

Prairie View A&M University

Digital Commons @PVAMU

[All Theses](#)

5-2023

Digital Twin Modeling And Optimal Control Of Soft-Bodied Robotics Using Reservoir Computing

Jovan Cain

Follow this and additional works at: <https://digitalcommons.pvamu.edu/pvamu-theses>

DIGITAL TWIN MODELING AND OPTIMAL CONTROL OF SOFT-BODIED
ROBOTICS USING RESERVOIR COMPUTING

A Thesis by
JOVAN CAIN

Submitted to the Office of Graduate Studies of Prairie View A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2023

Major Subject: Electrical Engineering

DIGITAL TWIN MODELING AND OPTIMAL CONTROL OF SOFT-BODIED
ROBOTICS USING RESERVOIR COMPUTING

A Thesis by
JOVAN CAIN

Submitted to the Office of Graduate Studies of Prairie View A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

Dr. Xiangfang Li
Chair of Committee

Dr. Pamela Obiomon
Committee Member

Dr. Lijun Qian
Committee Member

Dr. Pamela Obiomon
Dean, Roy G. Perry College of
Engineering

Dr. Richard Wilkins
Committee Member

Dr. Annamalai Annamalai
Head of Department

Dr. Tyrone Tanner
Dean, Graduate Studies

May 2023

Major Subject: Electrical Engineering

ABSTRACT

Digital Twin Modeling and Optimal Control of Soft-bodied Robotics using
Reservoir Computing

(May 2023)

Jovan Cain, B.S., Prairie View AM University, Prairie View, Texas Chair of
Advisory Committee: Dr. Xiangfang Li

Soft-bodied robots have become increasingly popular due to their ability to perform tasks that are difficult or impossible for traditional rigid robots. However, accurately modeling and controlling the movement and behavior of soft robots are very challenging due to their complex and dynamic nature. In recent years, Reservoir Computing has emerged as a promising approach to modeling and controlling soft robots. In this thesis, reservoir computing was used to create a digital twin of soft-bodied robots. Specifically, a digital twin of a spring-mass system was created using echo state network, a popular reservoir computing model. Furthermore, an optimal controller was trained using reservoir computing to drive the spring-mass system to follow a desired trajectory. Extensive simulations were carried out to validate the proposed methods. The results demonstrate the effectiveness of the proposed approach. For example, the digital twin model achieved 2% MAPE and the optimal controller achieved 8.7% MAPE for a 20-node 54-spring system.

Index Terms: Deep Learning, Digital Twin Modeling, Echo State Network, Optimal control, Soft Bodied Robotics, Time series prediction

ACKNOWLEDGMENTS

To start, I want to give thanks to God for being a source of strength and guidance throughout my studies. I am grateful for the grace and privilege I have received during this research and throughout my graduate program. I am also grateful for my supervisor and advisor, Dr. Xiangfang Li. Your support, advice, and willingness to give knowledge were not unnoticed and very much appreciated. Special thanks to Dr. Mulugeta Haile, an Aerospace Research Lead at Army Research Laboratory, Dr. Xishuang Dong, and other members of the Center of Excellence in Research and Education for Big Military Data Intelligence (CREDIT Center) for their knowledge, advice and support throughout the completion of this research. I also immensely appreciate the members of my thesis committee, Dr. Pamela Obiemon, Dr. Richard Wilkins and Dr. Lijun Qian for their unrelenting support towards the timely completion of my thesis. Finally, I want to specially appreciate my father, Brentan Cain, mother, Erica Cain and my brother, Jeremy Cain for their love, understanding, and blessings. Simply, your phone calls and words of encouragement and prayers kept me going in this journey of pursuing my degree.

This research work is supported in part by the U.S. Army Research Office (ARO) under agreement number W911NF-20-2-0266 and the U.S. Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) under agreement number FA8750-15-2-0119. The United States Government is

authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Army Research Office (ARO) or the Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) or the United States Government.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGMENTS.....	iv
TABLE OF CONTENTS.....	vi
LIST OF FIGURES	ix
LIST OF TABLES.....	xi
1. INTRODUCTION	1
1.1 Background Overview.....	1
1.2 Levels of Complexity.....	2
1.2.1 Digital Computer.....	3
1.2.2 Analog Devices	3
1.2.3 Biological Systems.....	4
1.3 Reservoir Computing	5
1.4 Digital Twin	6
1.5 Problem Statement.....	8
1.6 Significance of Work	9
1.7 Outline of the Thesis	10
2. LITERATURE REVIEW.....	11
2.1 Background.....	11
2.2 Deep Learning	12
2.3 Reservoir Computing	13
2.3.1 Physical Reservoir Computing.....	15
2.4 Digital Twin	16
2.5 Robotic Control.....	18
3. METHODOLOGIES	20
3.1 Methods in Deep Learning.....	21
3.1.1 Supervised Learning.....	22
3.1.2 Classification	23
3.1.3 Regression	24
3.2 Deep Learning Models	25
3.2.1 Deep Neural Networks	26
3.3 Recurrent Neural Networks	28

3.3.1	Long Short Term Memory (LSTM)	29
3.3.2	Echo State Network	30
3.4	Reservoir Computing	31
3.4.1	Architecture	34
3.4.2	Echo State Property	34
3.4.3	Reservoir Computing Model	35
3.4.4	Physical Reservoir Computing	36
3.5	Mass-Spring Networks	37
3.5.1	Optimization	39
3.6	Digital Twin Reservoir Computing Model	39
3.6.1	Optimal Control	41
4.	EXPERIMENTAL RESULTS & PERFORMANCE ANALYSIS.....	44
4.1	Data Generation.....	44
4.2	Experiment Set-Up.....	46
4.2.1	Dynamics Simulation Set-Up.....	46
4.2.2	Control Experiment Set-Up	52
4.3	Evaluation Metrics	53
4.4	In-depth Performance of Reservoir Computing for Optimal Dynamics...54	
4.4.1	Comparison of Performance between Deep Learning Models.....	54
4.5	Performance of Reservoir Computing Models with different Set-Ups.....56	
4.5.1	Reservoir Sizes	57
4.5.2	Initialization Methods.....	58
4.5.3	Readout Methods	69
4.6	Performance of Enhanced Reservoir Computing Models.....	60
4.6.1	Pre-Trained Models	60
4.6.2	DeepESN	61
4.7	In-depth Performance of Reservoir Computing for Optimal Control.....	62
4.7.1	Performance of Reservoir Computing Model	63
4.7.2	Performance of Different Reservoir Sizes.....	66
4.7.3	Performance of Different Initialization Methods	67
4.7.4	Performance of Different Readout Methods.....	68
4.8	Summary of Results	69
5.	CONCLUSION AND FUTURE WORK.....	71
5.1	Conclusion	71

5.2 Future Work.....	72
REFERENCES.....	73
CURRICULUM VITA.....	81

LIST OF FIGURES

FIGURE	Page
1.1 A Reservoir Computing Architecture on a Physical Body [1]	6
3.1 Architecture of a Deep Neural Network [2]	26
3.2 Architecture of an LSTM [3]	29
3.3 Architecture of an Deep Echo State Network with Successive Reservoirs [4]	31
3.4 A typical Reservoir Computing Architecture [5]	32
3.5 Depiction of a Software Reservoir Computing Model for Soft-Robotic Structures	36
3.6 Flow of a Physical Reservoir Computing Model	37
3.7 An example of a 20 node structure with attached springs [4]	38
3.8 Flow of a Digital Reservoir Computing Model for Digital Twin Modeling	41
3.9 Flow of a Reservoir Computing Model for Optimal Control in Digital Twin Modeling	43
4.1 20 node structure with modeled nodes derived from [4]	47
4.2 Evolution of Node 1 Dynamics over 20,000 time steps	48
4.3 Evolution of Node 11 Dynamics over 20,000 time steps	51
4.4 Detailed View of X-Position Evolution over 2,000 time steps	51
4.5 Sample Plot of RC Model Predicted Data	56
4.6 Plots of Model Performance on Different Reservoir Sizes	58
4.7 20 node structure with modeled springs	63
4.8 Tested Spring Control Signals over 20,000 time steps.....	66

LIST OF TABLES

TABLE	Page
4.1 Dynamics Simulation Training Data Description	45
4.2 Control Simulation Training Data Description	46
4.3 Control Simulation Data Description.....	53
4.4 Model Parameters for Different Models.....	55
4.5 Performance of the Deep Learning models on each Dataset.....	56
4.6 Model Performance Comparison for Different Reservoir Sizes	57
4.7 Model Performance Comparison for Different Initialization Approaches	59
4.8 Model Performance Comparison for Different Readout Methods	60
4.9 Model Performance Comparison for Pre-Trained Models.....	61
4.10 Model Performance Comparison for Deep Echo State Network	62
4.11 Performance of RC Model on Control Parameters.....	64
4.12 Model Performance Comparison for Different Reservoir Sizes	67
4.13 Model Performance Comparison for Different Initialization Approaches	68
4.14 Model Performance Comparison for Different Readout Methods	69

CHAPTER 1

INTRODUCTION

1.1 Background Overview

Soft-bodied robotics is a subfield of robotics that focuses on the design, fabrication, and control of robots made of soft, flexible materials. Unlike traditional robots, which are typically made of hard materials such as metal and plastic, soft robots are made of materials that can bend, stretch, and deform, allowing them to interact with their environment in unique ways. The use of soft-bodied robots can be beneficial for several reasons. Soft robots can be more adaptable and versatile than traditional robots. Because they can deform and change shape, they can navigate through tight spaces, squeeze through small openings, and interact with delicate objects without causing damage. This makes them suitable for a range of applications, including medical procedures, search and rescue operations, and exploration. Also, soft-bodied robotics has the potential to improve the safety of robots. Because soft robots are less likely to cause harm to humans or damage the environment, they can be used in settings where traditional robots would be too dangerous. Due to the potential innovation and benefit, this has become increasingly popular in interest due to its complexity in modeling and dynamics.

This thesis follows the *Journal IEEE Style Manual*

Soft robots are highly nonlinear. This means that their behavior can be difficult to predict and model. This is due to the complexity of structures and spring stiffness can change depending on external forces and pressure, thus changing the robot's movement and behavior. Also, soft robots are often subjected to multiple external physical forces and interactions. To model these forces and potential interaction instances accurately require advanced modeling techniques and lots of computational resources. Another challenge is that soft robots often require complex control strategies to achieve desired mechanics, such as locomotion. To greatly perform modeling performance, developing optimal control strategies can reduce the impact of nonlinear behavior.

To model this nonlinear behavior, I propose the use of various Deep Learning and Machine Learning techniques. Deep learning is a powerful tool for modeling highly nonlinear dynamics because it is capable of automatically learning complex patterns and relationships in large datasets without requiring explicit knowledge of the underlying physics. This makes it particularly well-suited for modeling complex systems with nonlinear behaviors, such as soft robots. Generally, deep learning models, such as Recurrent Neural Networks, are composed of multiple layers of internal memory and processing units, each performing nonlinear transformations. In using different layers and approaches, deep learning models can learn to represent highly nonlinear and complex functions.

1.2 Levels of Complexity

Overall, deep learning is a powerful tool for modeling highly nonlinear

dynamics in a wide range of applications, including soft robotics. By leveraging the power of deep learning, we can develop more accurate and robust models of soft robot behavior, which can help to advance the field and enable new applications. However, such robust tasks can be difficult to model and tune. In addition, more complex tasks require more computational power which can provide another level of hardware complexity. There are different levels of complexity in regard to modeling and control of soft robotics discussed below.

1.2.1 Digital Computer. The specific dynamical system (hardware) for computing depends on the application, requirements, and constraints. The digital computer is the most common among all the available artificial hardware systems. A standard digital computer has decent enough computing power to run various applications and model various complex tasks. They are characterized by the transmission of information by discrete and binary signals. This makes them less susceptible to noise while maintaining good data transmission bandwidth with efficient data storage and high scalability [6]. However, for soft robots, this approach may not be feasible as the digital computer does not have enough computing power and hardware capabilities to efficiently and effectively control robots. For purposes of modeling robots, this approach can be beneficial as decent computers can provide enough computing power and speed.

1.2.2 Analog Devices. Systems implemented with electronic circuits and devices have been actively studied for developing machine learning devices with low training costs. Generally, existing neural networks and circuits are available

as reservoirs in the form of an analog device. However, simpler configurations have been explored to reduce energy consumption, speed up computation, and handle noise in hardware [5]. More specifically, field-programmable gate arrays (FPGAs) and graphics processing units (GPUs) can be used to accelerate the training and prediction of machine and deep learning models used for controlling robots. By leveraging the parallel processing capabilities, it is possible to train more complex and accurate models in less time [7]. This can help to improve the performance and adaptability of soft robots in a variety of applications.

FPGAs have often been used to implement neural networks as their architecture is well suited for complex processing and adaptive weight updating [8][9] which can be very beneficial for the adaptability of soft robots. Many FPGA implementations have been studied to improve reservoirs and/or readouts [10]. This implementation can be very good for certain tasks as GPUs and FPGAs have valuable computational resources that can be innovative and important for speed and accuracy.

1.2.3 Biological Systems. The system responsible for the computing ability of the brain is vastly studied and researched, due to its ability to affect AI. Considerable efforts have been made to explain the computing function using a link to computational models in artificial neural networks (ANN) and machine learning. In attempts to understand the relationship between temporal information processing in the brain and RC, researchers have speculated about which part of the brain can be regarded as a reservoir or a readout as well as about how sub-networks of the brain

work in the RC framework.

1.3 Reservoir Computing

Reservoir Computing (RC) is a computational framework that is well-suited for sequential data processing. This framework can be used in parallel with recurrent neural network models, including echo state networks (ESN) and long short-term memory networks (LSTM). The key advantage of reservoir computing is that the 'reservoir' is a fixed set of nodes and connections and does not need to be trained. In this sense, reservoir computing can be much faster and more efficient than traditional RNNs. The ability to perceive the environment and receive feedback from the system, also makes RC a great approach for the purpose of embodied mechanical intelligence. Reservoir computing [11] can be used in soft robotics to process sensory input from the robot's environment and control the robot's movements. Soft robots often have many degrees of freedom, which can make traditional control techniques difficult to implement. However, reservoir computing can provide an effective way to control the complex and nonlinear dynamics of soft robots [12]. Mechanical systems, such as soft and compliant robots, are possible options for reservoirs. Soft and compliant robots with flexible bodies are difficult to control due to their complex body dynamics compared with rigid robots with stiff bodies. From the general principle of reservoir computing, modeling dynamical systems with relatively simple adaptive controls can be implemented for complex systems. A mass-spring network can be used as the reservoir where mass points are randomly connected to neighboring mass points with

nonlinear springs was proposed in [13], providing a foundation for reservoir computing as I can model the dynamics of the system. As shown in Fig. 1.1 , reservoir computing principles can be used to effectively control dynamic robot bodies as the system can sense feedback through the body.

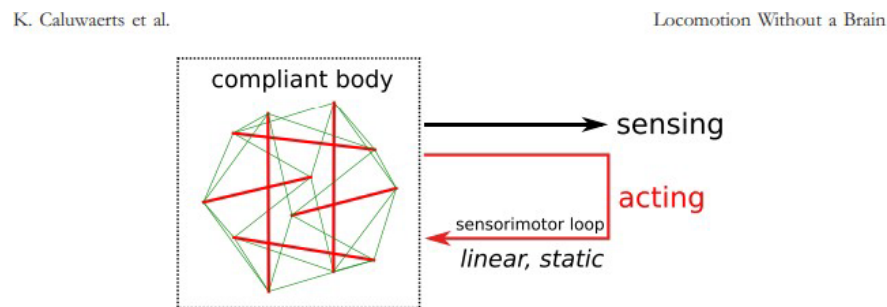


Fig. 1.1. A Reservoir Computing Architecture on a Physical Body [1]

1.4 Digital Twin

A digital twin is essentially a virtual simulation of a physical object or system. It uses real-time data and modeling to create a digital representation of the object, which can be used for analysis, testing, and optimization. By creating a virtual replica of a system, digital twins can be used to optimize the performance and control of the system. Digital twins are becoming increasingly utilized in many industries, including manufacturing, aerospace, and healthcare. For the purposes of soft-bodied robotics, this concept can be very beneficial as I can model the robot and certain features virtually. Studies have shown [14] how this

concept can be used for very complex models in AI applications. The potential to simulate dynamic systems makes digital twins a very powerful concept for modeling soft-bodied robots.

The proposed digital model can be used to simulate the behavior of the system in real-time, providing data to optimize its design. The ability to model highly dynamic systems comes with various challenges, however. Modeling complex dynamics requires a lot of data, and that data often comes from multiple sources, especially for more complicated structures. Also, complex dynamics often involve many variables and nonlinear interactions between those variables. This can make modeling and simulation computationally intensive, requiring significant processing power and memory. Last, in order to be truly useful, a digital twin model needs to be able to simulate the behavior of the physical system in real-time. Achieving real-time simulation requires a highly optimized model and a powerful computing infrastructure.

To achieve great performance given these challenges, various machine and deep learning concepts have been applied. Digital twins can be enhanced using reservoir computing. Reservoir computing can be used to create a dynamic model of the system that is simulated, which can improve the accuracy of the digital twin over time. In the RC framework, the reservoir can be used to capture the complex and nonlinear behavior of the physical system, which can then be used to simulate behaviors like locomotion [15]. The model can then be trained to allow it to learn and adapt to changing conditions. Reservoir computing can also be used to optimize

the parameters of the digital twin, such as the structure of the object being modeled and the architecture of the RC model itself. This can be beneficial in cases where the robotic system is complex or has many degrees of freedom, making it difficult to control using traditional techniques.

1.5 Problem Statement

In recent years, the power of artificial intelligence (AI) and machine learning (ML) has been leveraged to solve many complex problems and scenarios, such as natural language processing and even weather forecasting. As technology is constantly changing to solve emerging and complex problems, utilization of this technology is imperative. The goal of soft robotics is the design and construction of robots with physically flexible bodies and electronics. Robots with entirely soft bodies have tremendous potential. Compared to standard robots, soft robots have specific characteristics which bring multiple benefits. Soft robots can go places and do things those hard-body robots cannot. Also, soft robotics also provide a more delicate approach as robots are less rigid. Modeling these systems helps us understand soft robots better as they can be recreated with more real-life movements.

It has been proved in recent works that Reservoir Computing is a viable method for modeling highly complex and dynamic systems [16]. Instead of a conventional recurrent structure, various systems that utilize their nonlinear dynamics to work as computational resources have been discussed extensively. In fact, any dynamical system has the potential to be a reservoir for information

processing [17]. It has been observed in [18] how the concept of Reservoir Computing can be used to simulate various systems efficiently. While the embodiment of physical systems can be beneficial, it does come with its problems. In completing such tasks explained, the models have long training times and high computational costs. In addition, there is not an abundance of data available on the topic. In order to produce a highly effective model, many data points are needed for analysis. The levels of complexity, mentioned in Section 1.2, in solving these issues also provide a challenge.

This study aimed to alleviate these issues by developing an architecture to build a Reservoir Computing model with good predictive ability using simulated data, leveraging the capability of machine learning and deep learning approaches. This will potentially allow for use in many real-time physical applications in which high accuracy, low training time and computational costs are crucial.

1.6 Significance of Work

In the study of Reservoir Computing, developing an accurate and robust model using simulated data can have many benefits:

1. Many physical systems can be simulated, modeled and evaluated with less time and high predictive capability, mitigating the need for potential real-time data.
2. Tests carried out can be utilized across different levels of complex tasks, allowing for more complex structures.
3. Leveraging AI and ML to design an approach using various machine and deep learning models to model and evaluate structures effectively.

1.7 Outline of the Thesis

The remaining part of this study is structured as follows: Chapter 2 contains the Literature Review where some relevant research works in the recent past are discussed. Chapter 3 discusses the methodologies and practices utilized in this study. This focused on the process of creating the model and how to potentially improve it. Chapter 4 gives a detailed description of the data and of all the experiments carried out in this research, ranging from the data generation and collection process to model development and training processes. The results from these experiments are also discussed in this chapter. Chapter 5 concludes this study by summarizing the contributions of this work and highlighting potential research areas as future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Background

Soft robotics is a field that is rapidly growing in research and design as it can provide many advancements in technology. In contrast to hard-body robots, the softer material makes these robots more flexible and well-suited for tasks such as exploring uneven terrain and navigating tight spaces. Overall, the increasing popularity of soft robotics is driven by a combination of technological advancements, growing demand for flexible and adaptable robots, and a desire to create safer, more durable, and cost-effective robotic systems. Over the years, more study has gone into explaining the learning abilities of natural agents through reference to their bodily structure, and also to make extended use of this morphology for the learning of natural agents to simulate the same physics and interactions in artificial agents [11]. The control of this morphology has been studied for soft robotics to make them more optimal for challenging tasks. Modeling these control dynamics can provide great insight into physical structures and how they interact with environments.

For the purpose of control, many ideas and approaches have been applied to receive the desired locomotion and dynamics. Deep learning is widely utilized due to its effectiveness in data processing in a variety of fields. With respect to soft robotics, some studies have been also conducted based on deep learning

methods. In previous works [19][20][21], deep neural networks have proven to have the ability to accurately model features such as touch and gait recognition.

However, more complex methods may require a different combination of techniques.

The RC framework has been widely utilized and researched in the field, using various digital and physical architectures as the sensing body. In addition, other essential techniques were studied and implemented in this research.

2.2 Deep Learning

Modeling and control of soft robots may allow systems to operate in a fashion where it can accomplish tasks that are otherwise impossible for people and hard-bodied robots. Unfortunately, developing accurate models for complex dynamics may be high in cost, prediction error, and training time. Due to this, researchers have utilized deep learning as it presents an alternative modeling approach that only requires a sequence of the system's inputs and states, which can be easily measured or estimated. However, [22], explains how fully relying on learned models involves collecting large amounts of data from a prototype robot in order to model its complex dynamics. As explained by the authors, their approach is to use a simplified model to train multiple deep neural networks (DNN) to represent trends in state dynamics while also reducing errors in prediction. To accomplish this, they initially trained a DNN to learn a simplified model of general trends. Then they trained a second DNN to learn the complex physics of the simulation to map to reality. Furthermore, DNNs are able to learn the general form of the dynamics from simulation with much less hardware and

training data. This leads to increased accuracy by roughly 52%, while also decreasing the amount of training time and data needed. The researchers in [23] used a similar approach as they used a convolutional neural network; to analyze the bone marrow stem cells in rats. While it did not directly map to movement, they were able to effectively investigate the cell activity within systems at roughly 94% accuracy. The authors in [21] took a different approach as they used a combination of deep learning frameworks. In related studies, supervised deep learning models are mainly employed. Given that these algorithms are generally employed for classification, they can be used to distinguish different objects in contact. For the calibration of current soft sensors, a recurrent neural network (RNN), a deep learning algorithm specialized in time- series data, is frequently used. For sensors that have complex data types, such as skin, a convolutional neural network (CNN), an effective deep learning algorithm for image processing, has been used for tasks such as the classification of objects or forecasting. Overall, deep learning has shown a great ability to model complicated tasks with great accuracy and flexibility, thus providing an outlook into the capabilities of showing efficiency for soft-robotic tasks.

2.3 Reservoir Computing

Reservoir computing is a deep learning technique that uses a fixed and randomly initialized “reservoir” of neurons to process data. The reservoir is typically a randomly connected network of recurrent processing units, such as recurrent neural networks. These networks can be trained to model highly

complex dynamics. In the general framework of reservoir computing, input signals are transformed through the reservoir, then dynamics of the reservoir map the inputs into internal states. These states can then be transformed by a trainable readout layer to produce an output.

Reservoir computing has been shown to be used in soft robotics to control the behavior of the robot's actions. Soft-bodied robots are challenging to control because they can exhibit complex, nonlinear dynamics. Reservoir computing provides a way to quickly and accurately learn the dynamics of the robot and generate control signals that can produce desired behaviors. Studies such as [24][25] observe the navigation behaviors for mobile robots in simple and complex environments. The RC model provides an efficient way to train recurrent neural networks by letting the reservoir be fixed while only a linear readout output layer is trained. This allows for training to be so simple it enables the reservoir to emulate multiple nonlinear dynamics, perform pattern generation, and modulate outputs.

Studies such as [17][26] explain how the basic concept of RC is exploiting the dynamics of the reservoir by learning, which requires some parameter tuning, to the readout part. In this architecture, reservoirs do not have to be an RNN anymore but can be any dynamical system. This idea naturally leads us to exploit the dynamics as a reservoir instead of using the simulated dynamics inside the system. One example of the use of reservoir computing in soft robotics is the control of a soft robotic arm [27]. The reservoir network was trained to learn the dynamics of the arm as it grasped onto different objects of multiple sizes. The

learned dynamics were then used to generate control signals for the robotic arm that could adjust the grip strength and adapt to the object. This adaptation shows the potential use for RC within optimal and accurate robotic control algorithms.

Overall, reservoir computing provides a powerful tool for controlling soft robots. Using this method to effectively model and control a robotic structure could have a significant impact on the future of robotics and technology.

2.3.1 Physical Reservoir Computing. It has been studied how the reservoir does not have to be a digital system, but instead, the physical body does the sensing and acts as the reservoir. This notion of the physical body acting as the reservoir is observed as Physical Reservoir Computing (PRC). The reservoir can now mimic all kinds of animal motions, like octopus locomotion, insect flying, and fish swimming. These robots share many similarities with animals regarding their structure and kinematics [12]. However, their underlying actuation and control kinematics may be different. These sensing, actuation, and control features are all analyzed through PRC, wherein a physical body performs the computations. The researchers in [12] were able to utilize an origami structure as the physical reservoir body. They proved that modeling and a combination of compressions, extensions, and folds can generate the patterns for autonomous locomotion gait generation. In works such as [5][28], a reservoir model of a spring mass network was numerically evaluated to model the link between the property of the mechanical reservoir and its computational ability in learning. By replacing point masses in a mass–spring network with stiff bars, a tensegrity-like structure is

obtained. Tensegrity [29] (tension integrity) indicates a structural framework that uses isolated compression elements loaded in a continuous network of tension elements, such as springs and masses, leading to a physical structure combining strength and flexibility. Other works [30][31] explain how animals like structures such as the octopus and fish have been one of the major sources of inspiration for scientists for many years. The ability to use PRC to model and control the morphology of a physical object or robot can also show the usefulness of RC in more robust tasks.

2.4 Digital Twin

As explained in Section 1.4, digital twins can be used for soft robotics for various benefits.

- **Simulation and Design:** soft-bodied robots can have complex structures that can be difficult to model. Digital twins can help by creating a virtual model of the soft robot that can be used for simulations and design suggestions. This allows for testing different configurations of a proposed robot without the need for physical prototypes.
- **Control and Optimization:** soft robots can be difficult to control due to their highly-nonlinear interactions and movement. Digital twins can help by creating a virtual model of the robot and its environment, which can be used to optimize control algorithms and test different control strategies. This can help for improving the robot to achieve optimal distance and power.
- **Predictive Maintenance:** soft-bodied robots are often subject to damage due to

their potential terrain or material design. Because of this, digital twins have been used to monitor a robot's performance and predict when maintenance may be needed given certain forces. This feature can help relieve potential damage to the robot itself and the environment.

As explained in [32], a digital twin is essentially a virtual model designed to accurately reflect an actual, physical object. Digital twins utilize real-time data to create models or simulations to predict a process. In many studies, digital twins have been used to model various objects from large-scale wind turbines to cells in a system. The authors in [3] investigated epidemiological models to help with the more recent Covid-19 epidemic. They suggested using a Bi-Directional LSTM and a Generative Adversarial Network (GAN) based digital twin to predict and gain a better understanding of how diseases spread. In the results, these two network models performed accurately in comparison to previous predictive disease models and also led to faster execution time. This study helps prove how certain structures of neural networks can be beneficial for digital twin modeling. This digital twin concept has also been validated in [33] as it has been utilized to simulate more industrial environments, thus showing the ability to be robust in use. In this study, physics-informed neural networks were used to generate predictive models for fatigue crack growth prognosis on aircraft window panels. Trained over large quantities of historical airplane data and real-time data, the model showed flexibility to learn and predict the dynamics of the cracks. For soft-bodied robotics, knowing the dynamics and potential stresses of a system can have an impact on its

modeling and development.

Overall, digital twins can help accelerate the development and deployment of soft robots by reducing the need for prototyping and testing a physical robot, while also improving performance and reliability.

2.5 Robotic Control

Robotic control optimization involves finding the best ways to control and manipulate robots to perform specific tasks efficiently and effectively. There are many factors to consider when optimizing robotic control, such as the robot's physical capabilities, the environment it operates in, and the specific task it needs to perform.

One common approach to robotic control optimization is through the use of machine learning and deep learning techniques. For example, reinforcement learning algorithms can be used to train robots to perform certain tasks through trial and error, adjusting their behavior based on feedback from their environment. Studies have shown [34][35] how neural networks and reinforcement learning can be utilized for the grasping of objects and general manipulation of a robot. In these studies, it was shown that the final control policy was essentially a superposition of control signals from the robot. The robot learns to select the appropriate grip and force to pick up objects by receiving rewards for successful grasps and penalties for failed attempts.

Another aspect of robotic control optimization is the use of sensors and actuators. By choosing the right sensors and actuators, improvements can be

made to a robot's performance, speed, and efficiency. Studied in [36], a robot's sensor configuration determines how it perceives its environment, such as cameras, lidars, or force sensors. Different sensors provide different types of information, such as visual or tactile feedback, and have different noise and resolution characteristics. The control algorithm must be designed to process the sensor data and extract relevant features for decision-making promptly.

Finally, optimization can also be achieved through the intentional design of the robot's structure and control system. This can involve optimizing the robot's kinematics and dynamics, as well as developing control algorithms that take into account the robot's physical properties and the action to be performed. Since robots can be designed and configured in many different ways, there is no true optimization pattern for a robot. Shown in [37], more complex robots can perform many complex tasks but they also come with a more complex optimization algorithm. Overall, robotic control optimization is a complex problem that involves expertise in robotics, control theory, and machine learning.

CHAPTER 3

METHODOLOGY

For the purposes of robotics, artificial intelligence (AI), has been defined as the ability of a computer or computer-controlled robot to perform tasks with the ability to learn, commonly associated with intelligent beings [37]. This field of AI has a subset of fields which are machine learning (ML) and deep learning (DL). It can ingest unstructured data in its raw form such as images or text, and it can automatically determine the features which distinguish different categories of data from one another [38]. Unlike basic machine learning, it does not require human intervention to process data, allowing us to scale machine learning in larger or smaller fashions as needed. For the future of soft-bodied robotics, there have to be robust, flexible, and cost-efficient methods to ensure optimal control and movement.

In the field of deep learning, there are a variety of techniques and methods used to train models. Some of the deep learning techniques used for this application were supervised learning and recurrent neural networks. Supervised learning can be powerful in deep learning as it can leverage labeled datasets, also known as supervised learning, to train its algorithm, but it does not necessarily require a completely labeled dataset. In addition, recurrent neural networks have been known to be good for time-series forecasting, as the model is able to learn from previous internal states. These methods have been great for deep learning practices, so investigating these approaches are necessary to see if they are feasible for this application.

In general, this research sought to provide a deep learning approach to modeling soft-bodied robotics using reservoir computing. The physical system acts as the body to be studied which can be an efficient and cost effective way to utilize reservoir computing. In practice, this would allow for the investigation of these tactics for soft-bodied robotics.

3.1 Methods in Deep Learning

Deep learning is a subfield of machine learning that uses artificial neural networks (ANNs) with multiple layers to model and solve complex problems. It is called “deep” because it involves the use of neural networks with multiple hidden layers that can learn increasingly abstract representations of the input data. In contrast to traditional machine learning, which relies on hand-crafted features and rule-based algorithms, deep learning can automatically learn useful features and patterns from raw data. The basic idea behind deep learning models is to simulate the structure and function of the human brain’s neural network, which is composed of interconnected neurons that process and transmit information. Soft robots have several advantages over traditional robots, such as improved flexibility and enhanced mobility for various terrains. The idea for these types of robotic structures is that they can perform a wide range of tasks with precision. To perform these tasks, some form of self-learning has to be done to ensure mobility and adaptability for different environments. Deep learning methods allow for these robots to have a form of self-learning.

3.1.1 Supervised Learning. Supervised learning is a type of machine learning algorithm in which the computer is trained on a labeled dataset. In supervised learning, the computer is given a set of input data along with the correct output or label for each example. The goal of the algorithm is to learn a function that maps the input data to the correct output so that it can make accurate predictions for new, unseen data [39]. Supervised learning can be used in a variety of applications, such as image recognition, speech recognition, natural language processing, and prediction [40]. For example, in a supervised learning algorithm designed to identify digits, the algorithm would be given a dataset of digit images, along with their corresponding labels (the actual digit represented by the image). The algorithm would then learn to recognize the patterns in the images that are associated with each digit, and use this information to classify new, unseen images. It is a widely used approach in machine learning and is typically used when the task is well-defined and the labeled data is available.

However, the quality of the labeled data can significantly impact the accuracy of the algorithm, and labeling large datasets can be a time-consuming and expensive process. When analyzing robots, I can accurately map inputs to outputs given this method. As certain tasks may be more difficult to accomplish or the robotic structure itself is more complicated, this method serves well for prediction accuracy due to its ability to directly map inputs to outputs. A supervised learning algorithm can be simply written as equation (3.1) where $Y = (y_1, y_2, y_3, \dots, y_n)$ is the prediction and $X = (x_1, x_2, x_3, \dots, x_n)$

is the input vector. Supervised learning is generally divided into two sub-categories, classification and regression, to be discussed below.

$$Y = f(X)$$

Eq. (3.1)

3.1.2 Classification. Classification is a type of supervised learning task in machine learning where the goal is to predict the class or category of a given input. The input data is typically a set of features or attributes that describe a particular object or entity, and the output is a label that specifies the class or category to which the object or entity belongs. There are many types of classification algorithms, including decision trees, support vector machines (SVMs), and neural networks [41]. These algorithms use different techniques to learn the mapping from input features to output labels, and usually, the choice of algorithm depends on the specific task and attributes of the data. For example, a classification algorithm might be trained on a dataset of images of animals with complex locomotion features, such as an octopus or bird, with each image labeled respectively. The algorithm would learn to recognize the patterns in the images that are associated with each label, and use this information to classify new, unseen images as either an “octopus” or “bird.”- However, for these tasks, I did not focus on the class of the object, but rather on its previous and next actions. To perform this task, I utilized another supervised learning strategy, a regression algorithm.

3.1.3 Regression. Regression is a supervised machine learning algorithm that is used for predicting a continuous output variable based on one or more input variables. The goal of regression is to learn a function that can accurately map the input variables to the output variable [42]. The training process for a

regression model involves feeding the algorithm a dataset with known input and output values, and the algorithm learns to adjust its parameters to minimize the difference between the predicted output and the actual output in the training set. When the model has been trained, it can be used to make predictions on new data. The relationship between the input and outputs is represented by a mathematical formula, called a regression model, which is typically a linear or nonlinear function. Throughout the process of the study, I utilized several popular regression models such as linear, ridge, and lasso to investigate how the model performs with each algorithm.

3.2 Deep Learning Models

It has been proven how various deep learning methods can be used to develop controllers for soft-bodied robots that can operate in dynamic situations. For example, deep neural networks (DNNs) can be used to process sensor data from soft robots and identify the shape and location of objects, detect obstacles, and recognize gestures. In addition, Recurrent neural networks (RNNs) can be used to model the behavior of soft robots over time. RNNs have been used to predict the movement of soft robots and to control their motion in different environments. These are a only few methods that can be utilized in modeling dynamic systems.

The main advantages of these deep learning principles are their ability to automatically learn and represent hierarchical features from raw data. These models have been proven to be particularly useful for tasks that involve high-

dimensional data. Shown in recent works [2], various neural networks have been utilized for many complex tasks such as robotics. Two of the more popular neural networks are deep neural networks (DNN) and recurrent neural networks (RNN). These two models show potential for the future of soft robotics due to their ability to learn abstract information and provide accurate representations of complex architectures.

3.2.1 Deep Neural Networks. The main purpose of a deep neural network is to receive a set of inputs, perform progressively complex calculations on them, and give output to solve real-world problems. A deep neural network (DNN), typically consists of multiple hidden layers that are fully connected [2][43] as shown in Figure 3.1. Each hidden layer has several nodes which are termed the hidden units. For this study, I used two different DNN architectures compare with other deep learning models. The first architecture contained four dense layers with 256 nodes each except for the output dense layer which had 201 nodes corresponding to the maximum input dimension of the various structures modeled. The first three layers used the Rectified Linear Unit (ReLU) activation functions while the output layer had no activation function. For the second architecture, to provide a deeper

model while improving performance, I utilized six hidden layers. This would make the model more adaptable and robust. These models were developed using the Keras framework.

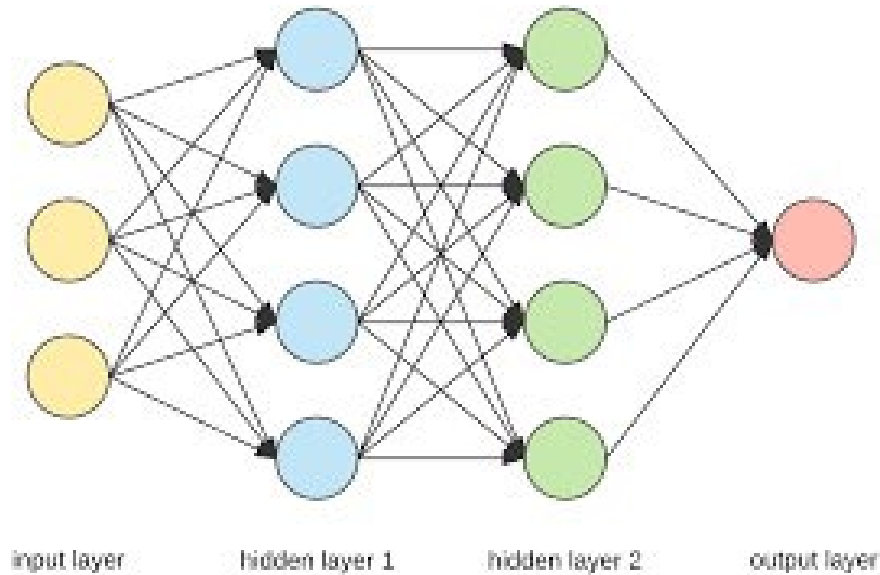


Fig. 3.1. Architecture of a Deep Neural Network [2]

3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of artificial neural network that is specialized for processing sequential data, such as time series or natural language. RNNs are designed to handle varied input sequences by maintaining a memory of past inputs and using this memory to influence the processing of future inputs. The key feature of RNNs is that they have loops in their architecture, which allow information to be passed from one time step to the next. Specifically, RNNs have a set of hidden nodes that are connected to both the input nodes and

the output nodes. During the processing of each time step, the hidden nodes receive both the current input and the output of the previous time step.

The main advantage of RNNs is their ability to capture the temporal dynamics of sequential data [44]. Unlike traditional feedforward neural networks, which process each input independently and in isolation, RNNs can use information from previous time steps to influence the processing of future inputs. This makes RNNs particularly useful for tasks such as speech recognition, language modeling, and time series forecasting, where the current input was influenced by past inputs. For soft robots, their locomotion is proportional to time. As such, most robotic simulation data can be sequential. Due to this property of the data, RNNs can provide great performance in modeling complex, time series data.

There are several variations of RNNs that have been developed to address specific challenges in sequential data processing. One common type of RNN is the Long Short-Term Memory (LSTM) network, which is designed to address the vanishing gradient problem [45] that can occur in standard RNNs. The vanishing gradient problem arises when the gradients that are propagated through the network during backpropagation become very small, making it difficult to train the network effectively. Another variation of RNNs is the Echo State Network (ESN), which is similar to the LSTM in terms of being beneficial for sequential data but has a different architecture. ESNs provide an architecture and supervised learning principle for RNNs.

3.3.1 Long Short Term Memory (LSTM). Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture. LSTMs were introduced to address the issue of vanishing gradients in traditional RNNs, which made it difficult for them to effectively capture long-term dependencies in sequential data. Essentially, an LSTM is made up of cells that can store information, and gates that control the flow of information. During training, the LSTM network learns to adjust the parameters of these gates to decide how much information should be stored or discarded at each step, in order to optimize the network's performance on a given task [3].

For the experiments, I investigated two different LSTM architectures for comparison against other deep learning models. The first architecture was made up of two LSTM layers with 320 memory cells each and a dense layer as the output layer. To provide a deeper layered approach, the second architecture used six LSTM layers and one dense layer as the output layer. The dense layer had 201 nodes corresponding to the maximum input dimension of the various structures modeled. The architecture of an LSTM unit is as shown in Fig. 3.2.

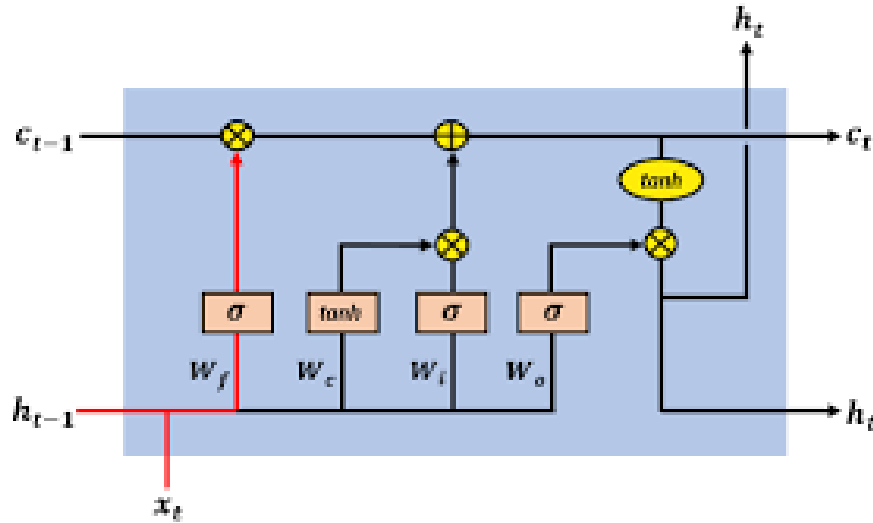


Fig. 3.2. Architecture of an LSTM [3]

3.3.2 Echo State Network. An Echo State Network is a type of RNN, part of the reservoir computing framework, explained in Section 4.5.1. The ESN was proposed as a simplified alternative to more complex RNNs such as long short-term memory (LSTM) networks. ESNs have an architecture of nonlinear processing elements that are densely interconnected and recurrent, forming a “reservoir” that stores information about the history of input and output patterns. The outputs of these internal processing elements are referred to as the “echo states”. The key idea behind an ESN is that the echo state can be used to “echo” the input sequence, allowing the network to learn a complex mapping from the input sequence to the output sequence. During training, only the output weights of the network are trained, while the fixed internal state is left unchanged. This makes the training process much faster and easier than in traditional RNNs.

One of the advantages of ESNs is that they require less training data and have fewer parameters to tune than other types of RNNs, which can make them easier to use in practice. The main idea is to introduce a random, fixed recurrent neural network with the input signal, and combine a desired output signal by a trainable linear combination of all of the signals. ESNs have been shown to be effective for tasks such as speech recognition and image captioning [46]. As mentioned, this framework derives from the reservoir computing framework which has proven to be useful on time-series data through its ability to learn from its previous internal state. Due to this nature, the ESN can be used to accurately map signals of a robotic structure with less training time and training loss.

3.3.2.1 *Deep Echo State Network.* Another method studied in this study was the idea of a Deep Echo State Network (DeepESN). This framework used the RC framework, but instead used stacked reservoirs that feed into each other successively, shown in Fig. 3.3. By using this technique, the model can provide a deeper level of feature extraction. It has been shown in previous research how this model provides higher accuracy on more complex tasks than standard RC [4]. For this experiment, I used the same parameters of the standard RC and varied the number of layers to investigate the effects of this method.

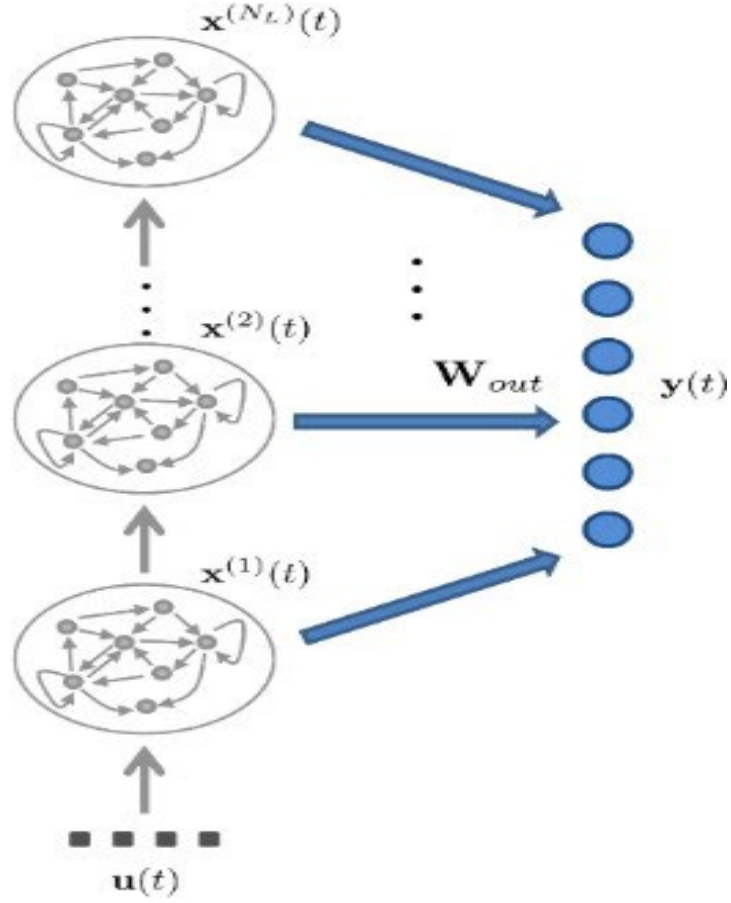


Fig. 3.3. Architecture of a Deep Echo State Network with Successive Reservoirs [4]

3.4 Reservoir Computing

The underlying method for this study stemmed from the concept of Reservoir Computing (RC). The reservoir is treated as a black box from which a simple readout mechanism is trained to read the state of the reservoir and map it to the desired output [47]. Overall, the RC model, shown in Fig. 3.4, encompasses three important parts: input layer, dynamic reservoir, and readout layer. These elements are explained further in upcoming sections.

To summarize, the input layer ingests data into the reservoir. The readout layer is used to analyze the state of the system. The readout layer is usually a simple linear regression. What makes this framework so powerful is that it uses fixed weights in most of the layers. In this entire framework, only the readout layer weights are trained. This combination leads to faster training and lower training costs. This framework has been utilized for various complex time series tasks with decent prediction accuracy [6][5]. Overall, these characteristics make RC easy and quick to train, thus proving to be a viable method for robotic modeling and control.

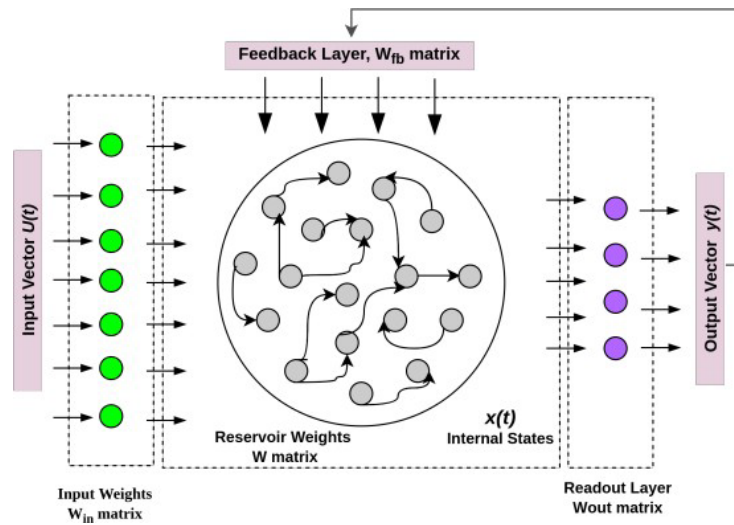


Fig. 3.4. A typical Reservoir Computing Architecture [5]

3.4.1 Architecture. The diagram shown in Fig. 3.4, depicts the architecture of RC. This model has three main sections: the input layer, reservoir, and readout layer. Between these layers are a system of nonlinear memory cells called weights. These weights are all fixed except at the readout layer. Another important part of the architecture is the RC model's ability to have internal states that can be learned from. This internal state can be modeled in Equation (3.2), where $x(t-1)$ is the state vector of the reservoir units, $u(t)$ is the input vector, W_{in} is the weight matrix for the input-reservoir connections, W is the weight matrix for the recurrent connections in the reservoir, W_{fb} is the feedback weight matrix and f is a hyperbolic tangent(tanh) activation function.

$$x(t) = f(W_{in} \cdot u(t) + W \cdot x(t-1) + W_{fb} \cdot y(t-1))$$

Eq. (3.2)

3.4.1.1 Input Layer. Overall, the function of the input layer is to process data into the reservoir with fixed random coefficients, W_{in} . In the function above, the input layer can be modeled by the combination of the input signal and input weight matrix, $W_{in} \cdot u(t)$. Throughout this study, the weight matrix was varied with different initialization approaches such as random and Xavier initialization.

3.4.1.2 The Reservoir. What makes the RC framework unique from others is the use of a fixed set of random, nonlinear nodes, the reservoir. The reservoir

is typically a recurrent neural network (RNN) with a large number of neurons, but other types of dynamical systems such as echo state networks (ESN) and liquid state machines (LSM) can also be used as reservoirs. The reservoir is called fixed because its connections and parameters are generated randomly and are not modified during the learning process. The input data is fed into the reservoir, and the reservoir's dynamics transform the input into a higher-dimensional representation, which is then fed into a linear readout layer to produce the output. Studies have been conducted and show how the number of recurrent units, the size of the reservoir, affect the performance in predictions [6][48]. For this study, I varied the size of the reservoir, ranging from 50 to 250 recurrent units, to encompass the different input dimensions for the different datasets.

3.4.1.3 Readout Layer. The output from the readout layer is also computed using the simple output layer equation as shown in equation (3.3) where f_Y is the output nonlinear activation function [49].

$$y(t) = f_Y(W_{out} \cdot x(t))$$

Eq. (3.3)

The task of training the readout is then reduced to a simple linear regression problem of minimizing the squared error. The regression model minimizes the mean square error between predictions, Y and the ground truth, \hat{Y} i.e. $|Y - \hat{Y}|^2$

3.4.2 Echo State Property. ESNs have an architecture of nonlinear processing units that are interconnected and recurrent, forming a “reservoir” that stores information about the mapping between inputs and outputs. The outputs of

these internal states are referred to as the “echo states.” The titles of the echo states stem from the input values echoing throughout the reservoir’s states due to the reservoir’s recurrent nature [16][22]. These echo states are fed into an, usually linear, readout network, which generates the network output. The performance of the ESN depends on the design of the RNN-based reservoir. It was observed [46] that the echo state property was obtained for any input if the spectral radius (i.e. the maximum absolute eigenvalue of W) was adjusted to be less than one.

3.4.3 Reservoir Computing Model. The reservoir computing model utilized for analyzing soft robotics is shown in Fig. 3.5. It provides an outlook of reservoir computing models that are modeled for dynamic systems. The input from the system, $u(t)$, feeds into a fixed input weight matrix, W_{in} , which feeds into the internal reservoir. The internal states are then trained at the output layer weight matrix, W_{out} . With an RC model for soft robotics, one can potentially accurately model many dynamic systems due to this architecture. In this study, I varied different parameters of the RC model such as the size of the reservoir, which correlates to the reservoir weight matrix. I also modified the input dimension corresponding with the appropriate number of nodes in the structure, which varied the input weight matrix as well.

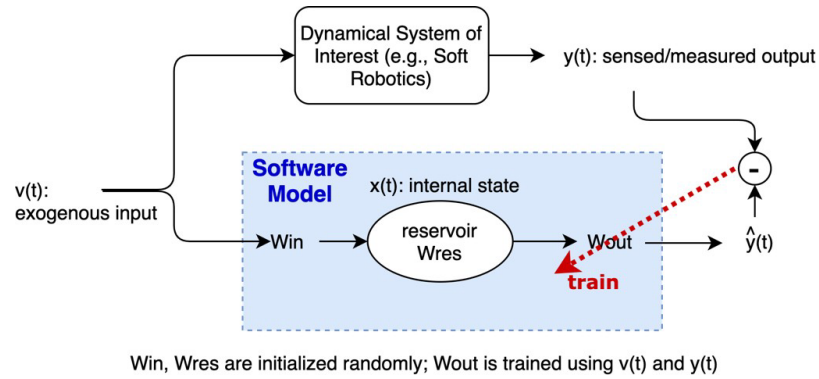


Fig. 3.5. Depiction of a Software Reservoir Computing Model for Soft-Robotic Structures

3.4.4 Physical Reservoir Computing. In some cases, mechanical systems, such as soft and compliant robots, are possible options for physical reservoirs. Soft and compliant robots with flexible bodies are difficult to control due to their complex body dynamics compared with rigid robots with stiff bodies. However, such complex behavior can be favorably leveraged to generate rich nonlinear dynamics required for RC. The idea of outsourcing computation to a physical body is also known as morphological computing in the field of robotics. Other types of physical bodies have been studied such as the limbs of an octopus and birds [18]. This study utilized this concept of reservoir computing to investigate the potential uses of physical reservoir computing and its performance on modeling soft robotics, shown in Fig. 3.6. To achieve this, I used the frameworks of reservoir computing within the digital space, while also using the principles of physical reservoir computing. The physical body studied for this was spring-mass systems, as explained in Section 3.5. I chose this due to their ability to model many complex,

dynamic structures, while also being easier to model in a digital twin as it provides nodes and springs to be observed.

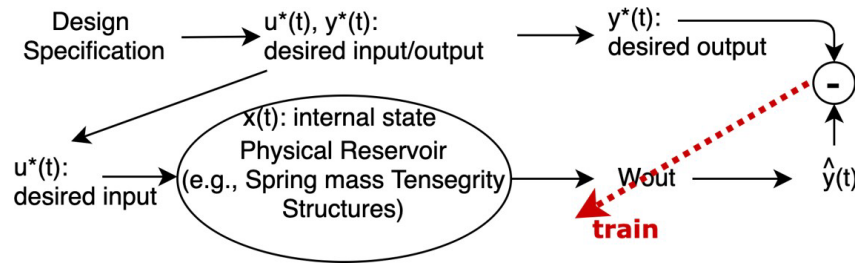


Fig. 3.6. Flow of a Physical Reservoir Computing Model

3.5 Mass-Spring Networks

For the digital twin concept, the body chosen to simulate was utilized with a Mass-Spring network. A Mass-Spring network is a mathematical model that is used to simulate the behavior of objects that are connected by a network of springs and masses, shown in Fig. 3.7. The basic idea is that the springs represent the connections between the masses, and the masses represent the objects themselves. In a Mass-Spring network, the behavior of each mass is determined by the forces exerted on it by the springs that are connected to it. These forces are calculated based on the displacement of the spring from its resting position, as well as the properties of the spring, such as its stiffness and damping.

In physics simulations, they are used to model the behavior of complex physical systems such as fluids, gasses, or solids. In engineering, they are used to model the behavior of structures such as buildings, bridges, or machines. For this

application, I combined physics simulation with engineering, providing a more realistic model of the system. Mass-spring networks were chosen as the body of choice because they were easy to implement with geometry and physics equations. Shown in equation (3.4), these systems can be modeled through formulas for each spring and node. Each node, N_i , is represented by its mass m_i , whereas the passive parameters for each connection are the spring stiffness k_j and the damper coefficients d_j for each spring. The default values used in the experiments are $C = 3$, $m_i = 1$ kg, $k_j = 100$ N/m, and $d_j = 10$ Ns/m

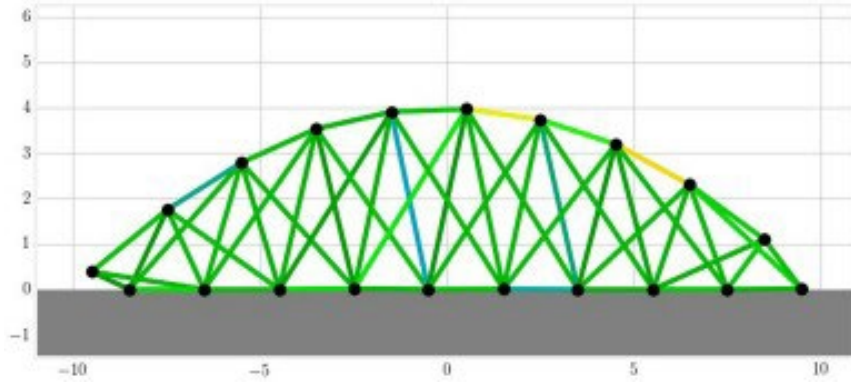


Fig. 3.7. An example of a 20 node structure with attached springs [4]

$$Number of Springs(S) = (N - 1 - (\frac{\frac{C}{2} - 1}{2})) \times \frac{C}{2}$$

Eq. (3.4)

3.5.1 Optimization. For the optimization of mass-spring network performance, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) has been utilized as the best optimization method. In certain studies, it has been described as one of the most promising methods for optimization [50], where control functions cannot be described using an explicit representation. CMA-ES samples solutions from a multivariate gaussian distribution. After evaluating all solutions, the solutions are sorted by evaluation values, then updating the distribution parameters based on the ranking of evaluation values.

For the simulation, locomotion traits were performed and evaluated through two performance metrics, distance, and power. It performed the optimization using the CMA-ES as it produces a decent convergence speed and requires few initialization parameters. In the CMA-ES algorithm, the number of iterations is tuned specifically for each optimization to ensure convergence, since optimizing small structures will converge faster than larger ones [28]. From each trial run, the best individual is retained as the optimized model structure. Typically, the optimized parameters, θ , of the MSD network are the controller amplitude a_j between 0 and 0.25, its frequency between 0 and 10 Hz, its phase ϕ_j between 0 and 2π , and the spring stiffness k_j between 0 and 100 N/m. To synchronize the actuators together and impose the fundamental frequency, the angular speeds ω_j are fixed to the same value.

3.6 Digital Twin Reservoir Computing Model

A digital twin represents a physical system that is created using digital

technology such as sensors, simulation software, and machine learning algorithms. Essentially, a digital twin is a digital replication of a physical system that is capable of simulating its behavior and performance. It ingests data and replicates processes so one can predict possible performance outcomes and issues that the real-world product might undergo. It can be utilized to model various systems ranging in complexity [51]. This concept is used to predict the outcomes of physical and digital systems by attempting to create a digital representation of the system. In efforts to further the research into soft-bodied robotics, in running the locomotion simulation, this work aimed to introduce an approach of reservoir computing for the implementation of a digital twin.

To achieve this, I utilized a simulator [28] that simulates various structures of spring-mass systems. This design and its locomotion were optimized with the CMA-ES optimization method explained in Section 3.5.1. This returns different optimal parameters such as its spring lengths and stiffnesses. With these optimal parameters and simulation, I used these as inputs to the digital twin model. From there, these inputs were transformed through the RC model to provide an output. This output was measured against the simulation output. In the end, this allowed us to measure the accuracy of modeling spring-mass systems with digital twins with an RC framework, shown in Fig. 3.8. This can provide progress in the field of soft-bodied robotics as I can more accurately study its dynamics for many structures.

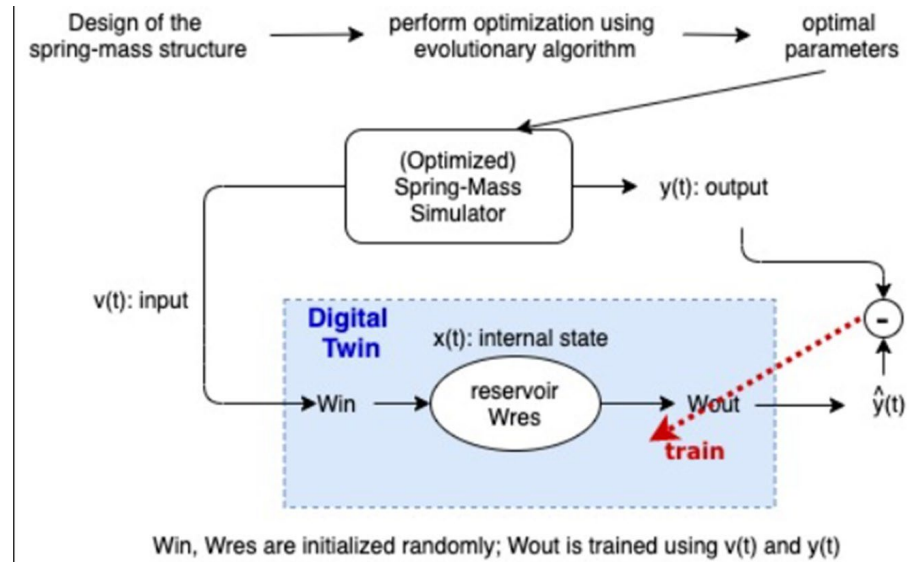


Fig. 3.8. Flow of a Digital Reservoir Computing Model for Digital Twin Modeling

3.6.1 Optimal Control. Soft robotic optimal control is the application of optimal control theory to soft robots, to find the best control strategies for these highly compliant systems. The main challenge in soft robotic optimal control is the highly nonlinear and complex nature of the system dynamics, which can make it difficult to derive analytical solutions.

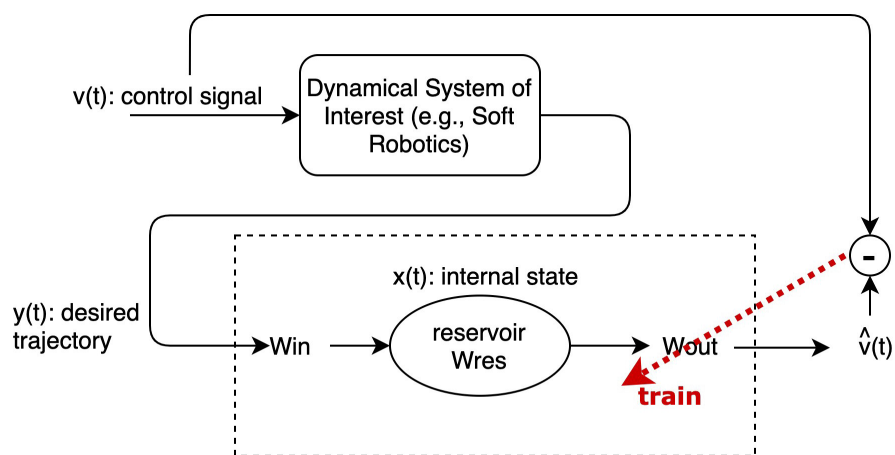
To optimize motion within robotic structures, I utilized reservoir computing to map inputs to potential outputs. As explained in Section 3.5, I utilized a mass-spring network simulator of a locomotion robotic structure. With optimal parameters moving a structure through time, I investigated mapping the trajectory of the robot to optimal spring parameters. To run the experiments and analysis, I utilized an MSD network simulator [28]. This simulator directly implemented mechanical equations using Python and Numpy to manipulate the simulated

structure. These networks [1][52] consisted of a set of nodes with mass, connected by springs which were all separately actuated.

$$L_{j,o} = l_{j,o} \cdot (1 + a_j \cdot \sin(\omega_j \cdot t + \Phi_j))$$

Eq. (3.5)

To actuate the spring using a control signal, I modulated the reference lengths of the springs $l_{j,o}$. In the simplest and default case, this was represented by a simple sinusoidal signal shown in equation (3.5). This input signal allowed for a set of tunable parameters $l_{j,o}$, ω_j , a_j for each spring in the simulation. All springs had an optimal base Φ_j . These parameters were implemented to obtain optimal spring lengths using equation (3.5). Shown in Fig. 3.9, these spring lengths were used as the labeled data. From this, I used the dynamic movement of the robot as input to the model to predict the optimal spring control signals, thus providing for optimal robotic control.



W_{in} , W_{res} are initialized randomly; W_{out} is trained using $v(t)$ and $y(t)$

Fig. 3.9. Flow of a Reservoir Computing Model for Optimal Control in Digital Twin Modeling

CHAPTER 4

EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

In this section, I provide and explain the two experimental set-ups: Dynamics and Control. In such experiments, the parameters of the simulation and the proposed RC model settings are detailed in Section 4.2. In a following Section 4.5, I examined the results of modeling the Dynamics of the structure and compare the performance of the ESN against various methods and parameters. In Section 4.6, I examined potential performance enhancements to the ESN. This section will over- all provide an overview of how RC performs across different metrics and schemes. To provide a deeper understanding of robotic control, I display the observed results from the Control experiment, provided in Section 4.7.

4.1 Data Generation

To generate data for the experiments, I utilized the MSD simulator explained in [28]. The simulation runs to achieve optimized parameters for the structure. In doing this, I was able to receive a 2-D simulation of a moving structure with optimized spring parameters, thus providing a set of trajectories of the simulated robot. The simulation was executed over 100,000 time steps, which was mitigated to 400 seconds of simulation time. Each time step provided a location and velocity of each node, which also allowed for an overall acceleration of each node over all of the time steps. These location, speed and acceleration parameters were used as the inputs for the RC, shown in Fig. 3.5.

The simulations provided for each of the simulations to be performed over a different number of nodes: 5, 10, 20, and 50, yet most other variables such as the number of springs and spring stiffness all remained the same. To explain the datasets used: Dataset 1 corresponds to the structure with 5 nodes, Dataset 2 corresponds to the structure with 10 nodes, Dataset 3 corresponds to the structure with 20 nodes, and Dataset 4 corresponds to the structure with 50 nodes, shown in Table 4.1. This was done in order to truly understand how robust the RC is given simple to more complex structures.

Table 4.1. DYNAMICS SIMULATION TRAINING DATA DESCRIPTION

Category	Number of Nodes	Size
Data 1	5	$100,000 \times 21$
Data 2	10	$100,000 \times 41$
Data 3	20	$100,000 \times 81$
Data 4	50	$100,000 \times 201$

For optimal robotic control, this simulation used the CMA-ES optimization strategy to provide optimal spring parameters for the best movement and performance. In this simulation, various optimal parameters for each spring were provided after each iteration. These parameters were mapped into a control signal for each spring, previously shown in Equation 3.5, which served as output to investigate methods of accurately modeling a structure's movement trajectory to

optimal spring lengths. In the control experiments carried out, the input to the model was the dynamics of the structure: position, velocity and acceleration of the nodes. The target outputs were the optimal control signal for each spring in the structure. In doing the experiments, I simulated 5, 10, 20 and 50 node structures with 9, 24, 54, and 144 springs, respectively. The label data dimensions are shown in Table 4.2.

Table 4.2. CONTROL SIMULATION TRAINING DATA DESCRIPTION

Category	Number of Nodes	Number of Springs	Size
Data 1	5	9	100,000×9
Data 2	10	24	100,000×24
Data 3	20	54	100,000×54
Data 4	50	114	100,000×114

4.2 Experiment Set-Up

4.2.1 Dynamics Simulation Set-Up. For the purposes of modeling soft robotics, I utilized the position, speed and acceleration of each node of the structure to accurately create a digital twin. Mapping these characteristics was important because it allowed me to model how the structure would move over time. In doing this, I could accurately observe the evolution of each node over time. From these plots, I was able see how the dynamics of each node were affected given various forces, thus making accurately modeling each node a challenge. The images shown in Fig.

4.2 and Fig. 4.3 represent nodes 1 and 11 in the simulated structure, shown in Fig. 4.1, providing a look into how each node trajectory is unique. Observed in the plots, each of the nodes evolved differently over time, causing such complexity. A more detailed view of the x-position of the mass is provided in Figure 4.4, showing how the structure did not have a constant pattern of motion. All of the characteristics of the moving masses make digital twin modeling a complicated task to solve.

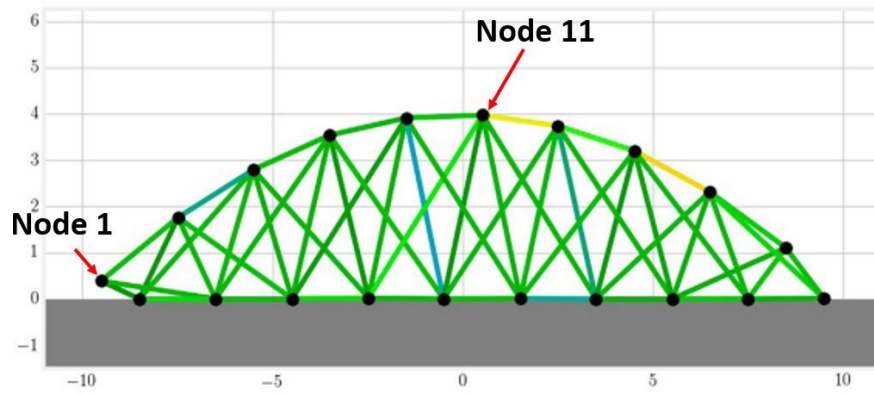
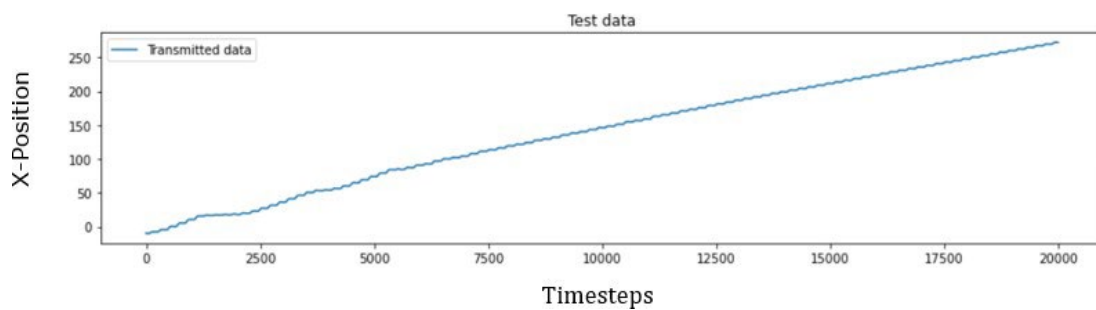
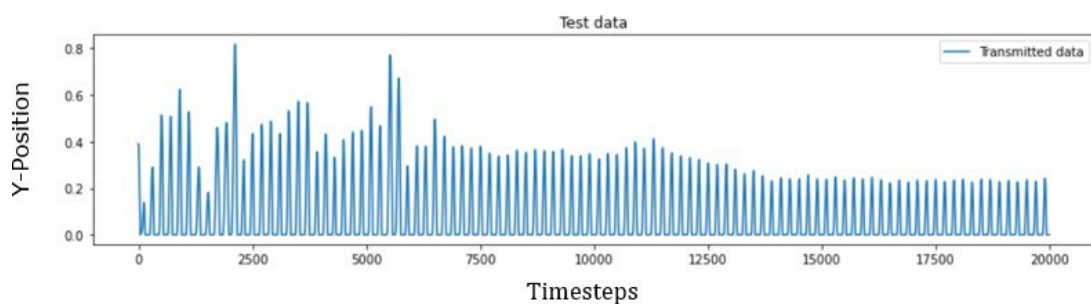


Fig. 4.1. 20 node structure with modeled nodes derived from [4]

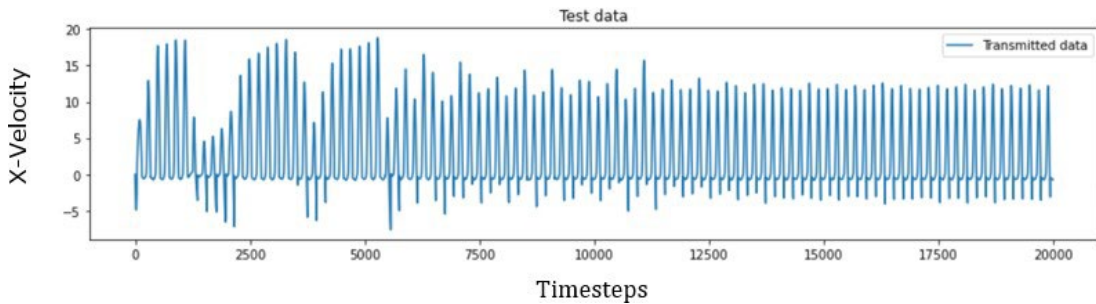
The simulated structures can be perceived as a reservoir because of its dynamics and high complexity. For each time step t_k , the system's current state is evaluated using the input vectors $[x_1, x_2..y_1, y_2..vx_1, vx_2...vy_1, vy_1...a]$, $a[k - 1]$, which comprise both X and Y components of all the nodes. Through this, I examined not only the acceleration, but also the position and speed of each node, thus creating a digital twin for a soft robotic structure.



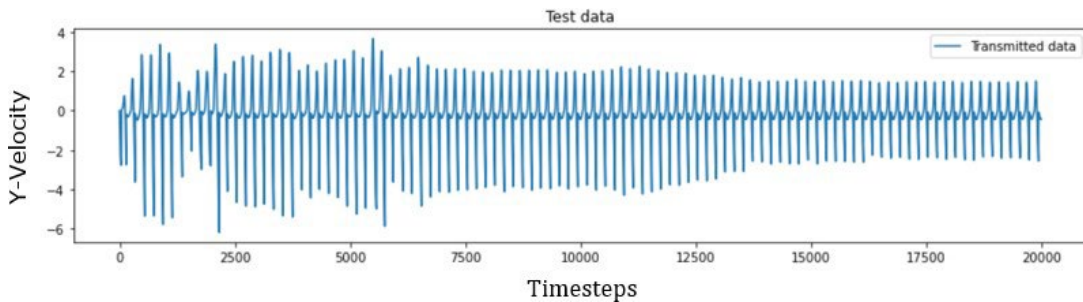
(a) Node 1 X-Position



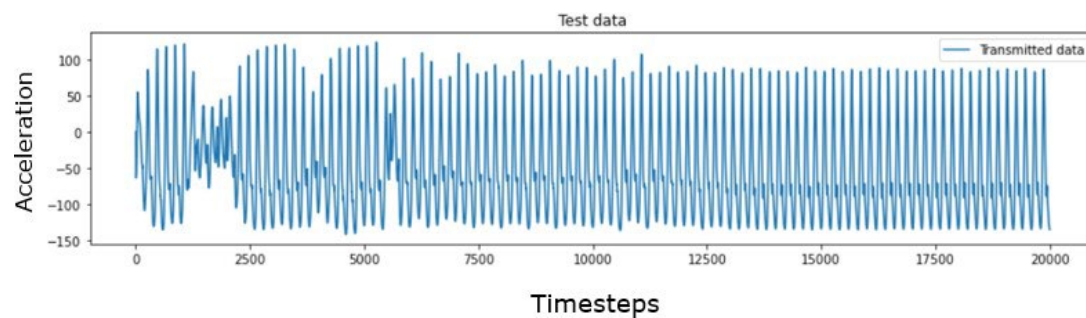
(b) Node 1 Y-Position



(c) Node 1 X-Speed

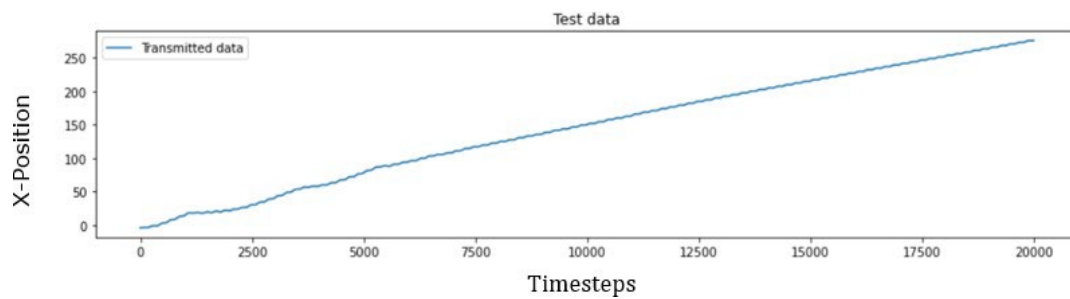


(d) Node 1 Y-Speed

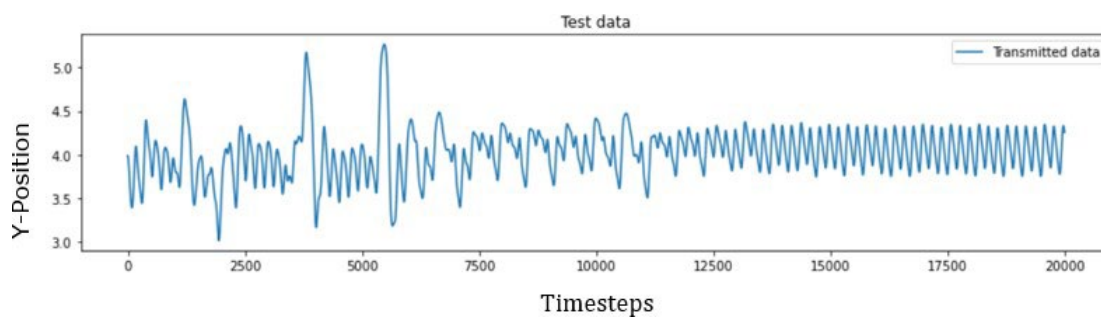


(e) Node 1 Acceleration

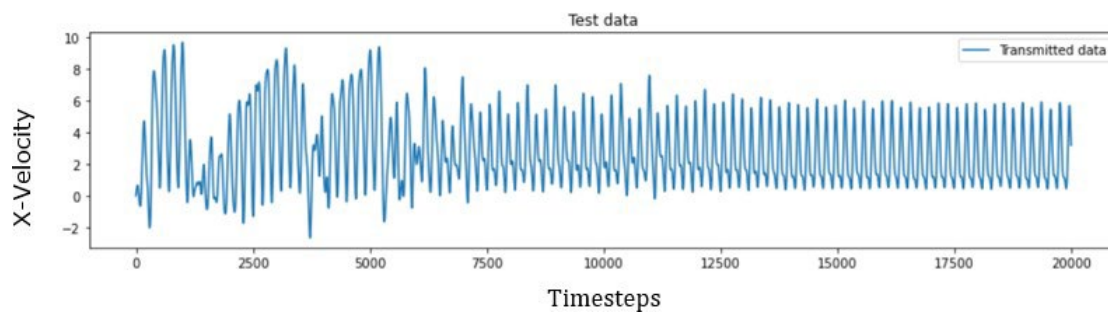
Fig. 4.2. Evolution of Node 1 Dynamics over 2,000 time steps



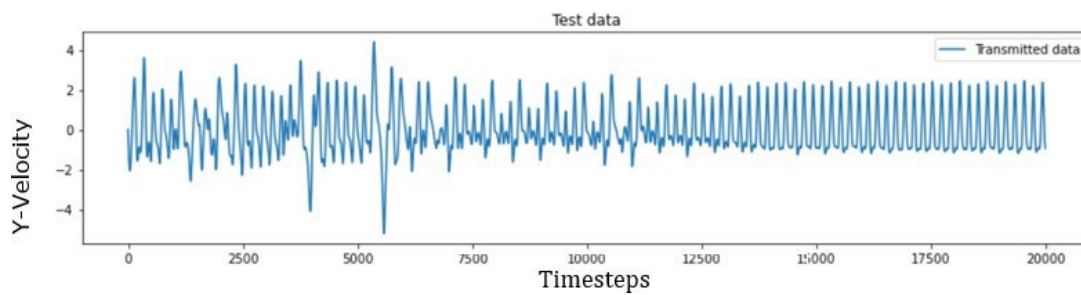
(a) Node 11 X-Position



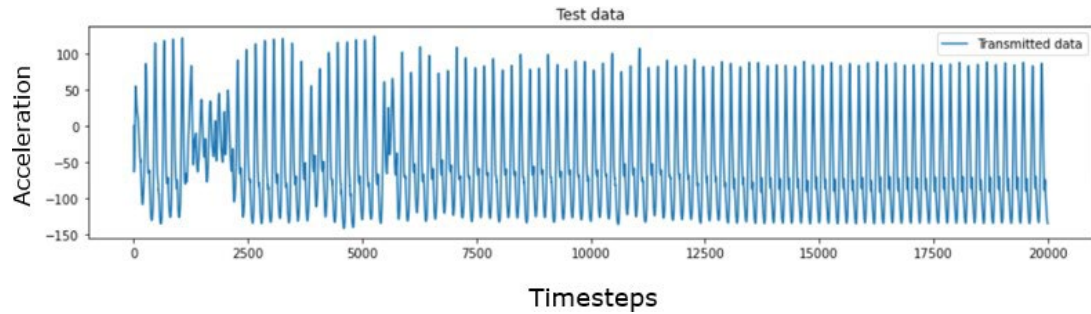
(b) Node 11 Y-Position



(c) Node 11 X-Speed

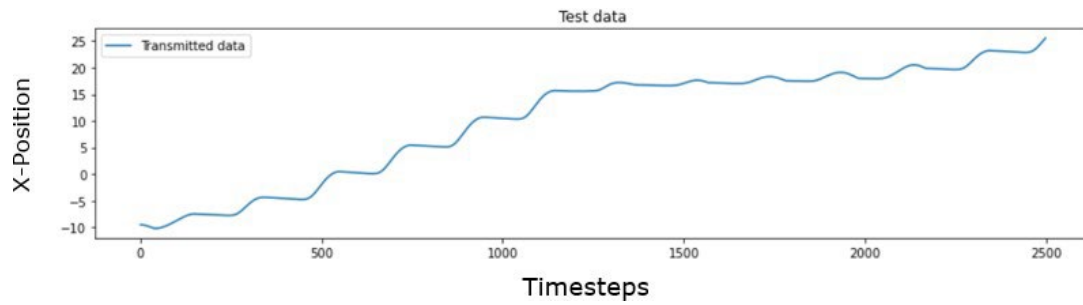


(d) Node 11 Y-Speed

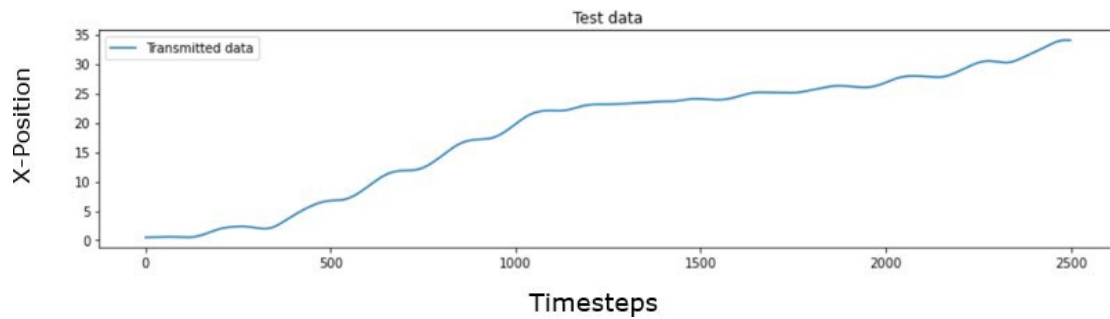


(e) Node 11 Acceleration

Fig. 4.3. Evolution of Node 11 Dynamics over 2,000 time steps



(a) Node 1 X-Position



(b) Node 11 X-Position

Fig. 4.4. Detailed View of X-Position Evolution over 2,000 time steps

4.2.2 Control Experiment Set-Up. From the simulation explained, I utilize the speed and position states to start the experiments. Each of the input vectors consisted of the speed, position and acceleration components at each time step, in the X and Y directions. Each of the output vectors consisted of the control signal for each spring in the structure.

The overall goal was to optimize motion within robotic structures, utilizing reservoir computing to map dynamic inputs to control outputs. In doing the following experiments, I simulated 5, 10, 20 and 50 node structures with 9, 24, 54, and 144 springs, respectively. The label data dimensions are shown in Table 4.2. With optimal parameters moving a structure through time, I investigated mapping the trajectory of the robot to optimal spring parameters. Each input was the movement trajectory of each node of the robot, represented in Table 4.1, while the output was the correlated control length of each spring connected to each node, described in Table 4.2. The difference in dimensions shown in Table 4.3, was also a variable to be studied.

Table 4.3. CONTROL SIMULATION DATA DESCRIPTION

Category	Input Dimensions	Output Dimensions
Data 1	100,000×21	100,000×9
Data 2	100,000×41	100,000×24
Data 3	100,000×81	100,000×54
Data 4	100,000×201	100,000×114

4.3 Evaluation Metrics

Most of the experiments were evaluated using the Mean Absolute Percent Error (MAPE) metric. It represents the average of the absolute percentage errors of each input to calculate how accurate the predicted quantities were in comparison with the actual quantities. To compute the MAPE, percentage errors are added together without respect to sign, as shown in equation 4.1 where A_t is the actual value and F_t is the predicted value.

$$MAPE = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right|$$

Eq. (4.1)

MAPE is often effective for analyzing large sets of data and requires the use of dataset values other than zero which may result in undefined or infinite MAPE. This drawback occurs if the Y position of the node is located at zero or smaller

than 0.0001. Due to this, a nominal value of 0.7 is added to the input if it falls in that range. In all experiments, the average MAPE was taken for three to five trials for each experiment.

4.4 In-depth Performance of Reservoir Computing for Optimal Dynamics

In addition to performing the experiments on the RC model, I also tested the performance of the data against other deep learning models. I utilized the LSTM and DNN, due to their nature of having the ability to perform on time-series and more complex data. I then performed a deeper analysis on the proposed RC model with different parameter set-ups.

4.4.1 Comparison of Performance between Deep Learning Models. Deep learning models and ESNs were trained, and evaluated using Data 1 (5 Node Structure), Data 3 (20 Node Structure), and Data 4 (50 Node Structure). Model parameters used are provided in Table 4.4. For clarification, α is the learning rate, H_L is the number of hidden layers, L_N is the number of nodes per hidden layer, B_s is the batch size and N_E is the number of epoch used. In the first column, I_M was the initialization method used, $\rho(W)$ was the spectral radius, N was the size of the given reservoir, A_F was the activation function used, and R_M was the regression method used for training the output layer. X is Xavier initialization method, G is normalized Xavier initialization (gloriot) method, HE is HE initialization method, Ri is ridge regression, Li is linear regression, La is lasso regression and HT is hyperbolic tangent [53].

Results from these experiments were recorded in Table 4.5 . From the

results, I observed that the RC had better performance LSTM and DNN on all datasets. It is also observed that as the structure complexity increases, the performance decreases for all models as well. The RC model performs over 5% better than the other DNN and LSTM models studied.

Table 4.4. MODEL PARAMETERS FOR DIFFERENT MODELS

Parameters		
RC	DNN	LSTM
$I_M = [\text{Xavier, Random, Gloriot, He}]$	$\alpha = 0.001$	$\alpha = 0.001$
$N = [21, 41, 81, 201]$	$H_L = 6$	$H_L = 6$
$R_M = [\text{Li, Ri, La}]$	$L_N = 256$	$L_N = 256$
$A_T = \text{HT}$	$B_S = 64$	$B_S = 64$
$\rho(W) = 0.9$	$N_E = 100$	$N_E = 100$

In RC principles, computational time and power are very important. For the purposes of soft-bodied robotics, the time and energy it takes for the body to learn and perform actions can affect the overall mobility and ability to perform actions. Shown in Table 4.5, the training times given the various networks varied drastically. For the LSTM and DNN, training time significantly increased as the complexity increased. In the RC, the training time still stayed relatively lower, averaging several minutes. The experimental comparison is shown in Table 4.5

Table 4.5. PERFORMANCE OF THE DEEP LEARNING MODELS ON EACH DATASET

MAPE(%)								
Model	Data 1	Time(s)	Data 2	Time(s)	Data 3	Time(s)	Data 4	Time(s)
RC	1.19	302	2.22	418	2.51	577	4.25	701
LSTM	6.95	5420	8.15	6812	10.31	8645	19.11	1212
DNN	9.76	1132	10.02	1412	12.52	2248	31.12	4561

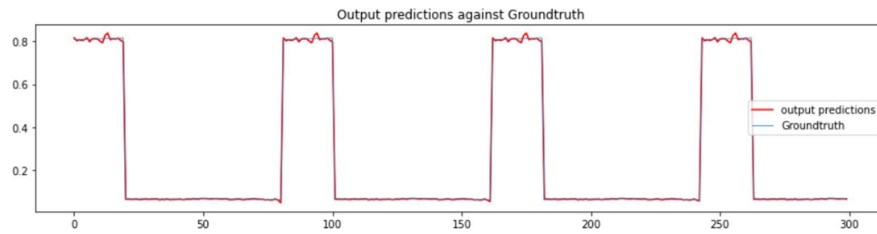


Fig. 4.5. Sample Plot of RC Model Predicted Data

4.5 Performance of Reservoir Computing Models with different Set- Ups

The experiments carried out were also given different set-ups to provide a more in-depth examination of how the model performed under different conditions. I varied several parameters including: initialization methods, readout methods, and reservoir size.

4.5.1 Reservoir Sizes. One of the more impactful parameters of the RC architecture is the size of the reservoir, referring to the number of recurrent memory

units with the reservoir. Typically, reservoirs can be designed to have the same number of units as the input dimension, but that is not always the case. Due to this, I investigated the effect of varying the number of nodes in the reservoir, N , on the performance of the model while keeping other parameters (spectral radius, initialization method, and the activation function) constant. Results from this experiment are as recorded in Table 4.6. Fig. 4.6 shows a graphical plot of the MAPE values for the different reservoir sizes across all the data categories.

From these results I observed how the models were optimized as the internal reservoir units were equal to the input dimension into the reservoir. This gives a better understanding of how to configure the reservoir for optimal dynamics.

Table 4.6. MODEL PERFORMANCE COMPARISON FOR DIFFERENT
RESERVOIR SIZES

Mape(%)				
Reservoir Size, N	Data 1	Data 2	Data 3	Data 4
= input dimension	1.89	2.41	2.51	3.12
25	1.77	2.54	2.95	5.83
50	1.93	2.52	2.66	7.96
100	2.58	2.98	2.72	12.56
200	2.31	4.88	3.78	5.77

250

4.21

4.23

5.22

7.96

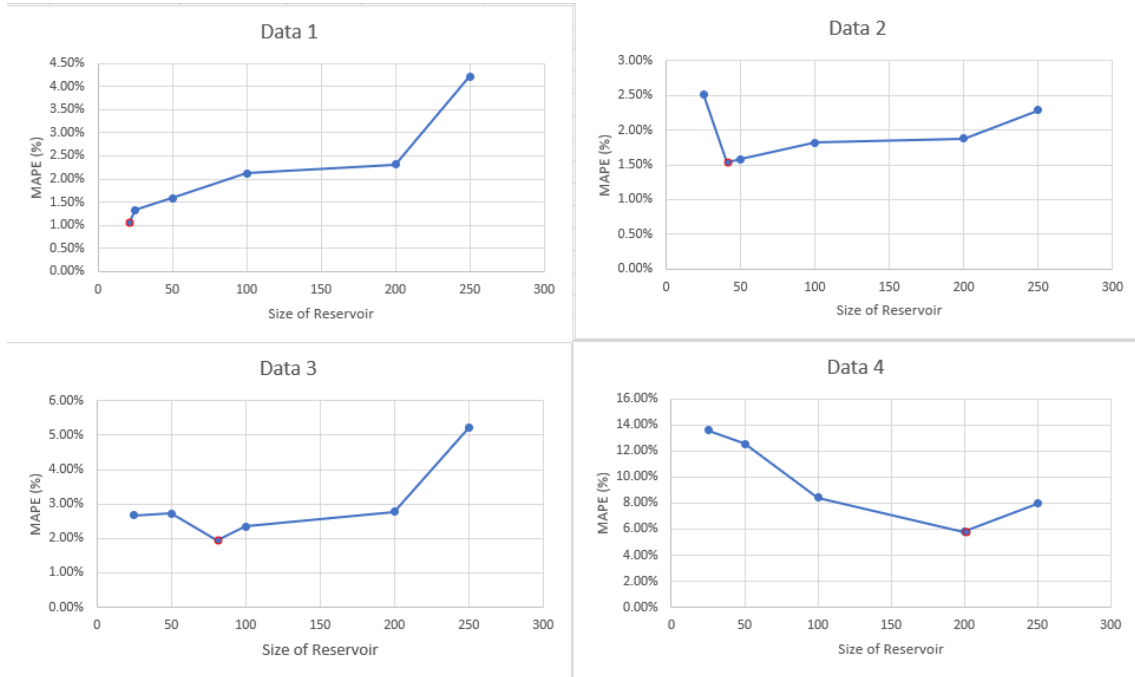


Fig. 4.6. Plots of Model Performance on Different Reservoir Sizes

4.5.2 Initialization Methods. Currently, RC was not as constrained as there was no systematic way of defining the dynamic reservoir. The reservoir is usually randomly generated at network initialization. In addition to randomly initializing the reservoir, I explored the option of using some other weight initialization approaches used in general deep learning [54][55] and investigated how they affect the performance of the ESN. I ran experiments on all datasets using the random, Xavier, normalized Xavier (gloriot), and HE initialization methods while keeping all other parameters constant. The model performances when each of the initialized methods were used are recorded in Table 4.7. From the results, I can observe

how the model performed best using Xavier initialization for weight matrices.

Table 4.7. MODEL PERFORMANCE COMPARISON FOR DIFFERENT
INITIALIZATION APPROACHES

	MAPE(%)			
	Xavier	Random	HE	Gloriot
Data 1	1.89	5.09	7.01	7.15
Data 2	2.22	5.25	6.10	7.59
Data 3	2.49	4.53	3.52	5.22
Data 4	4.87	10.02	13.33	16.67

4.5.3 Readout Methods. In the basic RC framework, the only training is done at the readout layer. To reduce error at this readout layer, I also evaluated the three different regression models on the performance of the network: linear, ridge, and lasso regression, while keeping all other parameters constant. The results from these experiments are observed in Table 4.8.

Table 4.8. MODEL PERFORMANCE COMPARISON FOR DIFFERENT READOUT METHODS

	MAPE(%)		
	Linear	Ridge	Lasso
Data 1	1.98	1.96	2.12
Data 2	2.89	2.65	2.79
Data 3	2.95	2.79	2.96
Data 4	6.51	5.12	6.62

4.6 Performance of Enhanced Reservoir Computing Models

4.6.1 Pre-Trained Models. In attempts to improve the performance of the RC, I experimented with the use of pre-trained deep learning models as a replacement for the reservoir, instead of randomized weights. For these experiments, I used dataset 3 on DNN and LSTM models, due to the 20 node structure having moderate modeling complexity. I then used a pre-trained DNN and LSTM deep learning model to evaluate the performance. The results are shown in Table 4.9. Looking at the data, the pre-trained models showed some improvements in comparison to the standard DNN and LSTM, but not enough to consider it an effective replacement for the standard, fixed reservoir.

Table 4.9. MODEL PERFORMANCE COMPARISON FOR PRE-TRAINED
MODELS

Model Performance Comparison for Pre-Trained Models	
Model	MAPE(%)
ESN	2.58
LSTM	6.31
DNN	6.81
ESN + LSTM	4.25
ESN + DNN	5.95

4.6.2 DeepESN. In another attempt to improve the performance of the ESN, I experimented with a layered RC structure in which each reservoir was successively passed into the other. A DeepESN is a deep Recurrent Neural Network composed by a hierarchy of recurrent layers intrinsically able to develop hierarchical and distributed temporal features. Such characteristics make DeepESN suitable for time-series and sequence processing [4]. I experimented with various numbers of layers to effectively determine the performance of this approach. The results are

observed in Table 4.10. From the results, I observed that the layered approach provided decent accuracy but was not suitable for complex tasks as performance deteriorates with an increase in depth.

Table 4.10. MODEL PERFORMANCE COMPARISON FOR DEEP ECHO STATE NETWORK

Model Performance for Deep Echo State Network	
Layers	MAPE(%)
1	2.42
2	3.31
5	6.81
10	10.25
20	25.95

4.7 In-depth Performance of Reservoir Computing for Optimal Control

For optimal control in soft robotic structures, I investigated how the dynamics of the structure could be mapped to the length of each spring of the mass-spring network. Through various simulations, I was able to observe and analyze how the dynamics of the moving structure could be used to control the length of its springs, providing optimal control parameters. As shown in Fig. 4.7, springs can vary in

length and the nodes they connect to, which affects the overall dynamics for that spring and mass. For example, spring 1 would have less movement veracity compared to spring 14 as it connects two nodes with more dynamic movement. These signals were chosen to highlight and plot as they provided a more comprehensive look at how the configuration of the structure affected the kinematics and control features, thus making modeling more of a challenge.

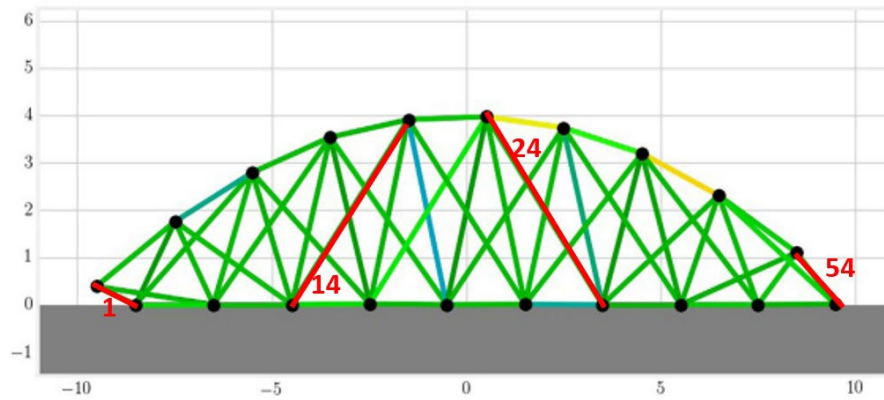


Fig. 4.7. 20 node structure with modeled springs

4.7.1 Performance of Reservoir Computing Model. Mapping highly linear dynamics accurately can provide an outlook into soft robotic control. As explained, the inputs represent dynamic movement trajectory, including the position, speed and acceleration at each node over time, shown in Figs. 4.2 and 4.3. These inputs are mapped into the lengths for each spring, resulting in optimal control for moving robotic masses. To accurately model the controls of a robot, I analyzed spring mass structures of 5,10, 20 and 50 nodes. The results of the RC

model on each dataset is shown in Table 4.11. From the results observed, it showed how the RC framework was able to provide accuracy and efficiency when predicting non-linear signals. For a moderately complex structure, Data 3, the model showed less than 10% error while also having a relatively low training time. For soft robotic purposes, this is important as it will allow the robot to be controlled with decent precision while also requiring less training time and data.

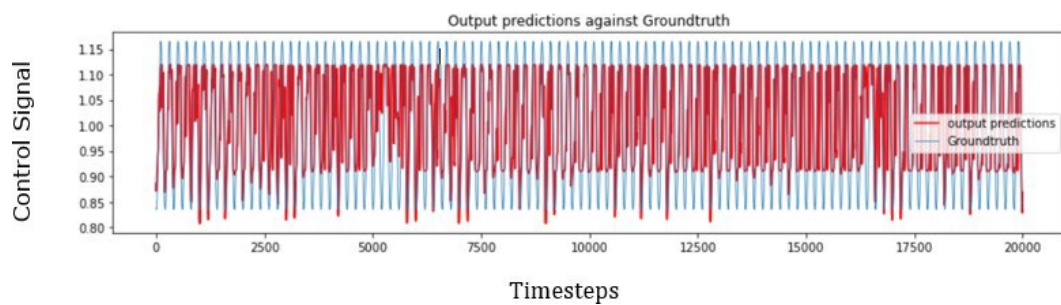
Depicted in Fig. 4.7, I chose to highlight a 20 mass structure with 54 connecting springs. This provided a scope of a moderately complex structure with a significant number of parameters to tune. The prediction is tested spring lengths

Table 4.11. PERFORMANCE OF RC MODEL ON CONTROL PARAMETERS

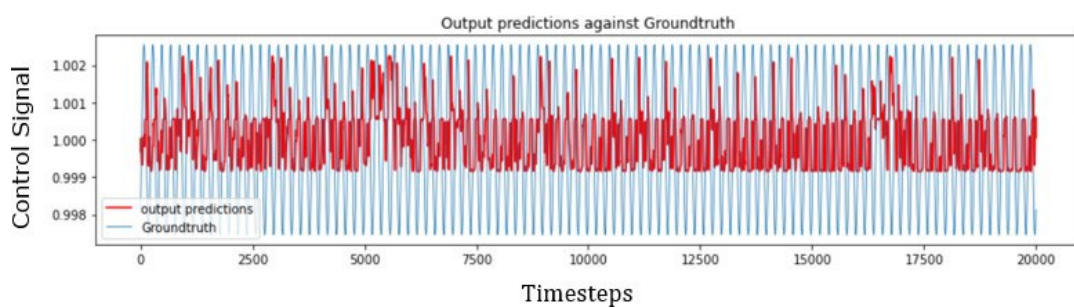
Model Performance for Reservoir Computing Model

Category	MAPE(%)	Training Time(s)
Data 1	6.02	321
Data 2	7.71	420
Data 3	8.67	598
Data 4	13.01	766

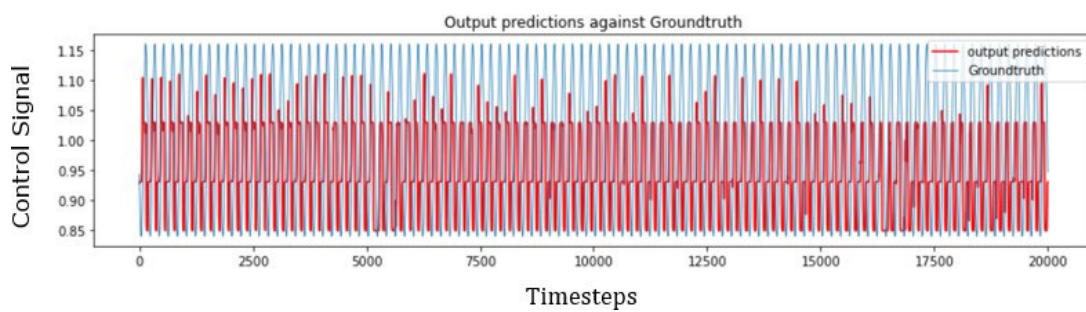
over 20,000 time steps are shown in Fig. 4.8. From these modeled signals, I observed how they did not exactly align with the true sin wave inputs due to the non-linear characteristics of the data. Overall, the model still provides a decent representation of the predicted control signal.



(a) Spring 1



(b) Spring 14



(c) Spring 24

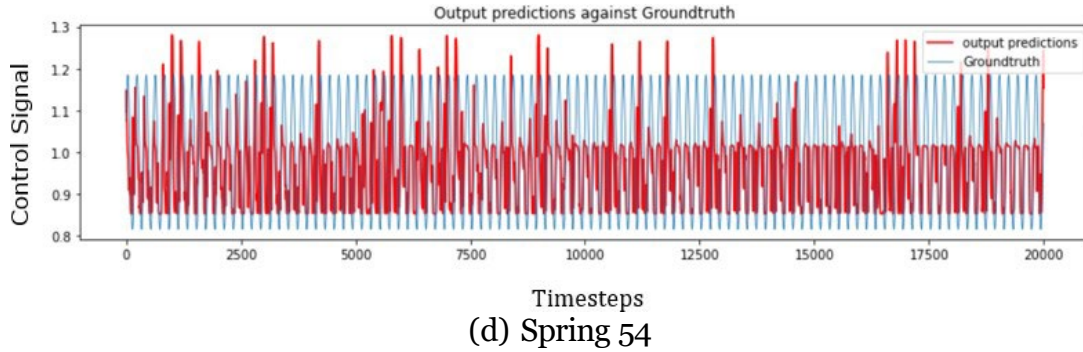


Fig. 4.8. Tested Spring Control Signals over 20,000 time steps

4.7.2 Performance of Different Reservoir Sizes. Explained in previous results, one of the more impactful parameters of the RC architecture is the size of the reservoir. In investigating of the dynamics, I observed how the model performed best when the size of the reservoir equaled the input data dimension. Due to this, I investigated the effect of varying the number of nodes in the reservoir, N , on the performance of the model while keeping other parameters constant. Results from this experiment are as recorded in Table 4.12.

From these results, it is observed how the models were optimized as the internal reservoir units were equal to the input dimension into the reservoir. This gives a better understanding of how to configure the reservoir for optimal control.

Table 4.12. MODEL PERFORMANCE COMPARISON FOR DIFFERENT
RESERVOIR SIZES

Mape (%)				
Reservoir Size, N	Data 1	Data 2	Data 3	Data 4
= input dimension	6.03	7.11	8.52	13.12
25	6.13	7.51	8.78	14.80
50	6.33	7.61	8.92	14.67
100	7.58	7.98	8.72	14.56
200	8.31	8.18	9.78	13.77
250	9.21	9.23	9.82	14.26

4.7.3 Performance of Different Initialization Methods. From previous experiments, it is observed that different initialization methods can have an effect on the performance of the model. I ran experiments on all datasets using the random, Xavier, normalized Xavier (gloriot), and HE initialization methods while keeping all other parameters constant. The model performances when each of the initialized methods were used are recorded in Table 4.13. From the results, I can observe how the model performed best using Xavier initialization for weight matrices.

Table 4.13. MODEL PERFORMANCE COMPARISON FOR DIFFERENT
INITIALIZATION APPROACHES

MAPE(%)				
	Xavier	Random	HE	Gloriot
Data 1	6.02	7.09	7.41	7.25
Data 2	7.71	8.25	8.10	8.89
Data 3	8.67	8.93	9.42	9.22
Data 4	13.01	14.02	16.33	16.67

4.7.4 Performance of Different Readout Methods. To reduce error at the readout layer, I evaluated the three different regression models: linear, ridge, and lasso regression, while keeping all other parameters constant. The results from these experiments are observed in Table 4.14. In the results of varying the initialization methods, one can see how each method had decent performance but performed best using ridge regression as the readout layer.

Table 4.14. MODEL PERFORMANCE COMPARISON FOR DIFFERENT
READOUT METHODS

	MAPE(%)			
	Ridge	Linear	Lasso	
Data 1	6.05	6.11	6.91	7.25
Data 2	7.86	8.30	8.19	8.89
Data 3	8.79	8.94	9.95	9.02
Data 4	13.52	14.42	16.33	16.77

4.8 Summary of Results

I performed many experiments over various methods to provide a vast amount of data needed to analyze how effective this approach could be to perform complex functions, such as soft robotics. Observations from these experimental results are summarized below: Section B

- RC models had better performance than the popular deep learning models such as DNN and LSTM by a significant margin. It also required much less training time, which is desirable for morphological computation.
- RC was very robust and efficient using different activation functions and regression methods used for training the output layer. When analyzing Table 4.5, there was no significant margin of error from each approach, allowing for this RC framework to be flexible.

- As expected, when the size of the reservoir matched the size of the input sequence into an RC model, the model had better performance. In Tables 4.6 and 4.12, the models performed best when equivalent to the length input of the given structure. Since each dataset provided a different input size, the results showed a clear observation of the effects of the size of the reservoir.
- RC can be used to efficiently model soft robotic structures as a digital twin. The RC model can efficiently learn the desired kinematics for a robotic structure of different configurations.
- RC showed the decent ability to model highly-nonlinear dynamic inputs and outputs, being able to predict the control signal from the desired movement kinematics. Shown in Fig. 4.8, the model can map the spring control signals within the robot from its desired movement trajectory, thus providing a deeper understanding of soft robotic control.
- Utilizing pre-trained models showed some minor improvements with respect to its initial model, DNN and LSTM, but it did not prove to be an effective replacement reservoir.
- While a multi-layered reservoir approach can be useful in some complex tasks, the approach was not effective enough for this data. The architecture can show greater performance with minimal layers but performs worse as more layers are introduced.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

Soft-bodied robots have attracted a lot of attention recently because of their capabilities of performing sophisticated tasks that rigid-bodied robots cannot. In this study, methods based on reservoir computing were proposed to address the challenges of modeling and control of future soft-bodied robots.

Specifically, the first goal of this study was to model complex, spring-mass system dynamics using reservoir computing techniques. This study showed that even though these can be modeled using general machine learning approaches with relatively good performance, the digital twin created using the principle of reservoir computing achieved better accuracy.

The second objective was to understand how complex systems are controlled. To achieve this, a simulator was used to provide optimal locomotion and spring parameters of a spring-mass system. This provided a clear understanding of how the body reacted under stress at different times. Using these mass-spring systems and reservoir computing allowed for modeling the node trajectory dynamics, such as position and speed, to control the lengths of the springs between the nodes.

Lastly, I tested different methods in which to use different reservoir sizes and configurations. In these cases, I varied the size of the reservoir to observe how

the number of recurrent units affected the reservoir control. Also, I tested how the reservoir component of the network was replaced with different pre-trained deep learning models and adding successive reservoirs in a layered approach. Overall, some of these approaches gives a moderate performance, they are not as efficient as the proposed conventional reservoir computing model with dynamic reservoirs.

In sum, the proposed digital twin and optimal controller based on reservoir computing showed excellent performance for future soft-bodied robots. Because a reservoir computing based approach not only has high accuracy but also requires much less computational resources and training time, makes it a promising solution for modeling and real-time control of complex dynamical systems.

5.2 Future Work

For the future of soft-bodied robotics, research is still being conducted to understand how these systems will be accurately modeled and utilized for optimal performance. The next phase of research will focus on optimizing these models to generate a physical reservoir computer with optimization to achieve desired dynamics. This can allow for physical robotic control, thus potentially being an innovative approach for larger, more complex structures.

REFERENCES

- [1] K. Caluwaerts, M. D’Haene, D. Verstraeten, and B. Schrauwen, “Locomotion Without a Brain: Physical Reservoir Computing in Tensegrity Structures,” *Artificial Life*, vol. 19, no. 1, p. 35–66, 2013.
- [2] I. N. Lenz and A. Saxena, *Deep Learning for Robotics*. PhD thesis, 2016.
- [3] C. Quilodrán-Casas, V. L. Silva, R. Arcucci, C. E. Heaney, Y. Guo, and C. C. Pain, “Digital Twins Based on Bidirectional lstm and gan for Modelling the Covid-19 Pandemic,” *Neurocomputing*, vol. 470, p. 11–28, 2022.
- [4] C. Gallicchio, A. Micheli, and L. Pedrelli, “Design of Deep Echo State Networks,” *Neural Networks*, vol. 108, p. 33–47, 2018.
- [5] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, “Recent Advances in Physical Reservoir Computing: A Review,” *Neural Networks*, vol. 115, p. 100–123, 2019.
- [6] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, “Deep Physical Neural Networks Trained with Backpropagation,” *Nature*, vol. 601, no. 7894, p. 549–555, 2022.
- [7] A. Morán, C. F. Frasser, and J. L. Rosselló, “Reservoir Computing Hardware with Cellular Automata,” *ArXiv*, vol. abs/1806.04932, 2018.
- [8] A. R. Omondi and J. C. Rajapakse, *FPGA implementations of Neural Networks*. Scholars Portal, 2006.
- [9] J. Zhu and P. Sutton, “Fpga Implementations of Neural Networks – A Survey of a

- Decade of Progress,” *Field Programmable Logic and Application*, p. 1062–1066, 2003.
- [10] P. Antonik, M. Gulina, J. Pauwels, and S. Massar, “Using a Reservoir Computer to Learn Chaotic Attractors, with Applications to Chaos Synchronization and Cryptography,” *Physical Review E*, vol. 98, no. 1, 2018.
- [11] V. C. Müller and M. Hoffmann, “What is Morphological Computation? On How the Body Contributes to Cognition and Control,” Feb 2017.
- [12] P. Bhovad and S. Li, “Physical Reservoir Computing with Origami and its Application to Robotic Crawling,” Jun 2021.
- [13] H. Hauser, A. J. Ijspeert, R. M. Fuchslin, R. Pfeifer, and W. Maass, “Towards a Theoretical Foundation for Morphological Computation with Compliant Bodies - Biological Cybernetics,” Jan 2012.
- [14] M. J. Kaur, V. P. Mishra, and P. Maheshwari, “The Convergence of Digital Twin, IOT, and Machine Learning: Transforming Data into Action,” *Internet of Things*, p. 3–17, 2019.
- [15] L.-W. Kong, Y. Weng, B. Glaz, M. Haile, and Y.-C. Lai, “Reservoir Computing as Digital Twins for Nonlinear Dynamical Systems,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 3, p. 033111, 2023.
- [16] G. M. Whitesides, “Soft Robotics,” *Angewandte Chemie International Edition*, vol. 57, no. 16, p. 4258–4273, 2018.
- [17] K. Nakajima, “Physical Reservoir Computing—An Introductory Perspective,” *Japanese Journal of Applied Physics*, vol. 59, no. 6, p. 060501, 2020.

- [18] K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, “A Soft Body as a Reservoir: Case Studies in a Dynamic Model of Octopus-inspired Soft Robotic Arm,” *Frontiers in Computational Neuroscience*, vol. 7, 2013.
- [19] Y. Lan, N. Huang, Y. Fu, K. Liu, H. Zhang, Y. Li, and S. Yang, “Morphology- based Deep Learning Approach for Predicting Osteogenic Differentiation,” *Frontiers in Bioengineering and Biotechnology*, vol. 9, 2022.
- [20] B. Premjith and K. P. Soman, “Deep Learning Approach for the Morphological Synthesis in Malayalam and Tamil at the Character Level,” *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 20, no. 6, p. 1–17, 2021.
- [21] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, and et al., “Review of Machine Learning Methods in Soft Robotics,” *PLOS ONE*, vol. 16, no. 2, 2021.
- [22] C. C. Johnson, T. Quackenbush, T. Sorensen, D. Wingate, and M. D. Killpack, “Using First Principles for Deep Learning and Model-based Control of Soft Robots,” *Frontiers in Robotics and AI*, vol. 8, 2021.
- [23] Y. Lan, N. Huang, Y. Fu, K. Liu, H. Zhang, Y. Li, and S. Yang, “Morphology-based Deep Learning Approach for Predicting Osteogenic Differentiation,” *Frontiers in Bioengineering and Biotechnology*, vol. 9, 2022.
- [24] E. A. Antonelo and B. Schrauwen, “On learning Navigation Behaviors for Small

- Mobile Robots with Reservoir Computing Architectures,” *IEEE transactions on neural networks and learning systems*, vol. 26, no. 4, pp. 763–780, 2014.
- [25] E. A. Antonelo and B. Schrauwen, “Supervised Learning of Internal Models for Autonomous Goal-oriented Robot Navigation Using Reservoir Computing,” in *2010 IEEE International Conference on Robotics and Automation*, pp. 2959– 2964, IEEE, 2010.
- [26] T. Kawase, T. Miyazaki, T. Kanno, K. Tadano, Y. Nakajima, and K. Kawashima, “Pneumatic Reservoir Computing for Sensing Soft Body: Computational Ability of Air in Tube and its Application to Posture Estimation of Soft Exoskeleton,” *Sens. Mater*, vol. 33, pp. 2803–2824, 2021.
- [27] T. Li, K. Nakajima, M. Cianchetti, C. Laschi, and R. Pfeifer, “Behavior Switching Using Reservoir Computing for a Soft Robotic Arm,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4918–4924, 05 2012.
- [28] G. Urbain, J. Degraeve, B. Carette, J. Dambre, and F. Wyffels, “Morphological Properties of Mass–Spring Networks for Optimal Locomotion Learning,” *Frontiers in Neurorobotics*, vol. 11, 2017.
- [29] Z. Kan, H. Peng, B. Chen, X. Xie, and L. Sun, “Investigation of Strut Collision in Tensegrity Statics and Dynamics,” *International Journal of Solids and Structures*, vol. 167, p. 202–219, 2019.
- [30] K. Nakajima, “Muscular-hydrostat computers: Physical Reservoir Computing for Octopus-inspired Soft Robots,” *Diversity and Commonality in Animals*, p. 403–414, 2017.

- [31] Y. Horii, K. Inoue, S. Nishikawa, K. Nakajima, R. Niiyama, and Y. Kuniyoshi, “Physical Reservoir Computing in a Soft Swimming Robot,” *The 2021 Conference on Artificial Life*, 2021.
- [32] L. Girletti, M. Groshev, C. Guimarães, C. J. Bernardos, and A. de la Oliva, “An Intelligent Edge-based Digital Twin for Robotics,” in *2020 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2020.
- [33] “Using Hybrid Physics-informed Neural Networks for Digital Twins in Prognosis and Health Management,” Aug 2022.
- [34] S. Gu, E. Holly, T. Lillicrap, and S. Levine, “Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-policy Updates,” *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [35] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, “Residual Reinforcement Learning for Robot Control,” *2019 International Conference on Robotics and Automation (ICRA)*, 2019.
- [36] S. Zilberstein and S. J. Russell, “Anytime Sensing, Planning and Action: A Practical Model for Robot Control,” in *IJCAI*, vol. 93, pp. 1402–1407, 1993.
- [37] G. Crabé, “Optimal Control for Robotics: Part 1,” Sep 2021.
- [38] R. Bogue, “The Role of Artificial Intelligence in Robotics,” *Industrial Robot: An International Journal*, 2014.
- [39] T. Jiang, J. L. Gradus, and A. J. Rosellini, “Supervised Machine Learning: A Brief

- Primer,” *Behavior Therapy*, vol. 51, no. 5, p. 675–687, 2020.
- [40] O. F.Y, A. J.E.T, A. O, H. J. O, O. O, and A. J, “Supervised Machine Learning Algorithms: Classification and Comparison,” *International Journal of Computer Trends and Technology*, vol. 48, no. 3, p. 128–138, 2017.
- [41] H. Tan, “Machine Learning Algorithm for Classification,” *Journal of Physics: Conference Series*, vol. 1994, no. 1, p. 012016, 2021.
- [42] S.-J. Kim, S.-J. Bae, and M.-W. Jang, “Linear Regression Machine Learning Algorithms for Estimating Reference Evapotranspiration Using Limited Climate Data,” *Sustainability*, vol. 14, no. 18, p. 11674, 2022.
- [43] J. J. Moolayil, “A Layman’s Guide to Deep Neural Networks,” May 2020.
- [44] N. M. Rezk, M. Purnaprajna, T. Nordström, and Z. Ul-Abdin, “Recurrent Neural Networks: An Embedded Computing Perspective,” *IEEE Access*, vol. 8, pp. 57967–57996, 2020.
- [45] Y. Sudriani, I. Ridwansyah, and H. A Rustini, “Long Short-Term Memory (lstm) Recurrent Neural Network (rnn) for Discharge Level Prediction and Forecast in Cimandiri River, Indonesia,” *IOP Conference Series: Earth and Environmental Science*, vol. 299, p. 012037, 2019.
- [46] H. Jaeger, “Echo State Network,” *Scholarpedia*, vol. 2, no. 9, p. 2330, 2007.
- [47] A. Holm, A. T. Gayle, J. Gomila, M. Mangus, and T. He, “Reservoir Computing - Wiki.”
- [48] X. Wang, G. Mao, J. Ge, M. Drack, G. S. Cañón Bermúdez, D. Wirthl, R. Illing, T. Kosub, L. Bischoff, C. Wang, and et al., “Untethered and Ultrafast

Soft-bodied Robots,” Sep 2020.

- [49] S. Sayyaparaju, M. S. Ara Shawkat, and G. S. Rose, “Robust Implementation of Memristive Reservoir Computing with Crossbar Based Readout Layer,” in *2020 IEEE 14th Dallas Circuits and Systems Conference (DCAS)*, pp. 1–4, 2020.
- [50] O. Krause, D. R. Arbonès, and C. Igel, “CMAES with Optimal Covariance Update and Storage Complexity,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [51] P. Almasan, M. Ferriol-Galmés, J. Paillisse, J. Suárez-Varela, D. Perino, D. López, A. A. P. Perales, P. Harvey, L. Ciavaglia, L. Wong, et al., “Network Digital Twin: Context, Enabling Technologies, and Opportunities,” *IEEE Communications Magazine*, vol. 60, no. 11, pp. 22–27, 2022.
- [52] E. J. Hermans, M. J. Henckens, M. Joëls, and G. Fernández, “Dynamic Adaptation of Large-scale Brain Networks in Response to Acute Stressors,” *Trends in Neurosciences*, vol. 37, no. 6, p. 304–314, 2014.
- [53] O. Onasami, M. Feng, H. Xu, M. Haile, and L. Qian, “Underwater Acoustic Communication Channel Modeling Using Reservoir Computing,” *IEEE Access*, vol. 10, pp. 56550–56563, 2022.
- [54] T. G. Thuruthel and F. Iida, “Morphological Computation and Control Complexity,” *IOP Conference Series: Materials Science and Engineering*, vol. 1261, no. 1, p. 012011, 2022.
- [55] H. Li, M. Krček, and G. Perin, “A Comparison of Weight Initializers in Deep Learning-based Side-channel Analysis.” Cryptology ePrint Archive, Paper

2020/904, 2020. <https://eprint.iacr.org/2020/904>.

CURRICULUM VITA

Jovan Cain

Department of Electrical and Computer Engineering Roy G. Perry College of
Engineering

Prairie View A&M University, Texas Email: jovancain3@yahoo.com

LinkedIn: <https://www.linkedin.com/in/jovancain/>

Education

M.S. Electrical Engineering, Prairie View A&M University, 2023.

B.S. Electrical Engineering, Prairie View A&M University, 2021.

Employment

Prairie View A&M University, Research Assistant, 2020–2022. IBM, Austin, Texas 1st
Level Packaging Intern, Summer 2022.

NASA-Jet Propulsion Lab, Pasadena, California Motor Control Electronics In-
tern, Summer 2021.

NASA-Jet Propulsion Lab, Pasadena, California Electric Flight Systems Intern,
Summer 2020.

Publications

J. Cain, X. Li, Haile, L. Qian. (2023) “Digital Twin and Optimal Control of
Spring-Mass Systems using Reservoir Computing Models,” to be submitted.