theory in terms of continuous mapping is used to create the mathematical model of neural networks for ODEs. The Lorenz model's mathematical formulation is represented by system (4.1,4.2,4.3), which uses feed forward MH-ANNs to express the approximation results and their first-order derivatives given as

$$\begin{cases} \hat{x}(t) = \sum_{i=1}^{k} \alpha_{xi} f(W_{xi}t + b_{xi}), \\ \frac{d\hat{x}(t)}{dt} = \sum_{i=1}^{k} \alpha_{xi} W_{xi} f'(W_{xi}t + b_{xi}), \\ \frac{d^2\hat{x}(t)}{dt^2} = \sum_{i=1}^{k} \alpha_{xi} W_{xi}^2 f''(W_{xi}t + b_{xi}), \\ \vdots \\ \frac{d^m\hat{x}(t)}{dt^m} = \sum_{i=1}^{k} \alpha_{xi} W_{xi}^m f^{(m)}(W_{xi}t + b_{xi}), \end{cases} \tag{4.1}$$

$$\begin{cases} \hat{y}(t) = \sum_{i=1}^{k} \alpha_{yi} f(W_{yi}t + b_{yi}), \\ \frac{d\hat{y}(t)}{dt} = \sum_{i=1}^{k} \alpha_{yi} W_{yi} f'(W_{yi}t + b_{yi}), \\ \frac{d^2\hat{y}(t)}{dt^2} = \sum_{i=1}^{k} \alpha_{yi} W_{yi}^2 f''(W_{yi}t + b_{yi}), \\ \vdots \\ \frac{d^m\hat{y}(t)}{dt^m} = \sum_{i=1}^{k} \alpha_{yi} W_{yi}^m f^{(m)}(W_{yi}t + b_{yi}), \end{cases} \tag{4.2}$$

$$\begin{cases} \hat{z}(t) = \sum_{i=1}^{k} \alpha_{zi} f(W_{zi}t + b_{zi}), \\ \frac{d\hat{z}(t)}{dt} = \sum_{i=1}^{k} \alpha_{zi} W_{zi} f'(W_{zi}t + b_{zi}), \\ \frac{d^2\hat{z}(t)}{dt^2} = \sum_{i=1}^{k} \alpha_{zi} W_{zi}^2 f''(W_{zi}t + b_{zi}), \\ \vdots \\ \frac{d^m\hat{z}(t)}{dt^m} = \sum_{i=1}^{k} \alpha_{zi} W_{zi}^m f^{(m)}(W_{zi}t + b_{zi}). \end{cases} \tag{4.3}$$

In the above equations, $b = [b_1, b_2, b_3, \ldots, b_m]$,$\alpha$ = [α_1, α_2, α_3, …, α_m] and w = [w_1, w_2, w_3, …, w_m] are real values that are arbitrarily bounded. The design of MH-ANNs will be introduced for the first time in order to solve the Lorenz model scenario.

The revised network equations from (4.1–4.3), incorporating the MHW activation function, are expressed as follows:

$$f(x) = \frac{2}{\sqrt{3}} \pi^{-\frac{1}{4}} (1 - t^2) e^{-\frac{t^2}{2}}, \tag{4.4}$$

Using activation function, the new form of equations and their derivatives are given as

$$\hat{x}(t)=\sum_{i=0}^{n}\alpha_{xi}\left(\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}\left(1-(W_{xi}t+b_{xi})^{2}\right)e^{-\frac{(W_{xi}t+b_{xi})^{2}}{2}}\right), \tag{4.5}$$

$$\hat{y}(t)=\sum_{i=0}^{n}\alpha_{yi}\left(\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}\left(1-(W_{yi}t+b_{yi})^{2}\right)e^{-\frac{(W_{yi}t+b_{yi})^{2}}{2}}\right), \tag{4.6}$$

$$\hat{z}(t)=\sum_{i=0}^{n}\alpha_{zi}\left(\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}\left(1-(W_{zi}t+b_{zi})^{2}\right)e^{-\frac{(W_{zi}t+b_{zi})^{2}}{2}}\right), \tag{4.7}$$

$$\begin{aligned}\hat{x}'(t)=\sum_{i=1}^{n}\Big[&-2\alpha_{xi}\cdot\frac{2}{\sqrt{3}}\pi^{-0.5}e^{-0.5(W_{xi}t+b_{xi})^{2}}W_{xi}(W_{xi}t+b_{xi})\\&-\alpha_{xi}\frac{2}{\sqrt{3}}\pi^{-0.5}e^{-0.5(W_{xi}t+b_{xi})^{2}}W_{xi}(W_{xi}t+b_{xi})\left\{1-(W_{xi}t+b_{xi})^{2}\right\}\Big],\end{aligned} \tag{4.8}$$

$$\begin{aligned}\hat{y}'(t)=\sum_{i=1}^{n}\Big[&-2\alpha_{yi}\cdot\frac{2}{\sqrt{3}}\pi^{-0.5}e^{-0.5(W_{yi}t+b_{yi})^{2}}W_{yi}(W_{yi}t+b_{yi})\\&-\alpha_{yi}\frac{2}{\sqrt{3}}\pi^{-0.5}e^{-0.5(W_{yi}t+b_{yi})^{2}}W_{yi}(W_{yi}t+b_{yi})(1-(W_{yi}t+b_{yi})^{2})\Big],\end{aligned} \tag{4.9}$$

$$\begin{aligned}\hat{z}'(t)=\sum_{i=1}^{n}\Big[&-2\alpha_{zi}\cdot\frac{2}{\sqrt{3}}\pi^{-0.5}e^{-0.5(W_{zi}t+b_{zi})^{2}}W_{zi}(W_{zi}t+b_{zi})\\&-\alpha_{zi}\frac{2}{\sqrt{3}}\pi^{-0.5}e^{-0.5(W_{zi}t+b_{zi})^{2}}W_{zi}(W_{zi}t+b_{zi})(1-(W_{zi}t+b_{zi})^{2})\Big].\end{aligned} \tag{4.10}$$

Applying Lorenz systems based on ANNs to a solution that contains ten (10) neurons in a single hidden layer with ninety (90) correspondence weights W = [α_{xi},w_{xi},b_{xi}, α_{yi},w_{yi}, b_{yi}, α_{zi},w_{zi},b_{zi}], W are unsupervised artificial neural network (ANN) weights and biases that are optimized with hybrid PSO-ASA. The MHANNs architecture is given in Fig 4.2.

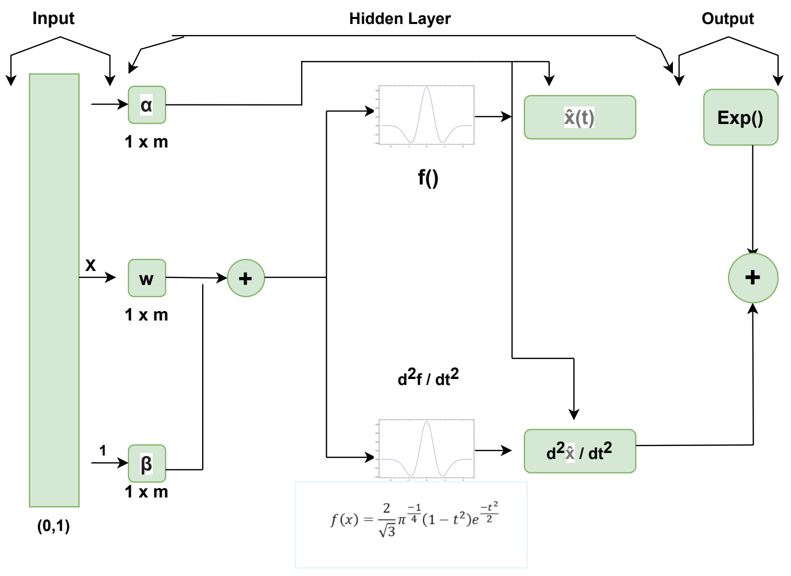


Figure 4.2: MHANNs architecture

4.6.2 Justification for Selecting the Mexican Hat Activation Function

In this work, the Mexican Hat function is used as the activation function because it can capture the oscillatory and chaotic behavior of the Lorenz system more effectively than common functions like sigmoid, tanh, or ReLU, which face problems such as saturation or dead neurons. Since the Mexican Hat is non-monotonic and wavelet-based, it provides better approximation for nonlinear systems and makes the approach more suitable for solving the Lorenz model. The input values are also normalized between 0 and 1 to keep the process stable, avoid issues with large numbers, and improve the efficiency of learning with the chosen activation function.

4.6.3 Fitness Function based on ANN

The fitness function serves as a key tool for determining how well the obtained solution represents the behavior of the Lorenz system. It measures the difference between the estimated results and the actual values of the system. This function directs the optimization process in both Particle Swarm Optimization (PSO) and the Active Set Algorithm by helping to identify more accurate solutions over time. A smaller fitness value reflects a more precise solution, guiding the algorithms toward accurately capturing the dynamics of the Lorenz model. For solving the Lorenz model, the fitness function is formulated as

$$\varepsilon_f = \varepsilon_x + \varepsilon_y + \varepsilon_z + \varepsilon_{ic}, \tag{4.11}$$

where ε_x , ε_y , ε_z , ε_{ic} from lorenz model are given as

$$\varepsilon_x = \left(\frac{d\hat{x}}{dt} - \sigma(\hat{x}(t) - \hat{y}(t))\right)^2, \tag{4.12}$$

$$\varepsilon_y = \left(\frac{d\hat{y}}{dt} - \hat{y}(t) + \rho\hat{z}(t) - \hat{x}(t)\hat{z}(t)\right)^2, \tag{4.13}$$

$$\varepsilon_z = \left(\frac{d\hat{z}}{dt} + \beta\hat{z}(t) - \hat{x}(t)\hat{y}(t)\right)^2, \tag{4.14}$$

$$\varepsilon_{ic} = \left((\hat{x}(t) - \hat{x}_0)^2 + (\hat{y}(t) - \hat{y}_0)^2 + (\hat{z}(t) - \hat{z}_0)^2\right), \tag{4.15}$$

The modified form of Eq. 4.11 is given as:

$$\begin{aligned} \varepsilon_f = &\left(\frac{d\hat{x}}{dt} - 0.1(\hat{x} - \hat{y})\right)^2 + \left(\frac{d\hat{y}}{dt} - \hat{y} + 0.3(\hat{z} - \hat{x}\hat{z})\right)^2 \\ &+ \left(\frac{d\hat{z}}{dt} - 0.2(\hat{z} + \hat{x}\hat{y})\right)^2 \quad + \left((\hat{x}(t) - \hat{x}_0)^2 + (\hat{y}(t) - \hat{y}_0)^2 + (\hat{z}(t) - \hat{z}_0)^2\right. \end{aligned} \tag{4.16}$$

4.6.4 Optimization Algorithms : PSO-ASA

In order to solve a non-linear Lorenz Model, this study combined an unsupervised artificial neural network with global optimzer that is PSO and ASA which is local optimizer, that are utilized to train the unknown adjustable parameter of ANNs. The primary goal is to determine the ideal ANN weights and biases. When traditional optimization approaches are unable to solve a problem because of the complexity or non-linearity of the objective functions and constraints involved, these optimization algorithms are made to find approximate solutions.

4.6.5 Particle Swarm Optimization (PSO) Algorithms

PSO is an evolutionary worldwide-search method within big multifaceted resolution gaps. It is an engineering model of swarm intelligence that draws inspiration from insect swarms, fish schools, and flocks of birds [66]. Kennedy and Ehrhart created the PSO's initial mathematical formulation [67]. The primary goal of PSO is to examine the nonlinear complex optimization issues that are challenging to resolve with conventional method. PSO has proven to be very helpful across the domains of robotics, wireless networks, load management, and power systems. Resource allocation and network parameter settings have been optimized within these fields. Researchers have applied PSO to a variety of challenging non-linear problems [68].

1. Initialization

   Choose N, the population size, or the number of particles.

   Initialize the position and velocity of each particle in the search space $(1 \leq i \leq N)$ at random.

   Assign the current position of each particle to its initial best position (pbest).

   As the best of all particle positions, set the global best position (gbest).

Particle Position and Velocity Update:
Regarding every particle $(1 \leq i \leq N)$:
The velocity is updated using the following formula:

$$u_{t+1} = u_t + \varphi_1(B - X) + \varphi_2(g - X), \tag{4.17}$$

where B is the particle's best-known position, g is the global best position, u_t is the particle's current velocity, and u_{t+1} is the updated velocity at time t. Local and global influences are taken into consideration by the terms $\varphi_1(B-X)$ and $\varphi_2(g-X)$, respectively.

Refresh the particle's location with:

$$X_{t+1} = X_t + u_{t+1}. \tag{4.18}$$

2. Assessing and Revising the Best Positions:

Regarding every particle $(1 \leq i \leq N)$:

Calculate the value of the goal function or fitness at its new location.

If the new position has a higher fitness value than its prior best position, update pbest .

If the new position yields a better result than the current global best, update the global best (gbest).

Repeat steps 2 and 3 until a termination condition is met, such as completing as many iterations as possible or obtaining the target level of fitness. By continuously updating the particles' positions and velocities based on their personal best and the global best positions discovered by the swarm, the algorithm aims to determine the most effective solution. The PSO flow is given in Fig.4.3

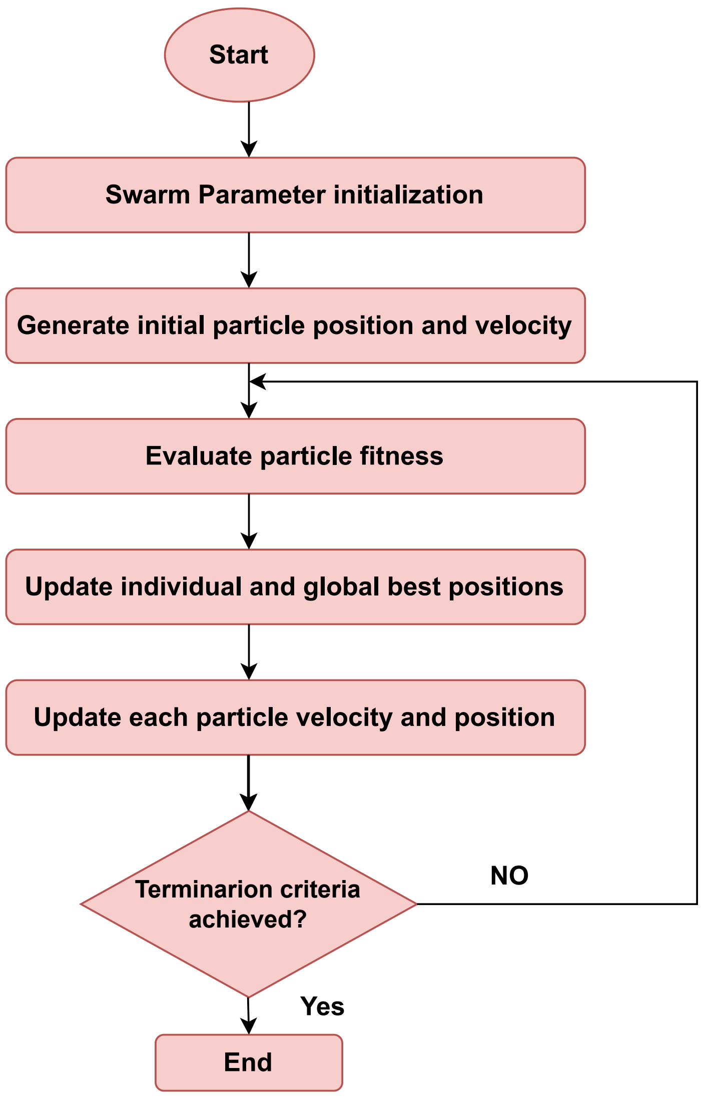

Start
Swarm Parameter initialization
Generate initial particle position and velocity
Evaluate particle fitness
Update individual and global best positions
Update each particle velocity and position
Terminarion criteria achieved?
NO
Yes
End

Figure 4.3: PSO Flowchart

4.6.6 Active Set Algorithm (ASA)

The Active Set Algorithm (ASA) is a numerical optimization technique used in constrained optimization problems, particularly in training neural networks, despite not being a standalone solution. It is a local optimizer. Standard optimization algorithms in neural networks don't explicitly deal with constraints, but ASA can be useful when network parameters are explicitly imposed or training involves constrained optimization problems. ASA enforces sparsity in neural networks, ensuring parameter updates respect constraints for constrained optimization tasks, and ensuring training with bounded weights for specific ranges of weights or biases.The flow of ASA is given in Fig.4.4.

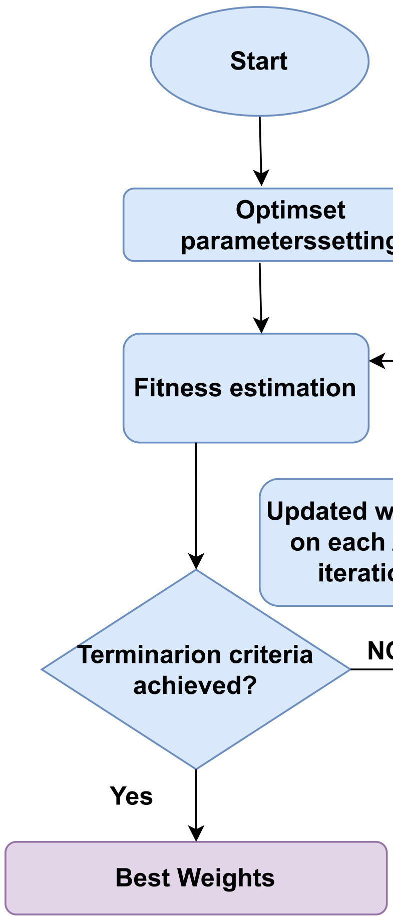

Start
Optimset parameterssetting
Fitness estimation
Updated weights on each ASA iteration
Terminarion criteria achieved?
NO
Yes
Best Weights

Figure 4.4: ASA Flowchart

4.6.7 Hybrid PSO-ASA Algorithm

To introduce a new hybrid approach, this study integrates an unsupervised ANN with two optimization methods: PSO as the global optimizer and Active Set Algorithm (ASA) as the local optimizer. The primary goal is to determine the ideal ANN weights and biases. Following are the steps involved:

- **Step 1:** Establish the Objective Function. Explain the fitness function that ANN forecasts utilize to reduce error.

- **Step 2:** Set the allowed upper and lower limits for the adjustable parameters of the ANN.These bounds ensure the stability of the network and keep the optimization process within acceptable ranges.
- **Step 3:** Put Particle Swarm Optimization (PSO) into Practice. Set up velocities, particles, and fitness calculations.
  Particle placements are updated according to individual and global best solutions.
  Boundary constraints should be used to avoid too much parameter drift.
  Keep iterating until either the maximum iteration count is met or convergence occurs.
- **Step 4**: Use Active Set Algorithm (ASA) for Local Refinement to increase precision, use ASA to fine-tune the PSO-optimized solution. Describe the function of ASA in constrained problem optimization.
- **Step 5:** Save the optimized parameters in step five. Save the final ANN parameters that have been optimized for analysis and validation.

MATLAB software tool help with the whole optimization process.

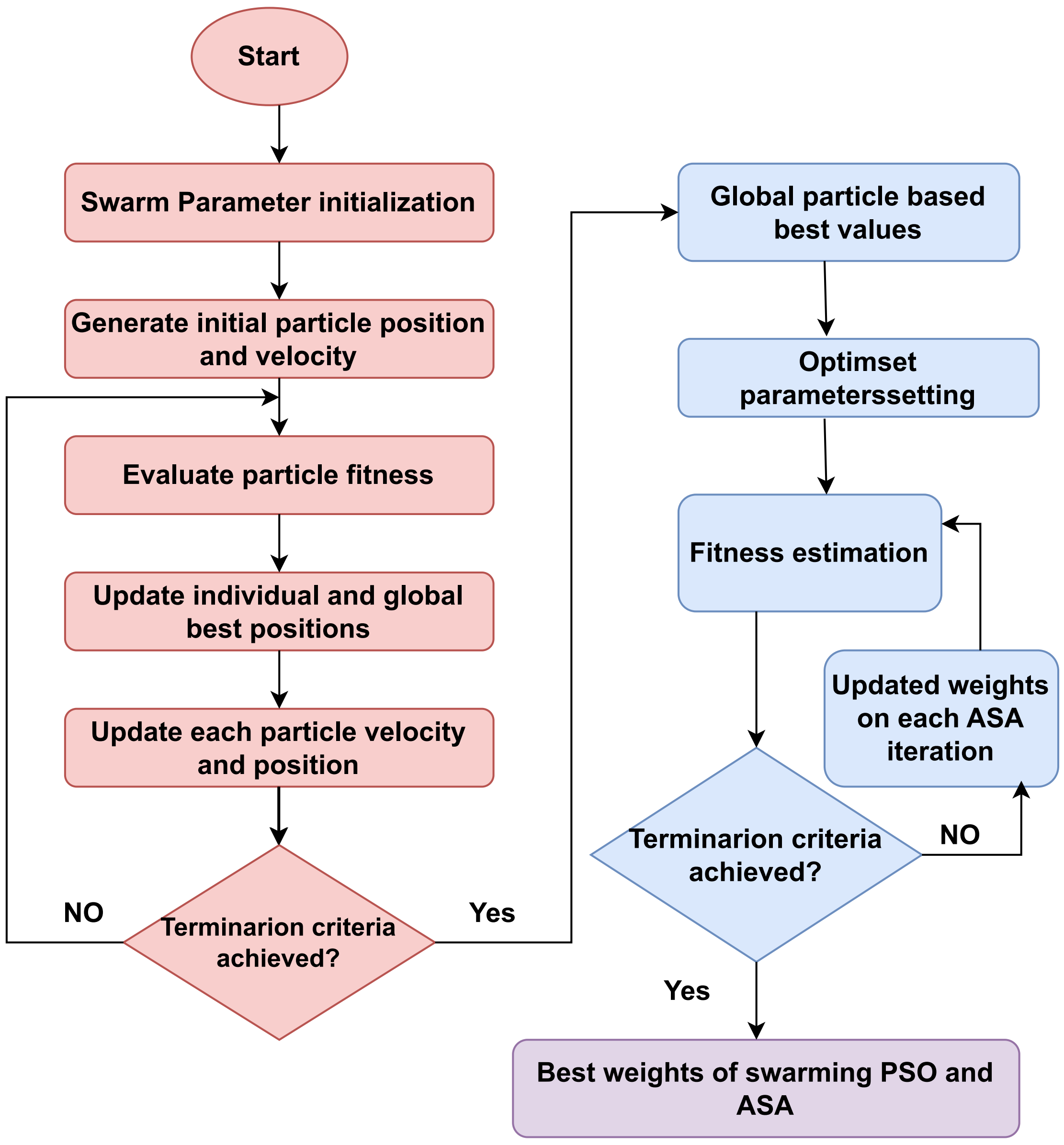

Start
Swarm Parameter initialization
Generate initial particle position and velocity
Evaluate particle fitness
Update individual and global best positions
Update each particle velocity and position
Terminarion criteria achieved?
NO
Yes
Global particle based best values
Optimset parameterssetting
Fitness estimation
Updated weights on each ASA iteration
Terminarion criteria achieved?
NO
Yes
Best weights of swarming PSO and ASA

Figure 4.5: Hybrid PSO-ASA Flowchart

4.6.8 Performace Measures

In order to evaluate the effectiveness of the model, performance metrics such as Absolute Error (AE), Mean Square Error (MSE), and Mean Absolute Deviation (MAD) are computed, defined as follows:

$$\text{Absolute Error} = \sum_{i=1}^{n} \left(|x_i - \hat{x}_i|, |y_i - \hat{y}_i|, |z_i - \hat{z}_i| \right). \tag{4.19}$$

$$\text{Mean Square Error} = \frac{1}{n}\sum_{i=1}^{n}\left[(x_i-\hat{x}_i)^2, (y_i-\hat{y}_i)^2, (z_i-\hat{z}_i)^2\right]. \tag{4.20}$$

$$\text{Mean Absolute Deviation} = \frac{1}{n}\sum_{i=1}^{n}\left(|x_i-\hat{x}_i|, |y_i-\hat{y}_i|, |z_i-\hat{z}_i|\right). \tag{4.21}$$

4.7 Result and Discussion

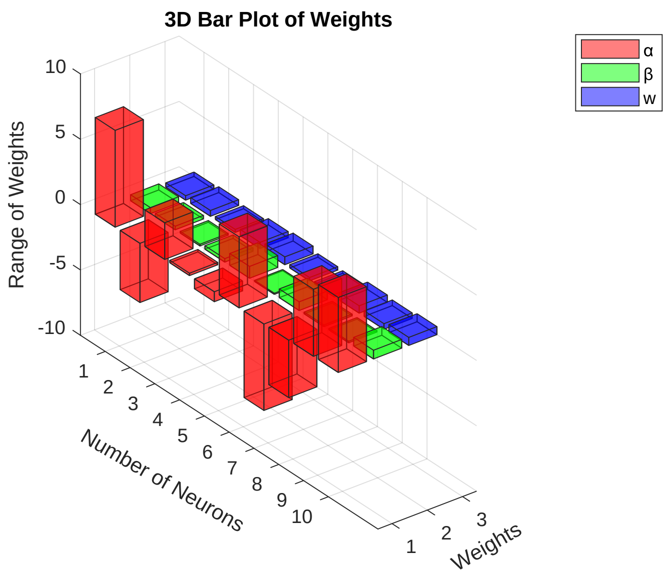
An Artificial Neural Network (ANN) framework was created for this study in order to get a numerical solution for the nonlinear coupled differential model known as the Lorenz system. The artificial neural network (ANN) was developed to lower the error between the expected and exact solutions by transforming the model equations into an unsupervised learning problem. The ANN was developed using MATLAB because of its strong computational capabilities and integrated tools for neural network modeling and optimization. Three layers make up the network: an input layer, a hidden layer that employs the Mexican Hat wavelet as the activation function, and an output layer that provides the values x(t), y(t), and z(t). To find out the best parameters for the network, combined method is applied which integrates particle swarm optimization and active set algorithm. For the determination of optimal settings, this integrated method worked effectively. In our research it has been explained that use of artificial neural network based numerical approach can accurately depicting the lorenz system unpredictable behavior, especially when chosen right parameters. This method presents reliable and efficient alternative to tackle non linear dynamical systems for some of those situations where traditional techniques were having difficulties for implementation.

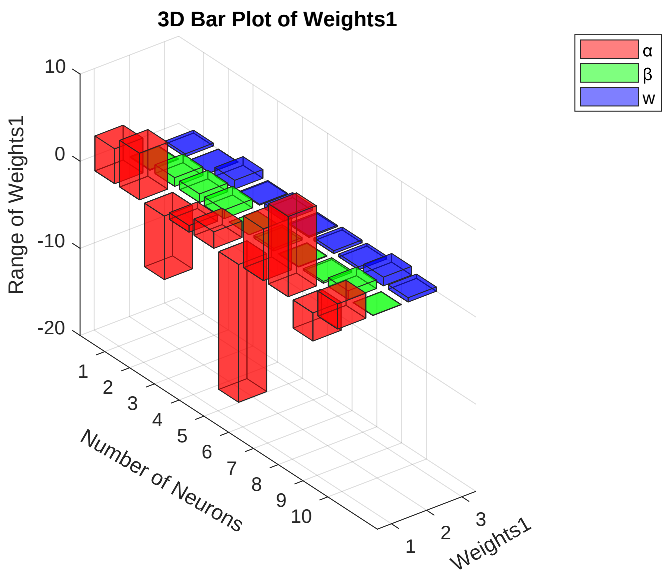
$$\begin{aligned}
\frac{d\hat{x}}{dt} &= 0.1(\hat{y}-\hat{x}),\\
\frac{d\hat{y}}{dt} &= -\hat{y}+0.3(\hat{z}-\hat{x}\hat{z}),\\
\frac{d\hat{z}}{dt} &= -0.2(\hat{z}+\hat{x}\hat{y}).
\end{aligned} \tag{4.22}$$

$$x(0)=0, \qquad y(0)=1, \qquad z(0)=0. \tag{4.23}$$

$$\begin{aligned}
\varepsilon_f = {} & \left(\frac{d\hat{x}}{dt}-0.1(\hat{x}-\hat{y})\right)^2 + \left(\frac{d\hat{y}}{dt}-\hat{y}+0.3(\hat{z}-\hat{x}\hat{z})\right)^2 \\
& + \left(\frac{d\hat{z}}{dt}-0.2(\hat{z}+\hat{x}\hat{y})\right)^2 \quad + \left((\hat{x}(0)-0)^2+(\hat{y}(0)-1)^2+(\hat{z}(0)-0)^2\right).
\end{aligned} \tag{4.24}$$

Combined approach includes active set algorithm (ASA)and PSO, these were used to solve non-linear lorenz system to optimize fitness function. This hybrid method PSO-ASA was carried out 50 times by using neural network with 10 hidden neurons and 90 optimal weights for each output variable. The network get familiar with system's behavior so provide best results by following the process. Evaluation and comparison of three approaches has been done in the study which are the new Mexican hat wavelet based ANN(MHW- ANN-PSO-ASA), The ANN trained with PSO-ASA, the conventional Runge Kutta RK solver. For the demonstration of accuracy and reliability of each approach, evaluation uses average and best fitness values from multiple runs.





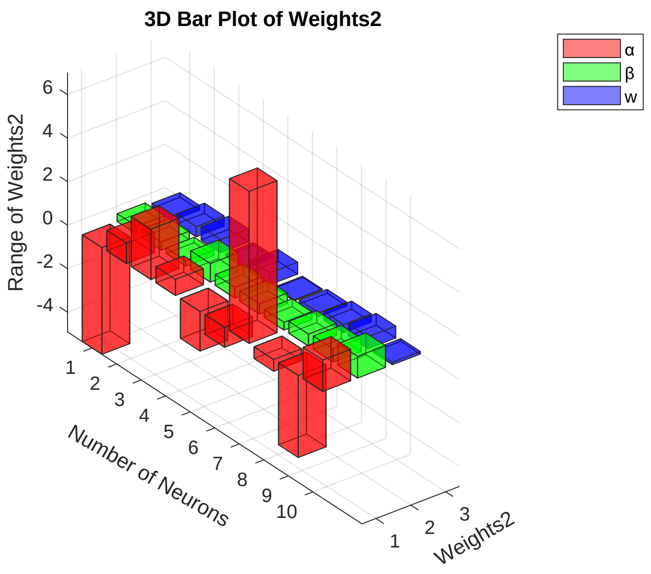


Figure 4.6: Optimized weights through ANN-PSO-ASA

We Compared the outcomes of our study which include Runge-Kutta (RK) numerical solver and the suggested MHW-ANN-PSO-ASA methods. These outcomes were obtained by using numerical approach and suggested ANN model. Thorough comparison is being provided in table 4.1, 4.2 and 4.3. The dichotomous values closely correlates which shows that ANN is providing reliable and accurate answer and the consistent outcomes show consistency of ANN based approach. By matching different values, the model's reliability is being demonstrated. In conclusion, ANN approach depicts the reliable behavior of numerical solution as shown in the table.

t	x_{ANN}	$x_{\text{Numerical}}$	Error
0.00	7.18×10^{-5}	0	7.18×10^{-5}
0.10	0.009515197	0.009468509	4.67×10^{-5}
0.20	0.018000418	0.017944442	5.60×10^{-5}
0.30	0.025599685	0.025525549	7.41×10^{-5}
0.40	0.032386080	0.032299963	8.61×10^{-5}
0.50	0.038432375	0.038347143	8.52×10^{-5}
0.60	0.043809945	0.043738725	7.12×10^{-5}
0.70	0.048587810	0.048539300	4.85×10^{-5}
0.80	0.052831835	0.052807101	2.47×10^{-5}
0.90	0.056604072	0.056594642	9.43×10^{-6}
1.00	0.059962247	0.059949279	1.30×10^{-5}

Table 4.1: Comparison of ANN and Numerical Solutions of $x(t)$ in the Lorenz Model

t	y_{ANN}	$y_{\text{Numerical}}$	Error
0.00	0.999916942	1	8.31×10^{-5}
0.10	0.905097711	0.904977299	1.20×10^{-4}
0.20	0.819079001	0.819251583	1.73×10^{-4}
0.30	0.741344017	0.741908769	5.65×10^{-4}
0.40	0.671299694	0.672123937	8.24×10^{-4}
0.50	0.608299522	0.609153155	8.54×10^{-4}
0.60	0.551665609	0.552325839	6.60×10^{-4}
0.70	0.500709145	0.501037699	3.29×10^{-4}
0.80	0.454748568	0.454744272	4.30×10^{-6}
0.90	0.413125004	0.412955031	1.70×10^{-4}
1.00	0.37521474	0.37522804	1.33×10^{-5}

Table 4.2: Comparison of ANN and Numerical Solutions of $y(t)$ in the Lorenz Model

t	z_{ANN}	$z_{\text{Numerical}}$	Error
0.00	5.80×10^{-5}	0	5.80×10^{-5}
0.10	0.000541195	0.000448245	9.29×10^{-5}
0.20	0.001632297	0.001609982	2.23×10^{-5}
0.30	0.003198362	0.003257724	5.94×10^{-5}
0.40	0.005111798	0.005216385	1.05×10^{-4}
0.50	0.007253367	0.007352457	9.91×10^{-5}
0.60	0.009514705	0.009565315	5.06×10^{-5}
0.70	0.011800273	0.011780249	2.00×10^{-5}
0.80	0.014028679	0.013942877	8.58×10^{-5}
0.90	0.016133418	0.016014693	1.19×10^{-4}
1.00	0.018063019	0.017969504	9.35×10^{-5}

Table 4.3: Comparison of ANN and Numerical Solutions of $z(t)$ in the Lorenz Model

The ANN and numerical solutions are compared in the graphs for the Lorenz system variables $x(t)$, $y(t)$, and $z(t)$.While the numerical solutions are indicated with different symbols (red stars for $x(t)$, green stars for $y(t)$, and blue stars for $z(t)$), the ANN solutions are displayed as smooth curves in each graph. While $y(t)$ displays a downward trend, the graphs of $x(t)$ and $z(t)$ show an upward trend. The accuracy of the ANN model in solving the Lorenz system is demonstrated by the close alignment of the ANN and numerical results in each of the three graphs. This consistency demonstrates how well the Mexican Hat Wavelet ANN models chaotic dynamics.

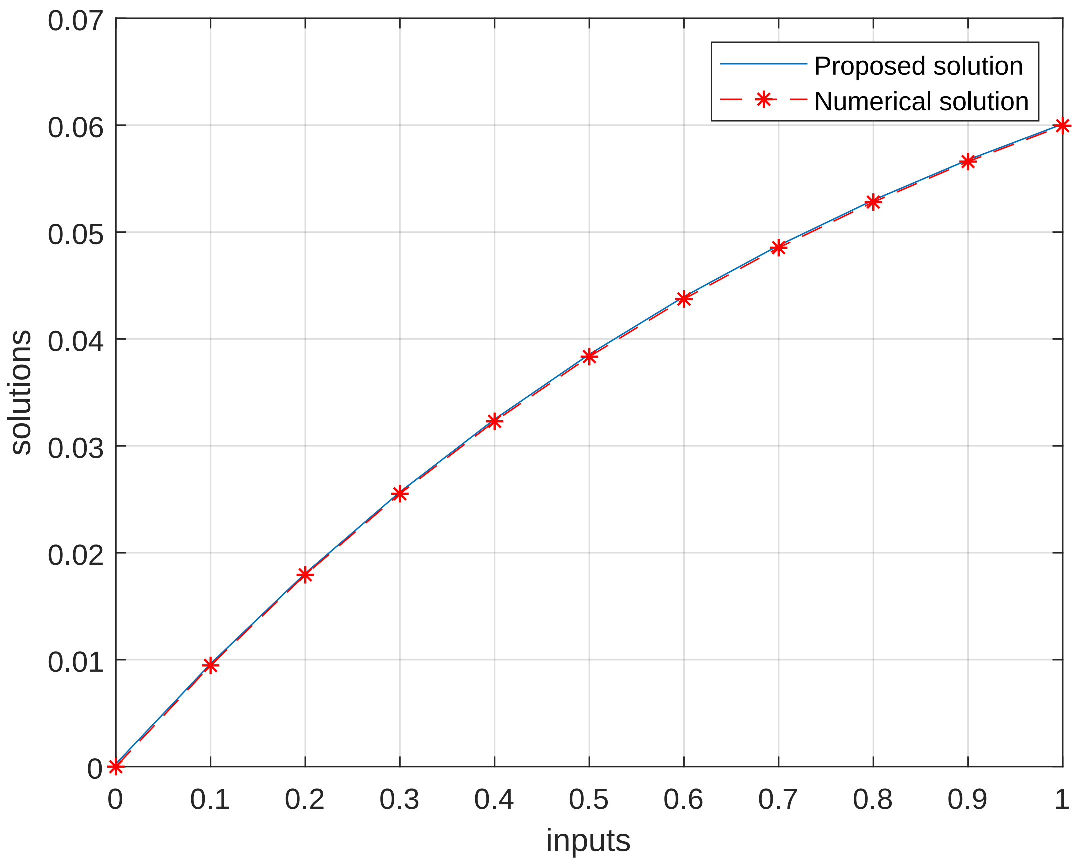


Figure 4.7: Comparison solutions for x(t).

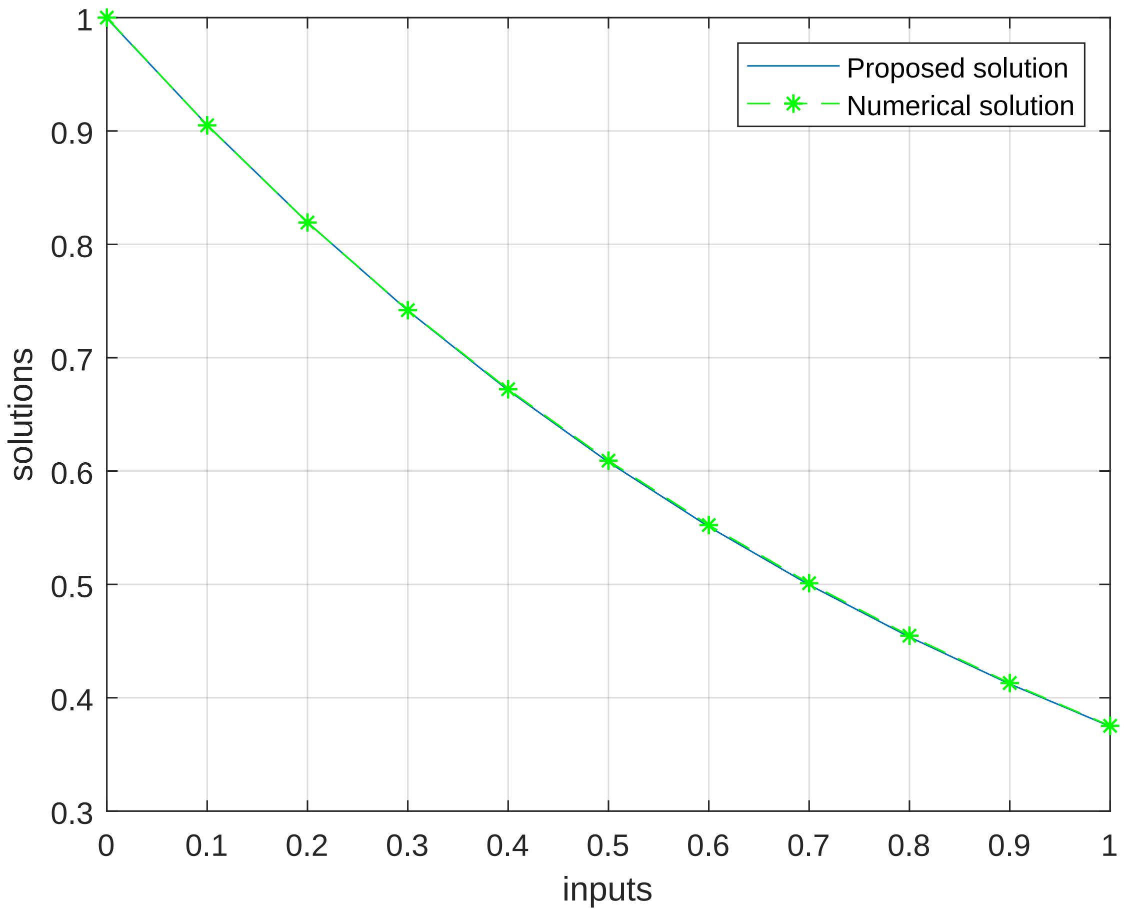


Figure 4.8: Comparison solutions for y(t).

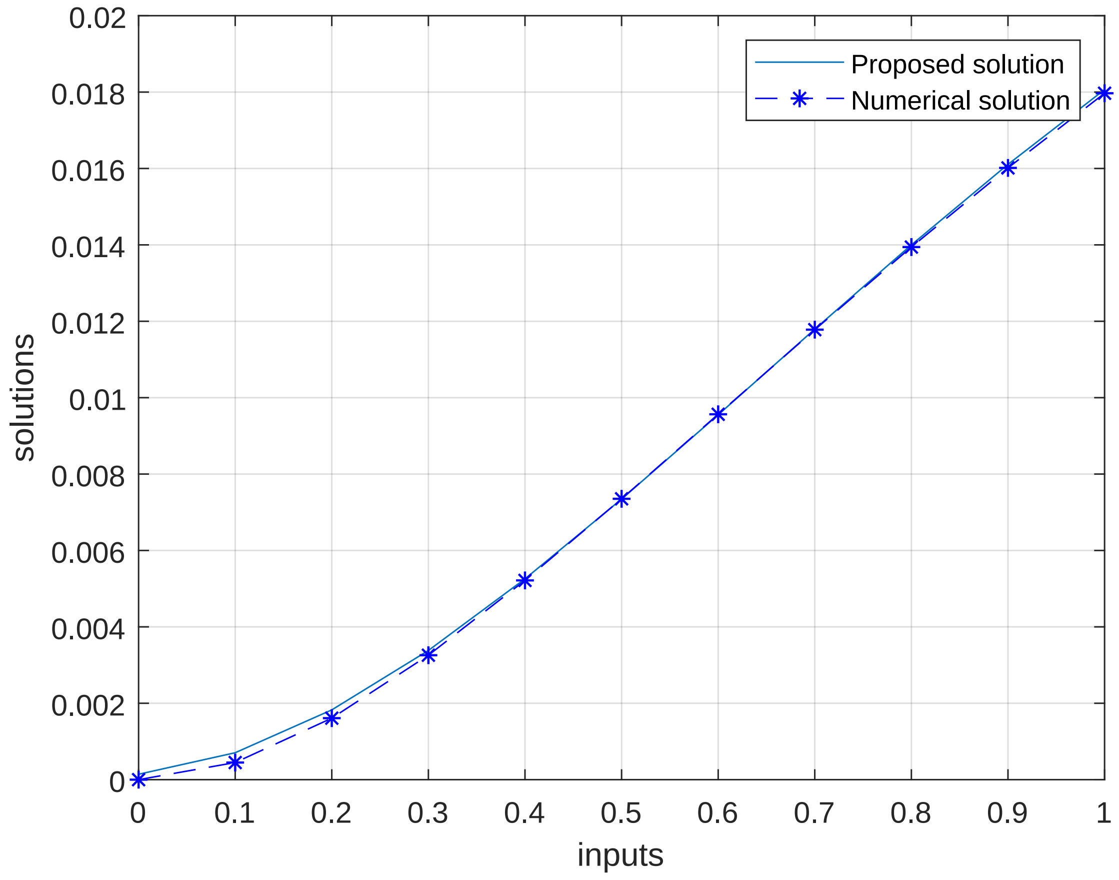


Figure 4.9: Comparison solutions for z(t).

Absolute error numbers were calculated using both mean and best-case situations in order to verify the accuracy of the suggested scheme. The results indicate that the MHW-ANN-PSO-ASA approach is an effective tool for solving complex nonlinear systems because it produces extremely precise solutions for the Lorenz system. This further shows how wavelet activation functions and hybrid optimization increase the ANN's performance.

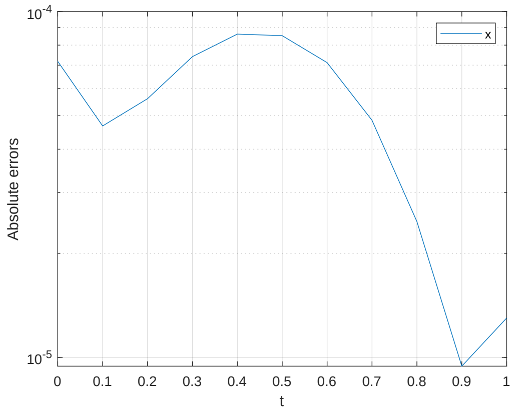


Figure 4.10: Absolute Error between ANN-PSO-ASA and Numerical solution for x.

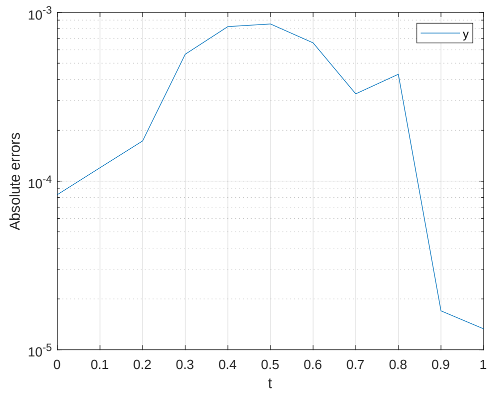


Figure 4.11: Absolute Error between ANN-PSO-ASA and Numerical solution for y.

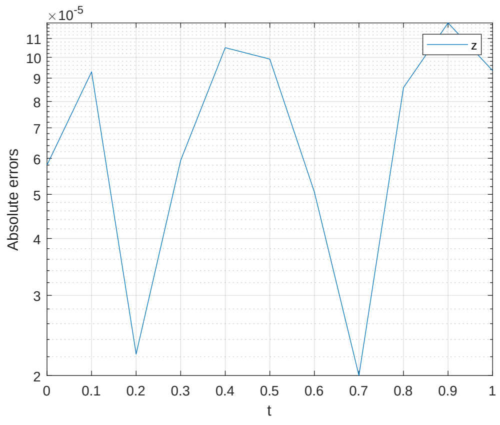


Figure 4.12: Absolute Error between ANN-PSO-ASA and Numerical solution for z.

The absolute error for x is very small, that is, between 10^{-6} and 10^{-5}, so we can do the comparison of numerical techniques' findings and ANN-based solutions. The absolute errors for y and z are also very small — for y, it is between 10^{-6} and 10^{-4}, and for z it is between 10^{-5} and 10^{-4}. The results of absolute errors show that the results are very precise and accurate. The mean square error is calculated for 50 runs in order to evaluate the effectiveness and accuracy of the Lorenz model.

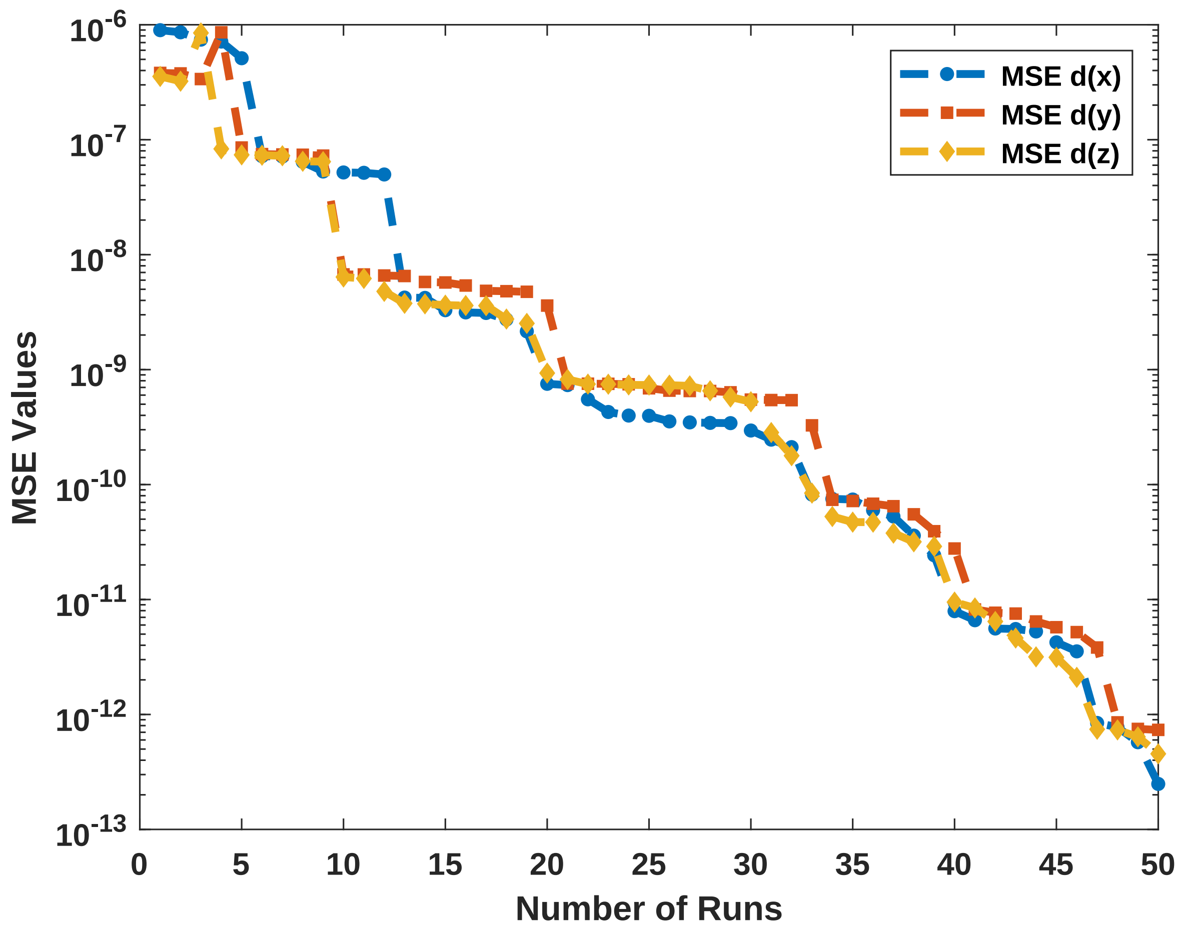


Figure 4.13: Mean Square Error

This graph illustrates the findings for the Mean Squared Error (MSE) of the Lorenz system's x, y and z components over 50 runs. The error values ranges between 10^{-13} to 10^{-6}. This graph shows that the neural network (NN) has a low MSE for all components, with the x-component having the lowest error at 10^{-13} , followed by the y-component coming in second. These findings tell us that the y-component is more difficult to model due to its comparatively higher error. Nevertheless, the general pattern demonstrates strong convergence and extremely low deviation across runs, highlighting the stability and dependability of the model. These outcomes show the effectiveness of the ANN model and verify its ability to solve the nonlinear Lorenz system using the Mexican Hat wavelet as the activation function.

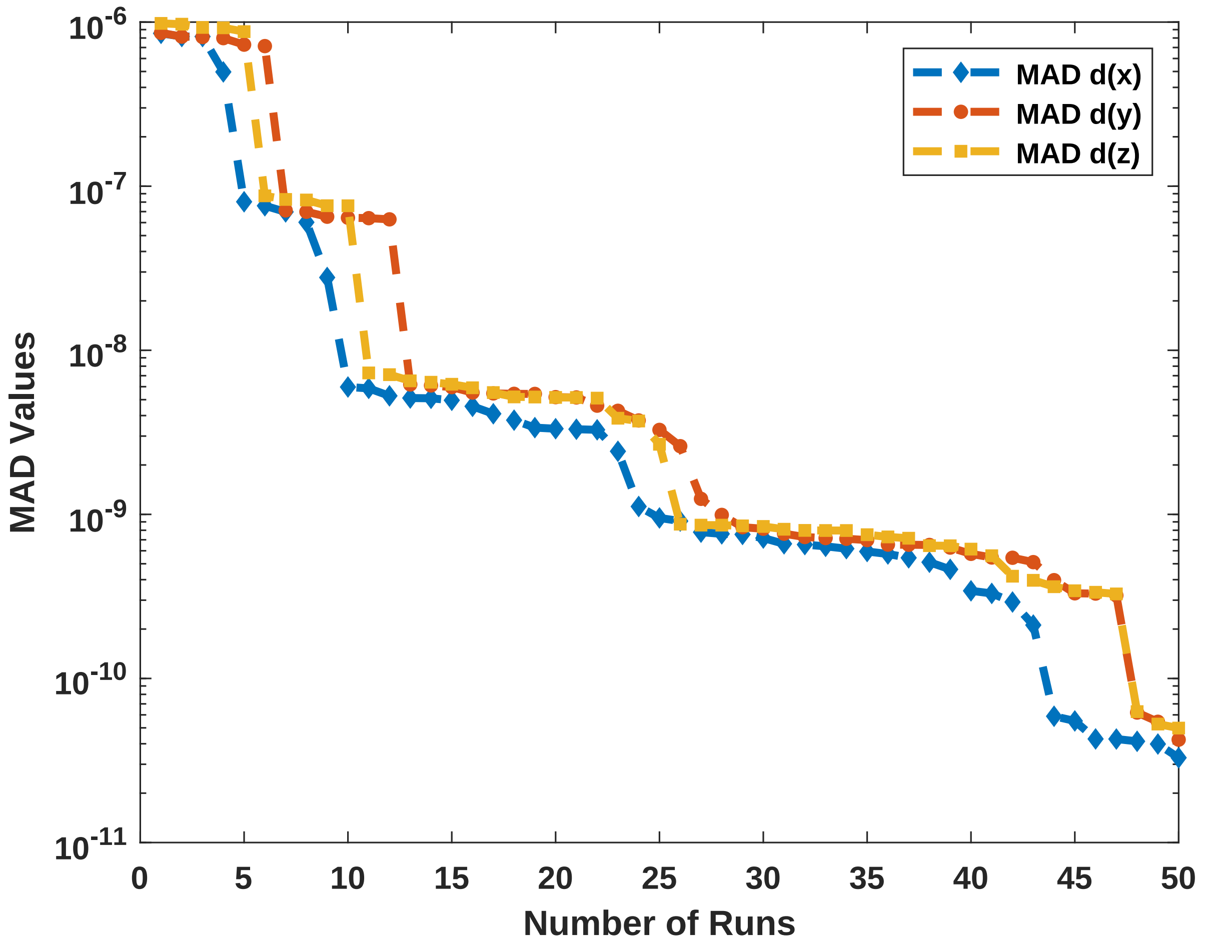


Figure 4.14: Mean Absolute Deviation

This graph represents the mean Absolute Deviation (MAD) for the Lorenz system's x, y, and z components over 50 optimization runs. The graph illustrates that NN is stable as well as accurate as the MAD $d(x)$, MAD $d(y)$, and MAD $d(z)$, remain low throughout the runs ranging from 10^{-11} to 10^{-6} on a logarithmic scale. MAD predicts the behavior of the x-component accurately, as the error value of MAD $d(x)$ is the lowest among all close to 10^{-11}. Although the MAD error $d(z)$ ranges from 10^{-8} to 10^{-7} and this represent it is difficult to model. MAD $d(y)$ produces some good results. This extremely low deviation in all three components indicates that the model is reliable. These findings show that the neural network, which uses the Mexican Hat wavelet activation function, is a dependable and precise method for solving the Lorenz system.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In order to solve the Lorenz system, this thesis presents an effective framework that is the Mexican Hat-based ANN model. The developed approach used Mexican Hat as an activation function and PSO-ASA to optimize weights. The Mexican Hat is used because of its ability to handle the chaotic behavior of differential equations and its ability to effectively represent localized behaviors. The neural network improves its ability to approximate nonlinear interactions and the complex interdependencies characteristic of the Lorenz system by using the Mexican Hat.

This study represents the hybrid optimization technique that integrates the local optimization power of ASA with the global optimization technique of PSO to find the solution of the Lorenz system. This approach combines the ASA with PSO to optimize the weights and basis of the Mexican Hat ANN model instead of using traditional training techniques. The hybrid PSO-ASA technique increases the accuracy by minimizing the error function and guiding the network towards an optimal solution.

In order to evaluate the accuracy and performance of the MHW-DEANN model, Mean Squared Error (MSE) and Mean Absolute Deviation (MAD) are calculated over 50 runs. The model exhibited low error values. It can be seen that the x-component shows the least deviation among all three variables of the Lorenz system. The accuracy and stability of the model are demonstrated

by the MAD performance.
The proposed ANN solution is verified through Absolute Error Analysis and graphical comparison with the RK-4 solution. The curves generated by the ANN solution overlap with the traditional numerical solution for all three components $x(t)$, $y(t)$ and $z(t)$. This overlap confirms the ability of the ANN model to produce accurate results for the Lorenz system.
The proposed ANN model, using the Mexican Hat as an activation function and with weights optimized through PSO-ASA, is an alternative and precise approach to solving the Lorenz system.The MHW-DEANN model is more stable and consistent, making it more suitable for solving differential equations and complex systems where numerical solvers struggle due to the complexity of the equations and the difficulty in managing chaotic behavior.

5.2 Future Work

The proposed MHW-DEANN model provides accurate and high-precision results when solving the Lorenz system. In order to evaluate the generalization of the MHW-DEANN approach for different nonlinear differential equations, its application to other models such as the Rössler, Chen, or Lü systems still needs to be studied. This approach can be implemented to approximate the solutions of more complex nonlinear systems in engineering and physical sciences, such as electrical networks, ecological systems, and thermodynamic processes. Future investigations involving other wavelet functions such as Haar, Daubechies, or Morlet may also reveal how their characteristics affect performance and accuracy.
To obtain better stability and minimize computational burden, the current hybrid optimization technique PSO-ASA can be replaced with other algorithms such as Ant Colony Optimization (ACO), Firefly Algorithm (FA), or Genetic Algorithms (GA). More complex systems, such as those with time delays, fractional dynamics, or higher-order nonlinear behavior, may also be handled by expanding the framework. This ANN model can be applied to real-world situations requiring online control or modeling of chaotic processes by deploying it onto real-time hardware platforms such as FPGAs or microcontrollers, thereby bridging the gap between theory and practice.

BIBLIOGRAPHY

[1] E. N. Lorenz, “Deterministic nonperiodic flow,” *Journal of Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[2] N. A. Kudryashov, “Analytical solutions of the lorenz system,” *Regular and Chaotic Dynamics*, vol. 20, pp. 123–133, 2015.

[3] L. Bougoffa, S. Al-Awfi, and S. Bougouffa, “A complete and partial integrability technique of the lorenz system,” *Results in Physics*, vol. 9, pp. 712–716, 2018.

[4] A. Algaba, M. Domínguez-Moreno, M. Merino, and A. J. Rodríguez-Luis, “Double-zero degeneracy and heteroclinic cycles in a perturbation of the lorenz system,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 111, p. 106482, 2022.

[5] E. J. Doedel, B. Krauskopf, and H. M. Osinga, “Global invariant manifolds in the transition to preturbulence in the lorenz system,” *Indagationes Mathematicae*, vol. 22, no. 3-4, pp. 222–240, 2011.

[6] G. Wu, L. Tang, and J. Liang, “Synchronization of non-smooth chaotic systems via an improved reservoir computing,” *Scientific Reports*, vol. 14, no. 1, p. 229, 2024.

[7] I. Alexeev, “Lorenz system in the thermodynamic modelling of leukaemia malignancy,” *Medical Hypotheses*, vol. 102, pp. 150–155, 2017.

[8] S. Krishnan and S. Malathy, “Solving lorenz system of equation by laplace homotopy analysis method,” in *Proceedings of the First International Conference on Combinatorial and Optimization, ICCAP*, 2021, pp. 7–8.

[9] B. Zlatanovska and B. Piperevski, “A particular solution of the third-order shortened lorenz system via integrability of a class of differential equations,” *Asian-European Journal of Mathematics*, vol. 15, no. 10, p. 2250242, 2022.

[10] M. Klöwer, P. V. Coveney, E. A. Paxton, and T. N. Palmer, “Periodic orbits in chaotic systems simulated at low precision,” *Scientific Reports*, vol. 13, no. 1, p. 11410, 2023.

[11] M. N. Aslam, M. W. Aslam, M. S. Arshad, Z. Afzal, M. K. Hassani, A. M. Zidan, and A. Akgül, "Neuro-computing solution for lorenz differential equations through artificial neural networks integrated with pso-nna hybrid meta-heuristic algorithms: A comparative study," *Scientific Reports*, vol. 14, no. 1, p. 7518, 2024.

[12] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 42, no. 1, pp. A292–A317, 2020.

[13] F. M. Riaz, S. Ahmad, J. A. Khan, S. Altaf, Z. U. Rehman, and S. K. Memon, "Numerical treatment of non-linear system for latently infected cd4+ t cells: a swarm-optimized neural network approach," *IEEE Access*, vol. 12, pp. 103 119–103 132, 2024.

[14] S. A. Bhat, Z. Sabir, M. A. Z. Raja, T. Saeed, and A. M. Alshehri, "A novel heuristic morlet wavelet neural network procedure to solve the delay differential perturbed singular model," *Knowledge-Based Systems*, vol. 292, p. 111624, 2024.

[15] M. El Jaadi, T. Haidi, A. Belfqih, M. Farah, and A. Dialmy, "Optimizing wind farm layout for enhanced electricity extraction using a new hybrid pso-ann method," *Global Energy Interconnection*, vol. 7, no. 3, pp. 254–269, 2024.

[16] M. Mattheakis, D. Sondak, A. S. Dogra, and P. Protopapas, "Hamiltonian neural networks for solving equations of motion," *Physical Review E*, vol. 105, no. 6, p. 065305, 2022.

[17] A. Abidi, A. I. Khdair, and R. Kalbasi, "Using ann techniques to forecast thermal performance of a vacuum tube solar collector filled with sio2/eg-water nanofluid," *Journal of the Taiwan Institute of Chemical Engineers*, vol. 128, pp. 301–313, 2021.

[18] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 42, no. 1, pp. A292–A317, 2020.

[19] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.

[20] T. Hagge, P. Stinis, E. Yeung, and A. M. Tartakovsky, "Solving differential equations with unknown constitutive relations as recurrent neural networks," *arXiv preprint arXiv:1710.02242*, 2017.

[21] F. M. Riaz, R. M. S. Ullah, A. Alasiry, M. Marzougui, and J. A. Khan, “Swarm-optimized numerical investigation of dengue fever model,” *Soft Computing*, pp. 1–17, 2025.

[22] M. Shoaib, R. Tabassum, M. A. Z. Raja, and K. S. Nisar, “Bio-inspired algorithm integrated with sequential quadratic programming to analyze the dynamics of hepatitis b virus,” *Beni-Suef University Journal of Basic and Applied Sciences*, vol. 13, no. 1, p. 71, 2024.

[23] S. A. Bhat, Z. Sabir, M. A. Z. Raja, T. Saeed, and A. M. Alshehri, “A novel heuristic morlet wavelet neural network procedure to solve the delay differential perturbed singular model,” *Knowledge-Based Systems*, vol. 292, p. 111624, 2024.

[24] J. Tang, J. Zhu, and Z. Sun, “A novel path planning approach based on appart and particle swarm optimization,” in *International Symposium on Neural Networks*. Springer, 2005, pp. 253–258.

[25] E. Pires, P. Oliveira, J. Tenreiro Machado, and J. B. Cunha, “Particle swarm optimization versus genetic algorithm in manipulator trajectory planning,” in *CONTROLO 2006-7th Portuguese Conference on Automatic Control*, 2006, pp. 1–7.

[26] M. Alrashidi and M. El-Hawary, “A survey of particle swarm optimization applications in power system operations,” *Electric Power Components and Systems*, vol. 34, no. 12, pp. 1349–1357, 2006.

[27] M. S. Couceiro, J. M. A. Luz, C. M. Figueiredo, N. F. Ferreira, and G. Dias, “Parameter estimation for a mathematical model of the golf putting,” in *Proceedings of WACI-Workshop Applications of Computational Intelligence. ISEC. IPC. Coimbra*, vol. 2, 2010, pp. 1–8.

[28] E. Solteiro Pires, J. Tenreiro Machado, P. de Moura Oliveira, J. Boaventura Cunha, and L. Mendes, “Particle swarm optimization with fractional-order velocity,” *Nonlinear Dynamics*, vol. 61, pp. 295–301, 2010.

[29] Y. Del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, “Particle swarm optimization: Basic concepts, variants and applications in power systems,” *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, 2008.

[30] F. M. Riaz, J. A. Khan, W. Bouchelligua, and H. Alshaya, “Efficient computational modeling of leptospirosis: A hybrid neuro heuristic perspective,” 2025.

[31] F. M. Riaz and J. A. Khan, “A swarm optimized ann-based numerical treatment of nonlinear seir system based on zika virus,” *Politeknik Dergisi*, pp. 1–1.

[32] P. H. Winston, *Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1992.

[33] S. A. Kalogirou, “Artificial neural networks in renewable energy systems applications: a review,” *Renewable and Sustainable Energy Reviews*, vol. 5, no. 4, pp. 373–401, 2001.

[34] N. Yadav, A. Yadav, M. Kumar *et al.*, *An Introduction to Neural Network Methods for Differential Equations*. Springer, 2015, vol. 1.

[35] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[36] J. Von Neumann, “The general and logical theory of automata,” in *Systems Research for Behavioral Science*. Routledge, 2017, pp. 97–107.

[37] ——, “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” *Automata Studies*, vol. 34, no. 34, pp. 43–98, 1956.

[38] F. Rosenblatt *et al.*, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books Washington, DC, 1962, vol. 55.

[39] M. Minsky and S. Papert, “Perceptrons," mit press, cambridge, ma,” 1969.

[40] S. Amari, “A theory of adaptive pattern classifiers,” *IEEE Transactions on Electronic Computers*, no. 3, pp. 299–307, 1967.

[41] K. Fukushima, “Visual feature extraction by a multilayered network of analog threshold elements,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 4, pp. 322–333, 1969.

[42] S. Grossberg, “Embedding fields: A theory of learning with physiological implications,” *Journal of Mathematical Psychology*, vol. 6, no. 2, pp. 209–239, 1969.

[43] A. H. Klopf and E. Gose, “An evolutionary pattern recognition network,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 3, pp. 247–250, 1969.

[44] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities.” *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation, parallel distributed processing, explorations in the microstructure

of cognition, ed. de rumelhart and j. mcclelland. vol. 1. 1986," *Biometrika*, vol. 71, no. 599-607, p. 6, 1986.

[46] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 22, no. 5, pp. 717–727, 2000.

[47] A. I. Lawal and M. A. Idris, "An artificial neural network-based mathematical model for the prediction of blast-induced ground vibrations," *International Journal of Environmental Studies*, vol. 77, no. 2, pp. 318–334, 2020.

[48] K. Suzuki, *Artificial Neural Networks: Architectures and applications*. BoD–Books on Demand, 2013.

[49] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *Ieee Potentials*, vol. 13, no. 4, pp. 27–31, 1994.

[50] M. Braun and M. Golubitsky, *Differential Equations and their Applications*. Springer, 1983, vol. 2.

[51] L. L. Littlejohn, "On the classification of differential equations having orthogonal polynomial solutions," *Annali di matematica pura ed applicata*, vol. 138, no. 1, pp. 35–53, 1984.

[52] D. Zwillinger and V. Dobrushkin, *Handbook of Differential Equations*. Chapman and Hall/CRC, 2021.

[53] V. Akinsola, "Numerical methods: Euler and runge-kutta," in *Qualitative and Computational Aspects of Dynamical Systems*. IntechOpen, 2023.

[54] B. Biswas, S. Chatterjee, S. Mukherjee, and S. Pal, "A discussion on euler method: A review," *Electronic Journal of Mathematical Analysis and Applications*, vol. 1, no. 2, pp. 2090–2792, 2013.

[55] B. Denis, "An overview of numerical and analytical methods for solving ordinary differential equations," *arXiv preprint arXiv:2012.07558*, 2020.

[56] K. G. Tay, T. H. Cheong, M. F. Lee, S. L. Kek, and R. Abdul-Kahar, "A fourth-order runge-kutta (rk4) spreadsheet calculator for solving a system of two first-order ordinary differential equations using visual basic (vba) programming," *Spreadsheets in Education*, vol. 8, no. 1, 2014.

[57] M. N. Aslam, M. W. Aslam, M. S. Arshad, Z. Afzal, M. K. Hassani, A. M. Zidan, and A. Akgül, "Neuro-computing solution for lorenz differential equations through artificial neural networks integrated with pso-nna hybrid meta-heuristic algorithms: A comparative study," *Scientific Reports*, vol. 14, no. 1, p. 7518, 2024.

[58] L. Yang, D. Zhang, and G. E. Karniadakis, "Physics-informed generative adversarial networks for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 42, no. 1, pp. A292–A317, 2020.

[59] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. ieee, 1995, pp. 1942–1948.

[60] Y. Del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Transactions on Evolutionary computation*, vol. 12, no. 2, pp. 171–195, 2008.

[61] M. Pakdaman, A. Ahmadian, S. Effati, S. Salahshour, and D. Baleanu, "Solving differential equations of fractional order using an optimization technique based on training artificial neural network," *Applied Mathematics and Computation*, vol. 293, pp. 81–95, 2017.

[62] H. Modares and M.-B. N. Sistani, "Solving nonlinear optimal control problems using a hybrid ipso–sqp algorithm," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 3, pp. 476–484, 2011.

[63] M. A. Z. Raja, M. A. Manzar, and R. Samar, "An efficient computational intelligence approach for solving fractional order riccati equations using ann and sqp," *Applied Mathematical Modelling*, vol. 39, no. 10-11, pp. 3075–3093, 2015.

[64] V. Akinsola, "Numerical methods: Euler and runge-kutta," in *Qualitative and Computational Aspects of Dynamical Systems*. IntechOpen, 2023.

[65] S. A. Bhat, Z. Sabir, M. A. Z. Raja, T. Saeed, and A. M. Alshehri, "A novel heuristic morlet wavelet neural network procedure to solve the delay differential perturbed singular model," *Knowledge-Based Systems*, vol. 292, p. 111624, 2024.

[66] S. Sivanandam, S. Deepa, S. Sivanandam, and S. Deepa, *Genetic Algorithms*. Springer, 2008.

[67] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. ieee, 1995, pp. 1942–1948.

[68] M. N. Aslam, M. W. Aslam, M. S. Arshad, Z. Afzal, M. K. Hassani, A. M. Zidan, and A. Akgül, “Neuro-computing solution for lorenz differential equations through artificial neural networks integrated with pso-nna hybrid meta-heuristic algorithms: A comparative study,” *Scientific Reports*, vol. 14, no. 1, p. 7518, 2024.