**PREDICTION OF CHAOTIC DYNAMICS IN HYPERMULTISTABILITY MEMRISTIVE SYSTEM USING RESERVOIR COMPUTING**

**BY**

**ADETAYO MATHEW TEMITOPE**

**MATRIC NO: 190808064**

**A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF PHYSICS, FACULTY OF PHYSICAL AND EARTH SCIENCE, UNIVERSITY OF LAGOS, NIGERIA. IN PARTIAL FULFILMENT OF REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE (B.SC.) DEGREE IN PHYSICS.**

**OCTOER, 2025**

## **CERTIFICATION**

This is to certify that ADETAYO MATHEW (190808064) provided this project report. It was carried out in the Department of Physics, University of Lagos, Akoka, Yaba, Lagos,

Nigeria in the partial fulfillment of the requirement for the award of a Bachelor of Science (B.Sc) Degree in Physics. We hereby certify that work taken to the best of our knowledge has not been submitted in the part or in full for the award of a B.Sc. or any other degree in this or any other University.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DR. K. S. OJO Date

(Project Supervisor)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DR. N.I. OBOT

(Departmental Project Coordinator)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PROF. O.I. OLUSOLA

(Head, Department of Physics)

## **DEDICATION**

With heartfelt affection, I dedicate this endeavor to my dear Uncle, MR HAKEEM KAREEM, and Aunt, MRS WASILAT ADEBOLA.

## **ACKNOWLEDGEMENTS**

I express my deepest appreciation to the Divine Creator, for His boundless mercy, grace, and guiding presence that illuminated my path during this research endeavor.

I would like to extend my sincere appreciation to my supervisor, DR. K. S. OJO, whose unwavering dedication, invaluable advice, exceptional patience, and unwavering guidance played a pivotal role in the successful completion of this project.

I extend my gratitude to the department staff for their cordial support and encouragement.

I am profoundly thankful to my family members, superiors, and friends. Your unwavering belief in me has been a tremendous source of inspiration, propelling me toward greater opportunities. I extend my heartfelt gratitude to each of you.

Proper recognition is granted to all the information acquired from external sources during the composition of this project.

## **ABSTRACT**

Chaotic systems, though deterministic, are susceptible to initial conditions, making their long-term prediction extremely challenging. Many systems exhibit chaos; however, they have remained difficult to model up till now. Memristive chaotic systems, which integrate nonlinear dynamics with memory effects, exhibit complex behaviors such as hyperchaos and multistability, offering potential applications in secure communication, neuromorphic computing, and random number generation. In this work, we investigate a four-dimensional memristive chaotic system derived from the boostable VB2 model with tangent nonlinearities and a flux-controlled memristor. Numerical analyses, including Lyapunov exponents and bifurcation diagrams, confirm the presence of chaos and multiple coexisting attractors. One thing Reservoir Computing (RC) can do is give us a new ability to understand and model such systems. While RC has been widely applied to classical chaotic systems, its application to continuous memristive chaos remains unexplored. Results show that RC accurately predicts short-term trajectories while preserving key dynamical features.

**Table of Contents**

[**CERTIFICATION** ii](#_Toc210612311)

[**DEDICATION** iii](#_Toc210612312)

[**ACKNOWLEDGEMENTS** iv](#_Toc210612313)

[**ABSTRACT** v](#_Toc210612314)

[**CHAPTER ONE** 1](#_Toc210612315)

[**INTRODUCTION** 1](#_Toc210612316)

[1.2 AIM AND OBJECTIVES 2](#_Toc210612317)

[1.3 DEFINITION OF TERMS 3](#_Toc210612318)

[1.3.2 Chaotic system 3](#_Toc210612319)

[1.3.3 Memristive system 5](#_Toc210612320)

[1.3.4 Artificial nueral networks 6](#_Toc210612321)

[1.3.5 Reservoir computing 6](#_Toc210612322)

[1.3.6 Lyapunov exponent 7](#_Toc210612323)

[1.3.6 Bifurcation analysis 8](#_Toc210612324)

[2.3 Numerical method for solving Memristive differential Equations 11](#_Toc210612325)

[2.4 Description of Artificial Neural Networks (ANN) 11](#_Toc210612326)

[2.5 Types of Artificial Neural Networks 12](#_Toc210612327)

[2.5.1 Feedforward Neural Networks (FNNs) 12](#_Toc210612328)

[2.5.2 Recurrent Neural Networks (RNNs) 12](#_Toc210612329)

[2.6 Activation Functions 13](#_Toc210612330)

[2.6.1 Types of Activation Functions 13](#_Toc210612331)

[2.6.1.1 Sigmoid Activation Function 13](#_Toc210612332)

[2.6.1.2 Tanh Activation Function 14](#_Toc210612333)

[2.6.1.3 ReLU Activation Function 14](#_Toc210612334)

[CHAPTER THREE 16](#_Toc210612335)

[MODEL DESCRIPTION AND METHOD 16](#_Toc210612336)

[3.1 BRIEF SYSTEM DESCRIPTION 16](#_Toc210612337)

[3.1.1 Description of the Memristive System 16](#_Toc210612338)

[3.1.2 Mathematical Formulation 18](#_Toc210612339)

[3.1.3 System Parameters and Initial Conditions 19](#_Toc210612340)

[3.2 DATA GENERATION 20](#_Toc210612341)

[3.21 Time Step and Simulation Duration 21](#_Toc210612342)

[3.22 Numerical Integration via Runge–Kutta Method 22](#_Toc210612343)

[**3.3 MODEL DESCRIPTION** 25](#_Toc210612344)

[3.3.1.1 Input-To-Node 26](#_Toc210612345)

[3.3.1.2 Node to Node Transformation. 27](#_Toc210612346)

[3.3.1.3 Node to output Transformation. 28](#_Toc210612347)

[3.3.3 Converting Data to Format for Prediction 29](#_Toc210612348)

[3.5 Lyapunov Exponent Calculation: 29](#_Toc210612349)

[3.4 Building models for reservoir computing 30](#_Toc210612350)

[3.4.1 ESN model architecture 30](#_Toc210612351)

[3.4.2 parameter Optimization for reservoir computing: 31](#_Toc210612352)

[CHAPTER FOUR 32](#_Toc210612353)

[RESULTS AND DISCUSSIONS 32](#_Toc210612354)

[4.1 MEMRISTIVE SYSTEM PREDICTIONS 32](#_Toc210612355)

[**4.2** Lyapunov exponents result 35](#_Toc210612356)

[4.3 DISCUSSIONS 36](#_Toc210612357)

[CHAPTER FIVE 39](#_Toc210612358)

[CONCLUSION 39](#_Toc210612359)

[NOTATIONS 39](#_Toc210612360)

[REFERENCES 40](#_Toc210612361)

## **CHAPTER ONE**

## **INTRODUCTION**

**1.1 BACKGROUND OF STUDY**

In 1971, **Leon Chua** predicted the existence of the **memristor** (Chua, 1971), short for "memory resistor." This unique component, posited as the fourth passive electronic element alongside the resistor, capacitor, and inductor, regulates electrical current based on its history, retaining a *memory* of past charge flow. Fast-forward to 2008: engineers at **HP Labs** detected an unusual signal. After a great deal of investigation and debate, they determined that this signal was evidence of the memristor Chua had theorized (Williams et al., 2008). The relationship of this fourth fundamental circuit element can be derived from the four basic parameters of charge (**q**), current (**i**), voltage (**v**), and magnetic flux (**φ**) (Chua & Kang, 1976; Di Ventra et al., 2009; Pershin & Di Ventra, 2011; Strukov et al., 2008).However, the concept has not been widely accepted due to the lack of widespread experimental confirmation and the rapid advancement of silicon-based digital circuits. Moreover, traditional circuit technology, based on the von Neumann architecture, faces inevitable limitations such as low computing speed and high-power consumption (Backus, 1978; Mead, 1990; Indiveri & Liu, 2015).

As a result, researchers have shifted focus toward the development of next-generation electronic devices. Fortunately, in 2008, HP Labs announced the physical realization of a memristor based on titanium dioxide materials [8]. The memristor’s properties—including non-volatility, adjustable conductivity, high-density integration, low power consumption, and fast operation speed—have positioned it as a promising component in applications such as non-volatile memory, artificial synapses, and neuromorphic computing (Yang et al., 2013; Jo et al., 2010; Kuzum et al., 2012; Zidan et al., 2018).

There are approximately one hundred billion neurons in the human brain, interconnected via synapses to form a complex, highly efficient, ultra-low-power neural network. This biological network supports learning, memory, and adaptive decision-making—capabilities modern computers are only beginning to emulate. The long-term vision for computer development is to create **brain-inspired systems** capable of flexible, autonomous decision-making and efficient information processing. (Mead, 1990; Indiveri & Liu, 2015).

Reservoir Computing (RC) has emerged as a powerful computational framework inspired by the dynamics of recurrent neural networks (RNNs). RC is particularly well-suited for processing time series signals and modeling nonlinear dynamical systems. A significant advantage of RC lies in its simplicity: only the readout layer is trained, while the internal "reservoir" remains fixed. This drastically reduces training time and computational complexity. One of the most promising applications of RC is in the modeling and prediction of memristive systems—nonlinear devices whose resistance depends on past input history. These systems are crucial components of neuromorphic computing, but their nonlinearity and memory-dependence make them difficult to model using conventional tools. Compared to other artificial intelligence models, such as traditional artificial neural networks (ANNs)

These qualities make Reservoir Computing the ideal type of RNN’s for the prediction of chaotic dynamics in complex systems such as multistable memristive systems.

## 

## **1.2 AIM AND OBJECTIVES**

This project aims to carry out modelling and predictions of a chaotic memristive system using reservoir computing

**OBJECTIVES**

* + 1. To confirm the chaotic dynamic memristive system.
    2. To confirm the multistable nature of the memristive system.
    3. To use machine learning (reservoir computing) to model and predict the system’s behavior over time.
    4. To identify the reservoir parameters that affect the accuracy of the prediction.

## **1.3 DEFINITION OF TERMS**

**1.3.1 Dynamic system**

A dynamical system is a mathematical model describing the evolution of a state over time according to a fixed rule. Formally, it consists of a set (the state space) and a law that determines how points in this set evolve with time. In discrete dynamical systems, the state evolves in steps according to an iteration rule, while in continuous dynamical systems, the state changes smoothly following differential equations (Fernandes, Ramos, Thapa, Lopes, & Grácio, 2018). The dynamical system gives a unified framework for the study of any system that evolves over time and by analyzing a dynamical system, we scientist are pegged with some questions to answer- does it settle into a stable state? does it oscillate and is the behavioral of the system chaotic?

## 1.3.2 Chaotic system

Chaos theory is an interdisciplinary branch of science and mathematics that studies complex systems governed by deterministic laws but behave in unpredictable ways due to extreme sensitivity to initial conditions. These systems, which once appeared to be completely random or disordered, contain underlying patterns, feedback loops, self-organization, and fractal structures.

One of the key ideas in chaos theory is the butterfly effect that illustrates how a tiny change in the starting conditions of a system can led to significantly different outcomes over time. A common metaphor for this is: *a butterfly flapping its wings in Brazil might cause or prevent a tornado in Texas (Lorentz , 1972)*.

This sensitivity makes chaotic systems difficult to predict long-term, even though they are deterministic in nature (not random). Examples of such systems can be found in weather patterns, electrical circuits, biological systems, and even financial markets.

The fact that some dynamical model systems showing the above necessary conditions possess such a critical dependence on the initial conditions has been known since the end of the last century. However, only in the last thirty years, experimental observations have pointed out that, in fact, chaotic systems are common in nature. They can be found, for example, in Nonlinear Optics (lasers) (Haken ,1985), in Electronics (Chua& Matsumoto circuit) , in Fluid Dynamics Rayleigh–Benard convection (Swiney &Gollub, 1985), etc. as being chaotic. They can be found in meteorology, the solar system, the heart and brain of living organisms, and so on.

Due to their critical dependence on the initial conditions, and due to the fact, that, in general, experimental initial conditions are never known perfectly, these systems are intrinsically unpredictable. Indeed, the prediction trajectory emerging from a Bonafide initial condition and the real trajectory emerging from the real initial condition diverges exponentially in course of time, so that the error in the prediction (the distance between prediction and real trajectories) grows exponentially in time, until making the system’s real trajectory is completely different from the predicted one at long times.

For many years, this feature made chaos undesirable, and most experimentalists considered such characteristics as something to be strongly avoided. Besides their critical sensitivity to initial conditions, chaotic systems exhibit two other important properties. Firstly, there is an infinite number of unstable periodic orbits embedded in the underlying chaotic set. In other words, the skeleton of a chaotic attractor is a collection of an infinite number of periodic orbits, each one being unstable. Secondly, the dynamics in the chaotic attractor are ergodic, which implies that during its temporal evolution the system ergodically visits small neighborhoods of every point in each one of the unstable periodic orbits embedded within the chaotic attractor.

A relevant consequence of these properties is that chaotic dynamics can be seen as shadows of some periodic behavior at a given time and erratically jumping from one to another periodic orbit. The idea of controlling chaos is then when a trajectory approaches ergodically a desired periodic orbit embedded in the attractor; one applies small perturbations to stabilize such an orbit. If one switches on the stabilizing perturbations, the trajectory moves to the neighborhood of the desired periodic orbit that can now be stabilized. This fact has suggested the idea that the critical sensitivity of a chaotic system to changes (perturbations) in its initial conditions may be, in fact, very desirable in practical experimental situations. Indeed, if it is true that a small perturbation can give rise to a very large response in the course of time, it is also true that a judicious choice of such a perturbation can directs the trajectory to wherever one wants in the attractor and produce a series of desired dynamical states. This is exactly the idea of targeting.

The important point here is that, because of chaos, one can produce an infinite number of desired dynamic behaviors (either periodic or not periodic) using the same chaotic system, with only the help of tiny perturbations chosen properly. We stress that this is not the case for nonchaotic dynamics, wherein the perturbations to be done for producing the desired behavior must, in general, be of the same order of magnitude as the unperturbed evolution of the dynamic variables.

## **1.3.3 Memristive system**

A memristive system is a class of nonlinear dynamical systems whose resistance depends on the history of the current or voltage applied to them. The concept was first introduced by chua 1971, who introduced the memristor as the fourth fundamental circuit element.

In recent years, memristor has been introduced into chaotic systems widely, which are used in neural networks, communication systems, computer engineering, chaotic signal control and image encryption. Memristors are semiconductors that join a capacitor, resistor, and inductor to make a fourth new kind of element whose resistance is called as memristance that varies as a function of current and flux. As per the theory, Memristors, a combination of “memory resistors”, is a kind of passive circuit element that maintains a relationship between the time integrals of current and voltage across a two-terminal element. Hence, the resistance of a memristor varies according to access to a “history” of the applied voltage. When the current flows in one direction the resistance increases, in contrast when the current flows in the opposite direction the resistance decreases. However, resistance cannot go below zero. When the current is stopped the resistance remains at the value that it had previously. It means MEMRISTOR “REMEMBERS “the current that had last flowed across it.

Memristors, which are a subdivision of a group of resistive RAMS, are one of several storage technologies that have been predicted to replace flash memory.

## **1.3.4 Artificial nueral networks**

Artificial Neural Networks (ANNs) are a class of machine learning models inspired by the structure and functioning of biological neural networks. They consist of interconnected nodes, called artificial neurons, that are organized into layers and connected through weighted links. These networks are capable of learning complex, nonlinear mappings between inputs and outputs through iterative training processes (McCulloch & Pitts, 1943; Rosenblatt, 1958; Rumelhart et al., 1986). The basic computational unit of an ANN is the artificial neuron, which receives weighted inputs, applies a nonlinear activation function, and produces an output. By combining large numbers of such units, ANNs can approximate arbitrary continuous functions, a property formalized in the universal approximation theorem (Hornik et al., 1989).

## **1.3.5 Reservoir computing**

Reservoir Computing is a framework for training Recurrent Neural Networks (RNNs). It involves two main components: a fixed, large, random recurrent neural network, known as the "reservoir", and a trainable output layer. The reservoir is used to transform the input data into a higher-dimensional space, while the output layer is trained to read the activity from the reservoir. Most introductions to Reservoir computing begin with echo state networks (ESNs) (Jaegar,2001), and liquid state machines (LSMs) (Maass et al., 2002), but the ideas from RC can be traced back further into the past . The principle behind RC was present already in the context of reverberation subsystems by Kirby 1991, who even pointed out the possibility of hardware implementations for AI applications. Schomaker described a reservoir-like system with spiking neural oscillators. In cognitive neurosciences, Dominey 1995 used a reservoir as a simplified model of the prefrontal cortex, which has led to fruitful research that continues to the present (Dominey, 1997).The idea behind RC has been (re)discovered multiple times. This may be on account of the conceptual simplicity of computing functions through a high-dimensional temporal expansion of the input. The term RC can be traced back to Verstraeten et al. (2007), in which the authors experimentally unified ESNs and LSMs into a general framework of computation using a reservoir.

## **1.3.6 Lyapunov exponent**

Lyapunov exponent is a concept in the field of dynamical systems and chaos theory that measures the rate of exponential divergence or convergence of nearby trajectories in a system. It provides valuable insights into the system’s sensitivity to initial conditions and its long-term predictability. The Lyapunov exponent characterizes the local stability properties of a dynamical system by quantifying how infinitesimally close trajectories evolve over time. It is defined for each direction in the system’s phase space and can be computed for discrete-time or continuous time systems.

A positive Lyapunov exponent indicates that nearby trajectories diverge exponentially, implying sensitive dependence on initial conditions and inherent chaotic behavior. In contrast, a negative exponent indicates that trajectories converge towards each other, indicating stability and regular behavior. A zero Lyapunov exponent indicates neutral stability, with trajectories neither significantly diverging nor converging. The largest Lyapunov exponent, often referred to as the dominant Lyapunov exponent, is of particular interest as it characterizes the overall system’s behavior. If this exponent is positive, the system is deemed chaotic. However, the presence of positive Lyapunov exponents does not guarantee chaos; other criteria, such as a strange attractor or sensitivity to system parameters, must also be considered.

Lyapunov exponents have diverse applications across various scientific fields. In physics, they help understand the behavior of complex systems such as fluid dynamics, celestial mechanics, and quantum chaos. In biology, they are employed to study population dynamics and neural networks. Furthermore, Lyapunov exponents play a crucial role in data analysis, synchronization, cryptography, and designing secure communication systems. Overall, the Lyapunov exponent provides a quantitative measure of the system’s sensitivity to initial conditions, offering valuable insights into the dynamics and predictability of complex systems, particularly those exhibiting chaotic behavior.

## **1.3.6 Bifurcation analysis**

Continuous dynamical systems that involve differential equations mostly contain parameters. It can happen that a slight variation in a parameter can have significant impact on the solution. In dynamical systems, a bifurcation occurs when a small smooth change made to the parameter values (the bifurcation parameters) of a system causes a sudden” qualitative" or topological change in its behavior. Generally, at a bifurcation, the local stability properties of equilibria, periodic orbits or other invariant sets change. It has two types.

Local bifurcations, which can be analyzed entirely through changes in the local stability properties of equilibria, periodic orbits or other invariant sets as parameters cross through critical thresholds; and Global bifurcations, which often occur when larger invariant sets of the system” collide” with each other, or with equilibria of the system. They cannot be detected purely by a stability analysis of the equilibria.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 Overview of Chaos and the Memristive system**

Chaos theory is a branch of nonlinear dynamics that specializes in deterministic systems exhibiting unpredictable and highly sensitive behavior. In chaotic systems, small changes in initial conditions can lead to vastly different outcomes—a property known as sensitivedependence **on** initialconditions. Despite their apparent randomness, chaotic systems are governed by deterministic rules and can often be described by a set of nonlinear differential equations. Common examples include weather systems, electrical circuits, population dynamics, and chemical reactions.

In recent decades, **chaotic dynamics** have been harnessed for practical applications such as secure communications, random number generation, and fault detection. Their complexity, however, poses significant challenges in modeling and long-term prediction, especially when extended to systems with hypermultistability—a condition where multiple attractors coexist within a single dynamical framework.

One of the most promising platforms for exploring chaos in physical systems is the **memristive system**. First postulated by Leon Chua in 1971, the memristor (short for memory resistor) was recognized as the fourth fundamental passive circuit element, alongside the resistor, capacitor, and inductor. Its key feature is its ability to retain a memory of the charge that has previously flowed through it, making its resistance a function of historical current. This memory property introduces a natural form of nonlinearity and hysteresis into circuits, which is ideal for inducing and sustaining chaotic behavior.

reservoir computing (RC) emerges as a powerful machine learning paradigm for modeling and predicting chaotic memristive systems. RC leverages the high-dimensional state space of a recurrent dynamical system (the “reservoir”) to map complex temporal patterns into a linearly separable form (Jaeger, 2001; Maass, Natschläger, & Markram, 2002). Given the intrinsic nonlinearity and memory properties of memristors, combining them with RC provides a promising framework for both understanding and predicting chaotic dynamics with high accuracy.

**2.2 A Brief Review of the memristive chaotic system**

Memristive chaotic systems extend traditional chaotic oscillators by incorporating **flux- or charge-controlled memristors** into their dynamics. The inclusion of a memristor introduces memory-dependent nonlinearity, enabling richer and more controllable chaotic behaviors compared to classical circuits. In this paper we will be modeling and predicting a new memristive chaotic system derived from the boostable VB2 system (Jafari et al., 2013). The original VB2 model contained one quadratic and four linear terms; however, by introducing tangent functions and a flux-controlled memristor, a more complex chaotic oscillator is obtained. This structure highlights the intrinsic coupling between the system’s dynamical variables and the memristor’s memory property. The additional state variable uuu introduced by the memristor increases the dimensionality of the system, thereby allowing hyperchaotic attractors and multistability phenomena to emerge (Sun et al., 2019; Bao et al., 2019).

For this project, we will be particularly interested in leveraging **Reservoir Computing (RC)** to model and predict the dynamics of this memristive chaotic system. RC is well suited to such tasks because it naturally captures temporal dependencies and nonlinear interactions, both of which are central to memristive chaos. By applying RC to this system, we aim not only to predict its future state but also to evaluate how well machine learning frameworks can characterize the interplay between chaos and memory effects. While RC has been widely applied to predict the trajectories of benchmark chaotic systems such as the Lorenz and Mackey–Glass systems (Jaeger & Haas, 2004; Pathak et al., 2018), and even to some discrete memristive hyperchaotic maps (Zhang et al., 2023), to the best of our knowledge no prior work has employed RC to predict the behavior of continuous memristive chaotic systems of the type studied here. This research therefore addresses a critical gap, combining the intrinsic memory properties of memristors with the computational power of RC to advance the state of chaotic system prediction

## **2.3 Numerical method for solving Memristive differential Equations**

Memristive chaotic systems, such as the one considered in this paper, are described by systems of coupled nonlinear differential equations. In most cases, these equations will be difficult and tedious and practically impossible to solve analytically due to their nonlinearity and complexity. Therefore, numerical methods are employed to approximate their trajectories and explore their dynamical behaviors. The most widely used approaches for solving such systems are based on **time discretization methods**, where the continuous-time equations are transformed into iterative update rules. Among these methods, the Runge–Kutta family of algorithms is particularly popular due to its balance between computational efficiency and accuracy. The classical fourth-order Runge–Kutta **(**RK4) method is often the method of choice in chaos and nonlinear dynamics research (Butcher, 2008; Press et al., 2007).

In alignment with these best practices, our study employs the RK4 method to integrate the nonlinear system equations derived from the VB2-boosted chaotic framework with a memristor. The state variables along with the memductance are updated at each iteration to form the data set used in our reservoir computing model. This approach ensures numerical stability and fidelity in capturing the fine structures of chaotic attractors. For this paper, we will employ the fourth-order Runge–Kutta method to numerically solve the proposed memristive chaotic system. This approach will allow us to generate reliable time-series data of the system’s state variables, which will later serve as input to the Reservoir Computing framework for modeling and prediction tasks.

## **2.4 Description of Artificial Neural Networks (ANN)**

Artificial neural networks (ANNs), usually simply called **neural networks** (**NNs**) or **neural nets**, are computing systems inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives signals then processes them and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called *edges*. Neurons and edges typically have a *weight* that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations in their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

## **2.5 Types of Artificial Neural Networks**

Several variants of neural network architectures have been developed to address specific types of problems. Below are some of the most relevant to time-series prediction and system modeling.

## **2.5.1 Feedforward Neural Networks (FNNs)**

Feedforward Neural Networks are the simplest form of ANNs. In FNNs, the flow of data is unidirectional—from the input layer to the output layer, without any cycles or loops. Each layer consists of multiple neurons, and each neuron in one layer is connected to every neuron in the next. FNNs are effective in problems like classification, regression, and function approximation, but they struggle to model temporal dependencies in data, which limits their use in forecasting chaotic systems or time-dependent signals.

**2.5.2 Recurrent Neural Networks (RNNs)**

Recurrent Neural Networks are specifically designed for handling sequential or time-dependentdata. Unlike feedforward networks, RNNs have loops that allow information to persist overtime, enabling the network to learn temporal patterns and dependencies. In RNNs, the output from the previous time step is fed back into the network as input for the current step, making them suitable for tasks like time-series prediction, speech recognition, and chaotic system modeling. However, basic RNNs suffer from issues such as vanishing and exploding gradients, which hinder their ability to learn long-term dependencies.

To address this, advanced RNN variants like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) have been introduced. These models are commonly used in forecasting the evolution of chaotic systems, including memristive dynamics.

## **2.6 Activation Functions**

Activation functions are mathematical tools employed in neural networks to introduce nonlinearity into the model, allowing the system to learn and approximate complex and non-linear mappings between inputs and outputs. Activation functions determine the non-linear transformation of signals within the reservoir layer, significantly influencing the internal dynamics, memory capacity, and representational power of the network.

In forecasting chaotic systems—such as memristive circuits that exhibit sensitivity to initial conditions and non-trivial attractor structures—activation functions must be selected with care to preserve the temporal structure and nonlinear evolution of the input signals. This section discusses the most used activation functions in time-series prediction tasks, particularly within RC architecture.

## **2.6.1 Types of Activation Functions**

## **2.6.1.1 Sigmoid Activation Function**

The sigmoid function is defined mathematically as:

and it compresses input values into the interval ( Historically, it was widely used in early neural networks due to its smooth and differentiable nature.

However, its use in Reservoir Computing, particularly for modeling chaotic dynamics, is limited. The sigmoid function tends to saturate for large absolute input values, resulting in small gradients and hence reduced sensitivity to input variations. This characteristic leads to a **loss of dynamic richness** in the reservoir and restricts the system’s ability to accurately capture the fast-changing states in chaotic time series.

Furthermore, the non-zero-centered output of the sigmoid function can cause gradient-related issues during training and reduce convergence speed in certain network architectures.

## **2.6.1.2 Tanh Activation Function**

The hyperbolic tangent (tanh) function is defined as:

mapping input values into the range Compared to the sigmoid function, tanh is zero-centered, which facilitates faster convergence during training and more balanced signal propagation in recurrent architectures.

In this study, the tanh activation function was adopted for the reservoir nodes due to its superior ability to capture oscillatory and bidirectional dynamics that are characteristic of chaotic systems. The function's output range allows the reservoir to effectively represent both positive and negative trajectories in the phase space of the memristive system. It also helps maintain the **echo state property**, which is essential for stable and responsive reservoir dynamics.

## **2.6.1.3 ReLU Activation Function**

The Rectified Linear Unit (ReLU) is defined as:

and is known for its computational efficiency and simplicity. ReLU is widely used in deep learning due to its ability to mitigate the vanishing gradient problem and introduce sparsity in activations. Despite these advantages, ReLU is less suitable for chaotic time-series forecasting in Reservoir Computing. Its unbounded and asymmetric output can lead to imbalances in the internal state space, and it lacks the smooth gradient transition required to model the **fine-grained nonlinearities** of chaotic dynamics. Moreover, ReLU can cause some reservoir neurons to become permanently inactive (the "dying ReLU" problem), thereby reducing the reservoir’s capacity to model long-term temporal dependencies.

Experimental evaluations conducted during this research confirmed that ReLU underperformed relative to tanh in terms of both stability and prediction accuracy when applied to the chaotic memristive system.

**Conclusion**

In summary, while various activation functions offer distinct advantages depending on the application, the **tanh function** was found to be the most appropriate for this project’s objectives. Its symmetrical output and nonlinear characteristics align well with the dynamic behavior of chaotic memristive systems, enabling the reservoir to effectively learn and forecast system evolution. The selection of tanh contributed significantly to maintaining the stability of the reservoir and enhancing the predictive accuracy of the Echo State Network architecture used in this study.

## **CHAPTER THREE**

## **MODEL DESCRIPTION AND METHOD**

## **3.1 BRIEF SYSTEM DESCRIPTION**

## **3.1.1 Description of the Memristive System**

In this paper, we analyze a four-dimensional chaotic system that combines the features of the VB2 boostable chaotic system with a **flux-controlled memristor**. The memristor introduces memory into the system by making its behavior dependent on the past values of one of the variables, specifically through the time-integrated value of the system state which we call magnetic **flux**.

Unlike traditional chaotic systems, this memristive system contains **nonlinear trigonometric functions** (such as tangent and arctangent) and a state-dependent feedback loop influenced by the memristor. This combination results in rich chaotic behavior, including multiple attractors and sensitive dependence on initial conditions.

The state of the system evolves in time based on four variables:

* which represents the main system states,
* which represents the flux (the accumulated value of overtime).

The memristive behavior is captured using a memductance function which depends on the flux and modulates how strongly influences the system. The governing equations are given as:

3.1

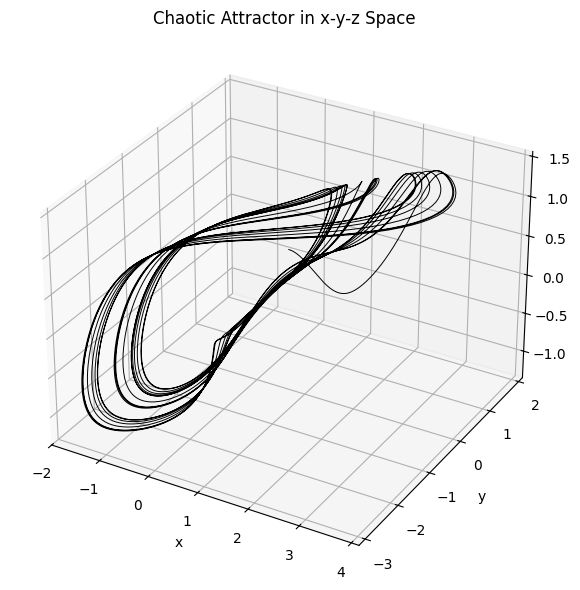


Fig 1 showing the chaotic attractor

**A screenshot of a graph

AI-generated content may be incorrect.**

Fig 2 Time Series Of variables

## **3.1.2 Mathematical Formulation**

**Description of Terms:**

* Dynamical state variables representing the system’s evolving states.
* : Internal state variable of the memristor, associated with flux.
* : Controls the nonlinear contribution of the tangent function.
* Parameters defining the linear flux dependence of the memductance.
* Flux-controlled memductance of the memristor.
* : Current through the memristor.

The system defined in Eq. (3.1) is a **four-dimensional nonlinear system**. The inclusion of the memristor introduces a memory term (u) that directly influences the system dynamics through W(u). This additional degree of freedom allows for the emergence of **complex chaotic and hyperchaotic behaviors**. In practice, solving Eq. (3.1) requires numerical integration techniques, since no closed-form analytical solution exists due to the nonlinear terms. In this project, the fourth-order Runge–Kutta method will be employed to obtain the system trajectories under different initial conditions and parameter values. The time-series generated data will subsequently serve as input for the Reservoir Computing framework, enabling modeling and prediction of the memristive chaotic dynamics.

where:

* represent the time derivatives of the variables,
* is the memductance (i.e., conductance of the memristor), defined as:

and:

* = is the flux linking the present with past system states.

This system is nonlinear and exhibits chaotic behavior due to the combination of the trigonometric terms and the memory feedback introduced by the memristor via

## **3.1.3 System Parameters and Initial Conditions**

To simulate and analyze the system numerically, we selected the following parameters based on existing literature and numerical experimentation:

which define the scaling behavior of the memductance function

The initial conditions are chosen to observe the system’s response and its sensitivity to starting values:

These values serve as the starting point for numerical simulation using a suitable method such as the **4th-order Runge-Kutta** (RK4), which is discussed in Section 3.2. The choice of parameters ensures that the system exhibits chaotic behavior with multiple attractors in different regions of the phase space.

## **3.2 DATA GENERATION**

Numerical solutions of ordinary differential equations are the most important technique in continuous time dynamics. Since most ordinary differential equations are not soluble analytically, numerical integration is the only way to obtain information about the trajectory. Many different methods have been proposed and used to solve accurately various types of ordinary differential equations. However, there are a handful of methods known and used universally (i.e., Runge Kutta, Adams–Bashforth–Moulton and Backward Differentiation Formulae methods). All these discretize the differential system to produce a different equation or map. The methods obtain different maps from the same differential equation, but they have the same aim; that the dynamics of the map should correspond closely to the dynamics of the differential equation. From the Runge–Kutta family of algorithms come arguably the most well-known and used methods for numerical integration (see, for example, Henrici [1962], Gear [1971], Lambert [1973], Stetter [1973], Chua & Lin [1975], Hall & Watt [1976], Butcher [1987], Press et al. [1988], Parker & Chua [1989], or Lambert [1991]). Thus, we choose to look at Runge–Kutta methods to investigate what pitfalls there may be in the integration of nonlinear and chaotic systems**.**

We employed the classical 4th-order Runge-Kutta (RK4) method due to its balance between computational efficiency and numerical accuracy in solving ordinary differential equations (ODEs). The system equations described earlier in Section 3.1.1 are integrated with a fixed step size over a total time duration units. This produces a sufficiently large dataset to capture the full range of dynamical states, including bifurcations, attractor transitions, and chaotic trajectories.

Initial conditions for the state variables are selected based on regions known to exhibit hyperchaotic and multistable behavior. A grid of initial values was also tested to analyze sensitivity and the basin of attraction. This systematic approach enables the generation of multiple trajectories that demonstrate coexisting attractors — a hallmark of hypermultistability.

The output data includes time-series for all four state variables sampled uniformly over the simulation window. These series are stored in matrices and exported as CSV files for further processing. Each row in the dataset corresponds to a time step, and each column corresponds to a system variable.

To verify the chaotic nature of the data generated, we computed the largest Lyapunov exponent using the Rosenstein algorithm. Positive Lyapunov exponents confirm sensitive dependence on initial conditions, validating that the system exhibits chaos.

This dataset forms the basis for the training and validation of the reservoir computing (RC) model, as it captures both short-term nonlinear interactions and long-term dynamical structure of the memristive system.

## **3.2.1 Time Step and Simulation Duration**

The choice of simulation duration and time step is essential in capturing the intricate dynamics of chaotic systems. A time step that is too large may skip over important nonlinear transitions, while one that is too small may increase computational cost without significant gain in accuracy.

For this simulation, the following parameters are selected:

* **Total simulation time**:
* **Time step size**:
* **Number of steps**:

This fixed time step provides sufficient resolution to capture fast transitions in the system caused by the tangent and memristive terms, while also ensuring numerical stability over long simulations. The resulting time series from the RK4 integration is used to generate 3D phase portraits and to prepare datasets for data-driven modeling via Reservoir Computing.

Multiple simulations are performed under varying initial conditions to explore the presence of coexisting attractors, which are critical for demonstrating the system’s hypermultistability nature

## **3.2.2 Numerical Integration via Runge–Kutta Method**

To simulate the dynamic behavior of the hypermultistable memristive system and generate training data for reservoir computing, we employ the **Runge–Kutta (RK) method**, a well-established numerical technique for solving ordinary differential equations (ODEs). This is particularly important in modeling systems with nonlinear and chaotic behavior.

Consider the general initial value problem (IVP):

The **Runge–Kutta method** approximates the solution given the current value using a series expansion derived from the Taylor series:

Let then the expansion can be written as:

To generalize the process, the **q-stage Runge–Kutta method** is introduced:

where is a weighted sum of intermediate slopes

Thus, the next value is computed as:

Each slope is calculated using the recursive formula:

For explicit Runge–Kutta methods (which do not require solving nonlinear equations at each step), the method simplifies with known coefficients , yielding the final update:

(

To simulate the dynamic behavior of the proposed memristive chaotic system, we employed the **fourth-order Runge–Kutta (RK4) numerical integration method**.

Let the state vector

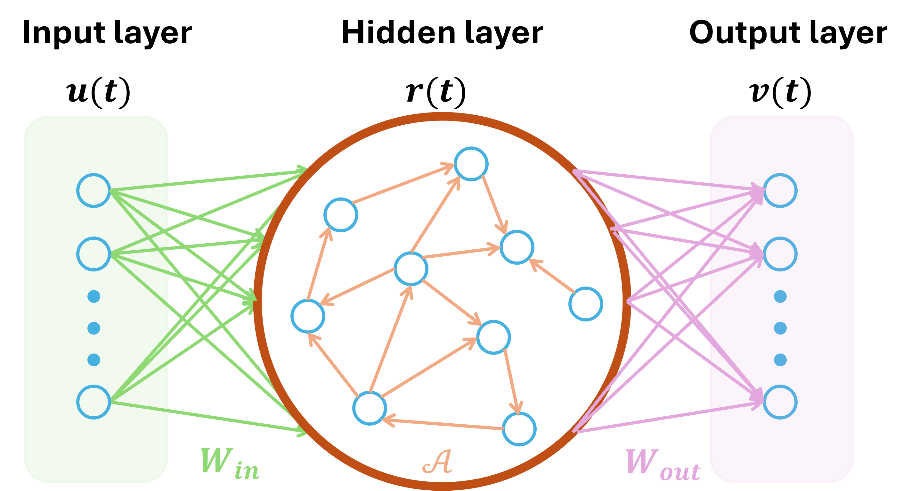
To update the state vector over time using RK4 with a time step h, we compute:

**3.3 MODEL DESCRIPTION**

Reservoir computing is a framework originally proposed in the early 2000s for training recurrent neural networks (RNNs). Currently, it is a vibrant and emerging AI research domain that has recently gained wide attention due to its extremely low training complexity. RC is an overarching concept that encompasses different RNN models, such as ESN, LSM, and the single node delayed feedback dynamical model. For this paper we will be focusing on the ESN because its robust to nonlinear, chaotic dynamics which is good for modeling memristive behavior.

**3.31 Echo State Network***.*

In this study, the ESN is designed to predict the next values of based on their past values, leveraging the reservoir’s ability to capture temporal and nonlinear patterns. The ESN consists of three components: the input-to-node layer, the node-to-node (reservoir) layer, and the node-to-output layer. These are described below, with parameters chosen to balance performance and simplicity for the chaotic system.



## **3.3.1.1 Input-To-Node**

The input-to-node layer connects the external input, which is the time-series of the chaotic system, to the reservoir. The input at time is a four-dimensional vector.

,

representing the current state of the memristive system. This vector is multiplied by an input weight matrix.

,

where is the number of nodes in the reservoir, where the reservoir size used is 500 Nx=500N\_x = 500. The reservoir state x(t)∈R500x(t) \in \mathbb{R}^{500}

where is the reservoir’s internal weight matrix, is a bias vector (set to zero), and is the hyperbolic tangent function, which introduces nonlinearity to capture the system’s terms. The input weights are randomly initialized from a uniform distribution over tanh⁡\tanh [-0.1,0.1], with a scaling factor of 0.1 to prevent the saturation the tanh⁡\tanhtanh function. This scaling is critical for handling the rapid changes in the chaotic system, such as those caused by The reservoir size () balances computational efficiency and the ability to model complex dynamics, suitable for implementation on a standard laptop.

When the echo state layer possesses this property, it can create a high-dimensional, nonlinear representation of the input signals, allowing the simple readout layer to learn the desired output more easily.

The reservoir size, spectral radius (the largest absolute eigenvalue of W), and sparsity are critical hyperparameters that influence the network’s memory and nonlinearity. Proper tuning of these parameters ensures the echo state property, where the influence of past inputs fades over time, allowing stable and meaningful dynamic responses.

## **3.3.1.2 Node to Node Transformation.**

The node-to-node layer, or reservoir, is a recurrent neural network with fixed, sparse connection. The reservoir weight matrixW∈R500×500W \in \mathbb{R}^{500 \times 500} , has a sparsity of 10%, meaning only 10% of its elements are non-zero, reducing computational complexity. Non-zero weights are drawn from a uniform distribution over [−1,1][-1, 1][-1, 1] and scaled to achieve a spectral radius ρ=0.9\rho = 0.9= 0.9, ensuring the echo state property. This property guarantees that the reservoir’s dynamics depend only on the input history, not its initial state, which is essential for stable predictions of the chaotic system. This property guarantees that the reservoir’s dynamics depend only on the input history, not its initial state, which is essential for stable predictions of the chaotic system. The reservoir state evolves according to Equation (2). The recurrence in allows the reservoir to “remember” past inputs, capturing temporal dependencies like the influence of

Th function mimics the system’s nonlinearities, such as tan⁡(z)\tan(z)tan(z) and the memristive term The reservoir was implemented in Python using the NumPy library, with sparsity enforced by setting 90% of ( W )’s elements to zero. The spectral radius ρ=0.9\rho = 0.9 = 0.9 was chosen based on guidelines for chaotic systems [Lukoševičius, 2012], ensuring the reservoir captures the oscillatory and chaotic behavior.

## **3.3.1.3 Node to output Transformation.**

The node-to-output layer generates predictions for the chaotic system’s state,

**,** using the output weight matrix Wout∈R4×500W\_{\text{out}} \in \mathbb{R}^{4 \times 500} **:**

The output weights are trained using ridge regression, which minimizes the mean squared error between predicted and true states, with a regularization parameter to prevent overfitting. The data consists of 10000-time steps of the memristive chaotic system time series generated by numerical integration of the runge-kutta method in section 3.3. Ridge regression was implemented using scikit-learn’s Ridge library in python. Performance was evaluated using mean square error.

Where y\_i(t)are the predicted and true values of the state variables at time t , and ( N ) is the number of time steps.

## **3.3.3 Converting Data to Format for Prediction**

To enable the Echo State Network (ESN) to predict the time-series of the memristive chaotic system defined in Equation (1), the raw data must be preprocessed into a format compatible with ESN’s input and output requirements. This section details the steps to generate, normalize, and split the time-series for ), addressing numerical challenges from nonlinear terms like . To capture the system’s multistability—the coexistence of multiple attractors depending on initial conditions—time-series are generated from multiple initial conditions and averaged. The Lyapunov exponent, a key measure of chaotic behavior, is computed to quantify the system’s sensitivity to initial conditions, ensuring the ESN preserves this chaotic and multistable nature. The ESN’s hyperparameters, particularly node-to-node connections (spectral radius and sparsity of the reservoir weight matrix W , are optimized using a random search to minimize Mean square Error (MSE), as described in Section 3.3.2.2. Time-Series Generation: The system’s differential equations (Equation 1) were solved using SciPy’s odeint function with a fourth-order Runge-Kutta (RK4) method.

## **3.5 Lyapunov Exponent Calculation:**

The Lyapunov exponent measures the rate of divergence of nearby trajectories in a dynamical system, indicating chaos when positive. For the memristive system, the largest Lyapunov exponent was computed using the Rosenstein algorithm [Rosenstein et al., 1993], which tracks the divergence of two trajectories starting from slightly perturbed initial conditions. The exponent is approximated as the given equation below:

Where is the Euclidean distance between trajectories at time t, and

is the initial perturbation. The algorithm uses a transient period (1,000 steps) to eliminate initial effects and evaluates divergence over 5,000 steps. For the system with a=1,b=0.5,c=1a = 1, b = 0.5, c = 1a = 1, b = 2 c = 2, , confirming chaotic behavior. This metric is critical for multistability, as it ensures the ESN captures the system’s sensitivity across different attractors. The Jacobian matrix, used to compute local divergence, is.

This was implemented in python, the Lyapunov exponent guides the hyperparameter optimization by ensuring the ESN’s dynamics matches the system’s chaotic properties

## **3.4 Building models for reservoir computing**

## **λmax\lambda\_{\text{max}}****3.4.1 ESN model architecture**

The normalized time-series was converted into a matrix of shape ((10000, 4)), where each row represents a 4-dimensional vector at time ( t ). This matrix format is required for ESN input, where each row is processed sequentially to update the reservoir state. The ESN expects inputs u(t)∈R4u(t) \in \mathbb{R}^4and produces outputs y^(t)∈R4\hat{y}(t) \in \mathbb{R}^4, matching the system’s four variable. ESN architecture was modelled in the following ways. The time-series matrix was split into training set 7,000-time steps (70%, rows 0–6,999) to train the ESN’s output weights, validation set 1,500-time steps (15%, rows 7,000–8,499) to optimize hyperparameters. test set 1,500-time steps (15%, rows 8,500–9,999) to evaluate final performance. A washout period of 100-time steps was applied during training to eliminate initial reservoir transients, ensuring the reservoir state stabilizes before weight optimization. The ESN generates predictions by mapping input time-series to reservoir states and then to outputs. The process is:

where is the input weight matrix, and W∈RNx×NxW \in \mathbb{R}^{N\_x \times N\_x}is the reservoir weight matrix with spectral radius ρ\rhoand sparsity. The output weights Wout∈R4×NxW\_{\text{out}} \in \mathbb{R}^{4 \times N\_x} are learned using Ridge regression on the training set (after washout), minimizing the error between predicted outputs and target outputs n For validation and test sets, the ESN processes input ( u(t) ) to compute reservoir states ( x(t) ), then generates predictions

. Predictions are denormalized to the original scale for mean absolute error

## **3.4.2 parameter Optimization for reservoir computing:**

Hyperparameter optimization or tuning is essential for our ESN model to ensure the best optimal performance of our data in this case the memristive system, this was evaluated using the mean square error shown in. The hyper parameters optimized include the special radius, Reservoir size, input scaling, sparsity and Regularization.

To assess the influence of these parameters, a parameter sweep was conducted, varying one hyperparameter at a time while holding the others constant. For each configuration, the memristive system was simulated using a Runge-Kutta numerical integrator (RK45), and the ESN was trained to predict the normalized x(t) trajectory.

The prediction performance was evaluated using the **Mean Squared Error (MSE)** between the predicted and true trajectories. The results were visualized as plots of MSE versus each hyperparameter. These plots clearly reveal the trade-offs associated with different parameter choices and highlight regions of optimal performance. For example, the MSE generally decreased as the reservoir size increased up to a saturation point, while spectral radius values slightly below 1 yielded the best predictive capability.

## **CHAPTER FOUR**

## **RESULTS AND DISCUSSIONS**

## **4.1 MEMRISTIVE SYSTEM PREDICTIONS**

In this study, variations of the memristive system with time have been investigated, the mathematical expressions for solving the system numerically have been derived in the last section (chapter 3), and the numerical and computational results are discussed through the graphical illustration using python to find out the results.

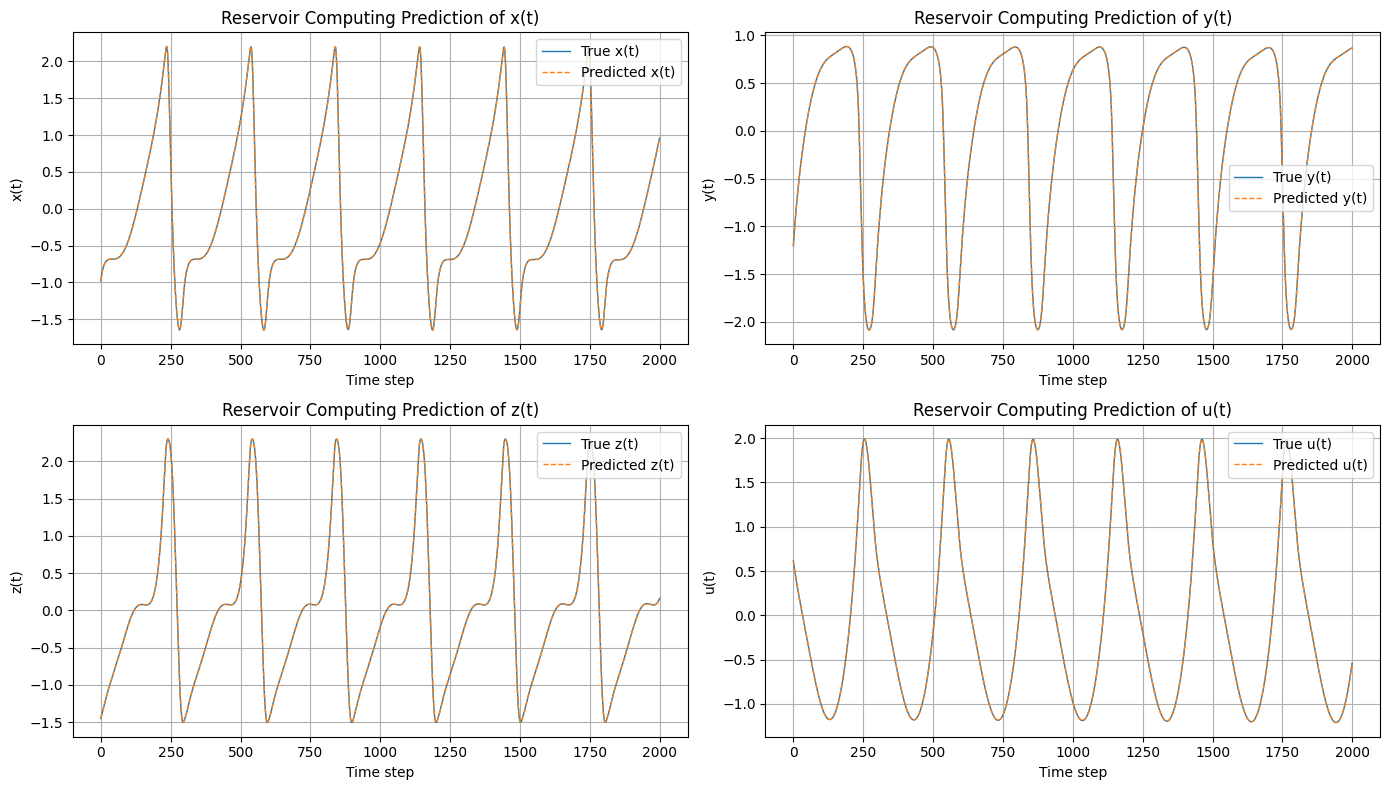


Figure 3: Shows the ESN prediction of our system

A group of blue lines

AI-generated content may be incorrect.

Figure 4: Shows the Prediction error with respect to time

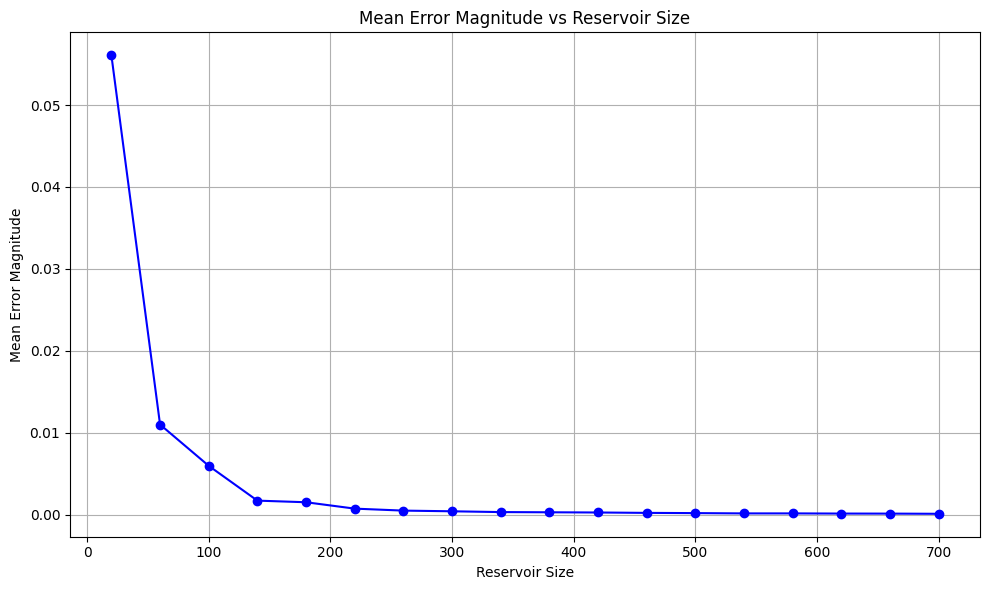


Figure 5: Reservoir size against Mean error magnitude

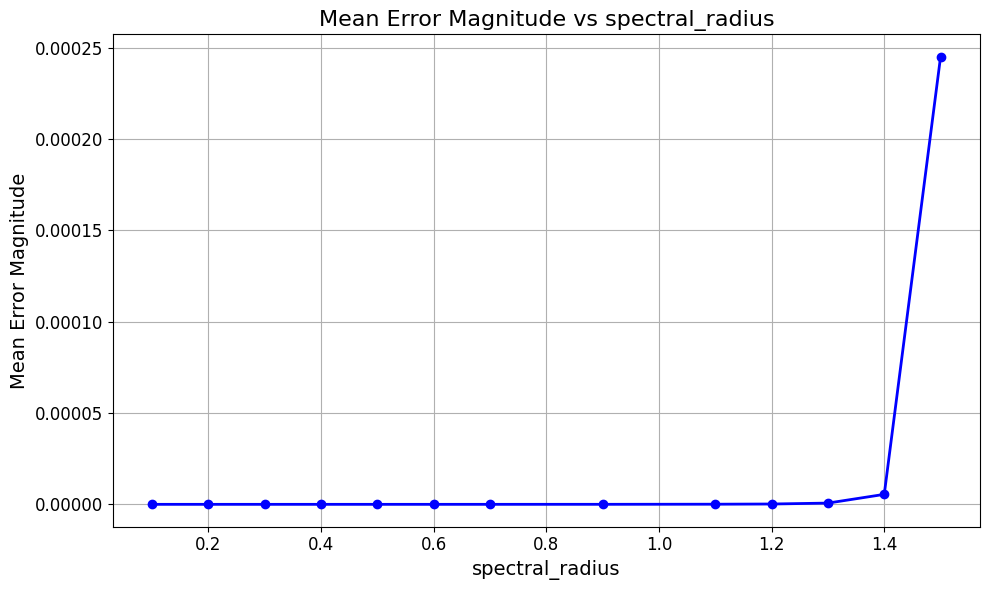
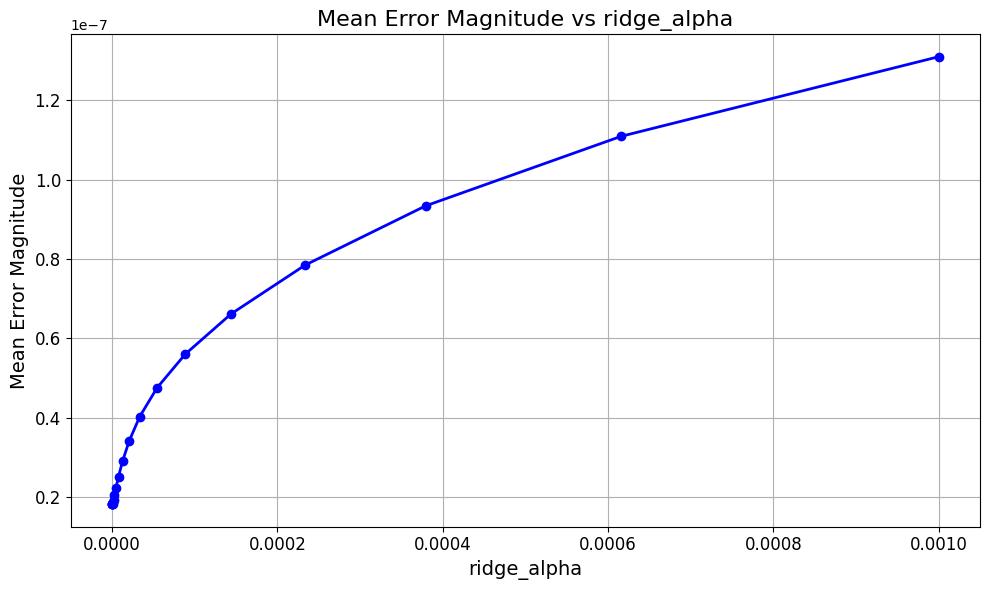


Figure 6: Spectral radius against Mean error magnitude



**Figure 7: ridge alpha against mean error magnitude**

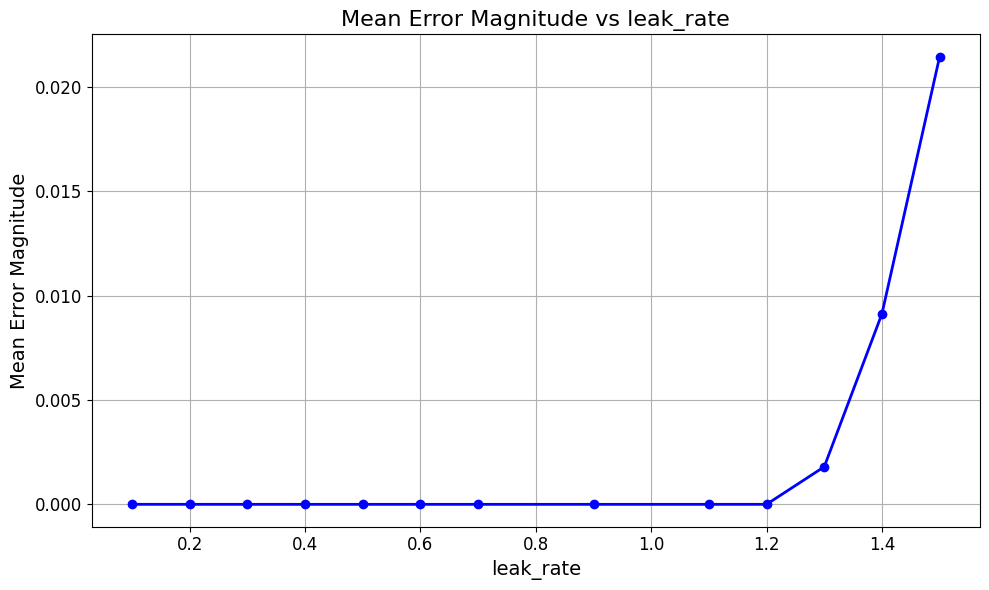


Figure 8: ridge leak rate against mean error magnitude

|  |  |  |
| --- | --- | --- |
| Exponent |  |  |
|  | +0.016 | Largest exponent (positive) |
|  | -0-0017 | Near Zero |
|  | -0.1504 | Moderately Negative |
|  | -3.8313 | Strongly Negative |

## **4.2** Lyapunov exponents result

Table 1 showing the Lyapunov exponents

## **4.3 DISCUSSIONS**

Fig 1: The 3D graph shows the complex dynamics of the system in the x-y-z phase space. It represents a chaotic attractor, which is a set of states toward which a system evolves over time. The intricate, seemingly non-repeating path shown in the graph is characteristic of a chaotic system, where trajectories diverge exponentially, indicating sensitive dependence on initial conditions.

Fig 2: This figure presents the **time-series plots** of the four state variables of the memristive chaotic VB2 system. Each subplot displays the evolution of one variable over time, helping to analyze the system’s **dynamical behavior**. Here's what each component means: variable shows sharp peaks followed by a slow decay, repeating over time. variable exhibits a slightly smoother but still nonlinear oscillatory behavior Appears bounded and non-periodic, reinforcing chaotic characteristics. on the other hand shows repeating structure and long-term dependency, confirming that past states influence current evolution, a hallmark of memristive memory effects**.**

Fig 3: **Report on Reservoir Computing Model Performance for Time Series Prediction**

This report presents a detailed analysis of the predictive performance of a reservoir computing model applied to a four-dimensional time series. The provided figure displays four subplots, each illustrating the model's prediction against the true values for the time series components x(t), y(t), z(t), and u(t) over 2000-time step. In all four subplots, the predicted values (the dashed orange line) closely track the true values (the solid blue line). The overlap between the true and predicted signals is visually almost perfect, indicating a very high degree of accuracy.

**Fig 4 error Distribution:**

The errors oscillate around zero, with no systematic bias. This suggests that the prediction is not drifting consistently above or below the true trajectory instead, the deviations are small, random-like fluctuations. Since chaotic systems are highly sensitive to initial conditions, even very small differences can grow quickly if the model is not accurate. The fact that errors remain bounded at such a low scale shows that your RC has successfully captured the short-term chaotic dynamics.

**Fig 5-9**

Figures 5 to 9 present a comprehensive evaluation of how varying key hyperparameters affect the prediction performance of the Echo State Network (ESN) model, with performance measured using the Mean Squared Error (MSE). Each graph illustrates the relationship between a specific hyperparameter and the resulting prediction error on the memristive chaotic dataset.

**Figure 5** demonstrates the effect of reservoir size on model performance. It is observed that as the number of reservoir neurons increases, the MSE decreases consistently. This indicates that larger reservoirs are better equipped to capture the nonlinear temporal dependencies inherent in chaotic systems. The error decreases rapidly until the reservoir size reaches about 200. After this point, the error becomes very small, and the line flattens out. Increasing the reservoir size from 200 to 700 results in almost no further reduction in error. This suggests that a reservoir size of around 200 is a good choice, as it provides high accuracy without the unnecessary computational cost of a much larger model.

**Figure 6** demonstrates a pattern like that of spectral radius. For leak rates up to approximately 1.2, the Mean Error Magnitude is very low and stable. This range represents a sweet spot where the reservoir's internal states are being updated at an optimal rate, allowing it to effectively remember past inputs without being overly influenced by them. However, for leak rate values greater than 1.2, the Mean Error Magnitude increases exponentially. This sharp rise indicates that as the leak rate becomes too high, the reservoir's memory of past input decays too quickly, and the system loses its ability to represent the necessary temporal dependencies of the input data.

**Figure 7** shows a clear trend as the value of ridge alpha increases; the Mean Error Magnitude also increases. This suggests that the regularization applied to the readout weights, which is controlled by ridge\_alpha, is a critical factor in model accuracy. The smallest error is observed at the lowest values of ridge\_alpha tested (approaching 0). This implies that for this specific problem and model configuration, a very small amount of regularization is optimal, and increasing regularization beyond this point leads to a decrease in model performance. The relationship appears to be approximately logarithmic, with the error increasing more rapidly at lower ridge\_alpha values and then leveling off.

**Figure 8** shows a similar pattern in that of the spectral radius. For leak rates up to approximately 1.2, the Mean Error Magnitude is very low and stable. This range represents a sweet spot where the reservoir's internal states are being updated at an optimal rate, allowing it to effectively remember past inputs without being overly influenced by them. However, for leak rate values greater than 1.2, the Mean Error Magnitude increases exponentially. This sharp rise indicates that as the leak rate becomes too high, the reservoir's memory of past input decays too quickly, and the system loses its ability to represent the necessary temporal dependencies of the input data.

## **CHAPTER FIVE**

## **CONCLUSION**

In this work, we introduced a novel computational framework inspired by the physics of memristive devices and systems, leveraging their memristive dynamics to design neural architectures suitable for time-series processing. We presented Echo State Network ESN) which combines the principles of reservoir computing to model and predict the system even with its chaotic hypermultistability behaviors. Our analysis provided a mathematical characterization of the stability of these models, and extensive computational experiments were conducted to evaluate their performance across a diverse set of time-series classification and regression tasks.

**NOTATIONS**

— Lyapunov Exponent— Frequency of forcing function (rad/s) b — Damping coefficients *β* — Damping coefficients *α* — Damping coeefficients

## **REFERENCES**

1. L. O. Chua, “Memristor—The missing circuit element,” *EEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.
2. Ajayi, A.A., Ojo, S.K., Vincent, E.U., Njah, N.A., 2014. Multi-switching synchronization of a driven hyperchaotic circuit using active backstepping. J. Nonlinear

Dynamics. 2014. Available from: http://dx.doi.org/10.1155/2014/918586.

1. S.H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, “Nanoscale memristor device as synapse in neuromorphic systems,” *Nano Letters*, vol. 10, no. 4, pp. 1297–1301, 2010.
2. J. M. Nyandieka, et al., 2023. *A Chaotic Multi‐Objective Runge Kutta Optimization Algorithm for Engineering Design Problems*. Mathematical Problems in Engineering, vol. 2023, Article ID 9973701.
3. Lakshmanan, M., Murali, K.: Chaos in Nonlinear Oscillators: Controlling and Synchronization. World Scientific, Singapore (1996).
4. A. Wolf, J. B. Swift, H. L. Swinney, J. A. Vastano, 1985. *Determining Lyapunov exponents from a time series*. Physica D: Nonlinear Phenomena, 16(3), pp. 285–317.
5. Veronica Pistolesi, Andrea ceni,Gianluca Milano, **Carlo Ricciard,** **Claudio Gallicchio. A memristive computational neural network model for time-series processing.** [**https://doi.org/10.1063/5.0255168**](https://doi.org/10.1063/5.0255168)
6. Y.C Kouomou, Pere Colet, Nicolas Gastaud, Laurent Larger. 2021. Effect of parameter mismatch on the synchronization of chaotic semiconductor lasers with electro-optical feedback.
7. A.L. Fradkov, Institute for Problems of Mechanical Engineering, Russian Academy of Sciences, St. Petersburg, RUSSIA. 2009 CONTROL OF CHAOTIC SYSTEMS
8. Michael R. A. Huth. Imperial College of Science, Technology, and Medicine. Secure Communicating Systems; Design, Analysis, and implementation.
9. L. M. Pecora and T. L. Carroll, "Synchronization in Chaotic Systems, Physical Review Letters, vol. 64, pp. 821–825, 1990.
10. Cen Wang, Xinyao Lei, Kaiming Cai, Xu Ge, Xiaofei Yang, Yue Zhang,” Improved long-term prediction of chaos using reservoir computing based on stochastic spin–orbit torque devices,”
11. C. W. Wu and L. O. Chua, "A simple way to synchronize chaotic systems with applications to secure communication systems," Int. Journal of Bifurcation and Chaos, vol. 3, pp. 1619–1627, 1994.
12. Jiayu Sun Akif Akgul, Chunbiao Li , Tianai Lu , And Fuhong Min , “A Memristive Chaotic System With Hypermultistability and Its Application in Image Encryption”. Digital Object Identifier 10.1109/ACCESS.2020.3012455
13. Njah, A.N., Ojo, K.S.: Backstepping control and synchronization of parametrically and externally excited van der Pol oscillators with application to secure communication. Int. J. Mod. Phys. B 23, 4581–4593 (2010).
14. Jiang, C., Liu, S., Wang, D.: Generalized combination complex synchronization for fractional-order chaotic complex systems. Entropy 17, 5199–5217 (2015).
15. Pecora, L.M. and T.L. Carroll, (1990) “Synchronization in chaotic systems,” Phys. Rev. Lett., vol. 64, pp. 821–823. [A paper introducing the idea of master-slave synchronization of chaotic systems and its possible application to secure communications.]
16. Pyragas K. (1992) “Continuous control of chaos by self-controlling feedback”. Phys. Lett. A., vol. 170, pp.421–428. [A paper introducing the idea of continuous-time delayed feedback.]
17. Wu C.W. and L.O.Chua, (1994), “A unified framework for synchronization and control of dynamical systems, Intern. J. Bifurcation and Chaos , vol.4, 979-998. [Unifying synchronization and control problems for chaotic systems based on goal-oriented and

Lyapunov functions approaches.]

1. Kapitaniak T. (2000) Chaos for Engineers. 2nd edition. Springer-Verlag. [Simple introductory exposition of some methods of control and synchronization of chaos.].
2. F.T. Arecchi, S. Boccaletti, M. Cio"ni, C. Grebogi, R. Meucci (Eds.), Theme Issue: Control of Chaos: New Perspectives in Experimental and Theoretical Nonlinear Science, Part I, Int. J. Bifurcation Chaos 8 (8) (1998).
3. F.T. Arecchi, S. Boccaletti, M. Cio"ni, C. Grebogi, R. Meucci (Eds.), Theme Issue: Control of Chaos: New Perspectives in Experimental and Theoretical Nonlinear Science, Part II, Int. J. Bifurcation Chaos 8 (9) (1998).
4. Y.-C. Lai, C. Grebogi, T. TeHl, Controlling transient chaos in dynamical systems, in: Towards the Harnessing of Chaos, the 7th TOYOTA Conference, Elsevier,

Amsterdam, 1994, pp. 153, 167.