THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF ELECTRICAL ENGINEERING

MODELING AND SIMULATION OF RENEWABLE ENERGY SYSTEMS USING MACHINE LEARNING TECHNIQUES

Anastasia Borochok
SPRING 2023

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Electrical Engineering
with honors in Electrical Engineering

Reviewed and approved* by the following:

Yan Li
Assistant Professor of Electrical Engineering
Thesis Supervisor

Julio Urbina
Associate Professor of Electrical Engineering
Honors Adviser

*Electronic approvals are on file in the Schreyer Honors College.

Abstract

With the growing increase in demand for renewable energy sources, there has also been an increase in demand for research techniques to improve their efficiency. The cost of fossil fuel energy on the planet's C02 levels requires an alternative and cleaner energy. One of the most common and popular alternative energy are photovoltaics. This thesis will specifically focus on photovoltaic (PV) systems. Common photovoltaic systems generally lose a lot of energy in the conversion phases from solar to electrical energy. This loss of energy leads to the overall inefficiency of solar panels. Because renewable energy systems are not very efficient there are many existing methods that attempt to compensate for this deficiency. One technique to improve this inefficiency is called next generation reservoir computing (NGRC), a branch of reservoir computing (RC). NGRC is a subsection of machine learning (ML), using artificial intelligence to model human behavior and better predict data of the systems. Machine learning is a subset of artificial intelligence. Within machine learning, models are built based on existing data. NGRC differs from the older technique, RC, because it requires less computational efforts and shows promising results. The algorithm analyzes previous data and makes predictions for future outcomes. Given a specific training data set, the algorithm of the ML can analyze this data and make suggestions for future recommendations of PV operation. Based upon these predictions, PVs see an increase in power output, addressing the need and dire demand for alternative energy replacements. NGRC analyzes the system using training data sets and linear optimization; it is very efficient because it does not require large and complicated calculations. NGRC is a promising system that with greater implementation may help to better the efficiency of renewable energy like PVs.

Table of Contents

List of Figures

Acknowledgements

Working on this project was not easy but with the great support of my advisor, Dr. Li, I completed my thesis while learning a lot along the way. This journey began in my second year, when I was first introduced to Dr. Li in a research topics class. Dr. Li was giving a presentation about her work with renewable energy and I knew that was something I wanted to work on. Throughout these past years, Dr. Li helped provide me guidance and teach me many technical skills when it comes to writing a thesis. Dr. Li assisted me with understanding technical details alongside writing and formatting metrics. She was also my professor in two courses and built up my baseline understanding to write this paper. She was consistently in communication with me, replying to my emails very quickly, and regularly meeting in person with me to discuss my progress. I am extremely grateful for her and all the work she has done with me.

As for other acknowledgments, I would also like to thank my advisor, Julio Urbina. As my professor and advisor, he was very helpful with any formatting and timing questions that I had. I would also like to thank my boyfriend, Joshua, my dad, Robert, and my brother, Bobby. I can always count on support from them. Finally, I would like to thank my mom, Luba. She was my first teacher and without her motivation and encouragement I certainly would not be where I am today. My mom always pushes me to challenge myself and for this I am grateful because she taught me I can do so much more than I think.

Academic Vitae - Anastasia Borochok

EDUCATION: Bachelor of Science in Electrical Engineering The Pennsylvania State University, University Park, PA Schreyer Honors College Anticipated Graduation: May 2023

ENGINEERING EXPERIENCE: Hardware Engineering Intern Northrop Grumman, Baltimore, MD May-August 2022 • Worked collaboratively on a 3-person team to instantiate core into FPGA • Designed layout for module emulator that successfully enabled testing of LCD screen for team usage • Expertly communicated updates to my supervisor weekly

Research and Development Intern Applied Research Lab, University Park, PA May - August 2021 • Developed prototype of software defined radio (SDR) • Worked on 6-person team to add modifications to SDR using Xilinx Vivado • Designed block diagram of SDR for future project developments

Civil Engineering Intern AECOM, Philadelphia, PA July - Aug 2019 • Designed curb ramps for ADA compliance for DelDOT using AutoCAD • Collaborated with team members to map reconstruction zones

RESEARCH: Honors Thesis Research September 2022 - Present • Tested photovoltaic efficiency using machine learning and Simulink/MATLAB

LEADERSHIP and INVOLVEMENT: Eco Rep, Member (PSU)'21-Present: • Acted as student leader for sustainable development on campus • Planned student outreach events to promote sustainable behavior • Assisted with "Zero-Waste" initiative in President's Box during football games Chair Member, Society of Women Engineers (PSU)'20-Present • Facilitated social events within SWE network for team building • Outreached to high school women interested in engineering through SWE Stayover

SKILLS: • Python, Java, AutoCAD, LabVIEW, MATLAB, Vivado, VHDL, Verilog, SolidWorks

COURSES: • Circuit Analysis I and II, Renewable Energy, Power Systems, Introduction to Nanoelectronics, Communication Systems, Electromagnetics, Power Systems Protection, Optics

Chapter 1

Literature Review

1.1 Photovoltaic Inefficiency

It is no secret that currently, renewable energy and photovoltaic (PV) systems are not efficient in power output. Although recent advancements have seen increases in efficiency, greater improvements [1] are necessary to consolidate for rising global temperatures. The demand to update older systems towards sustainable sources is extremely prevalent during these circumstances. This increase in popularity and urgency has ultimately skyrocketed the demand in research on PV systems. Solar sources have been used to varying degrees of success. Solar cells are typically costly to maintain and have a low energy conversion [1]. With the ever growing demand for a switch toward renewable sources, there has been a global increase in solar power research. The need to recover more energy from PVs is dire. These losses come from inverters, storage, and light gathering inefficiencies [1]. Currently, PV sources can recover about 32% of energy in a laboratory setting, and 15-20% in a real life environment [2]. Figure 1.1 is a graphic from the National Renewable Energy Laboratory showing the calculated efficiency of photovoltaic units from 1976 to present day. In 1976, solar panel efficiency ranged from 1% to 24%. In 2023, efficiency levels reach as high as 49%. The most common type of solar panels are crystalline silicon cells, indicated by the blue lines on the graph. These includes single crystal and multicrystalline cells. The efficiency of these type of PVs is around 21% to 27% [2].As seen from the general trend of the graph, the relationship between time and efficiency is a positive one, implying that with more time and research, the efficiency of photovoltaic units will only continue to increase.

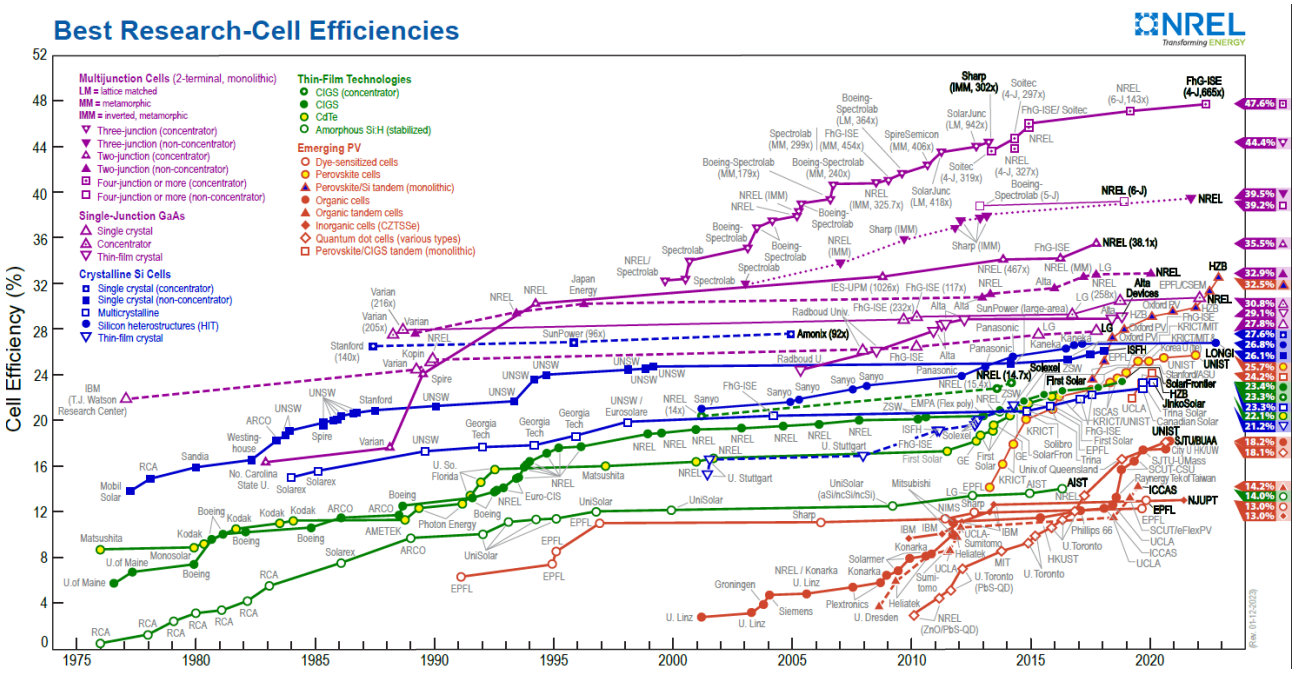

Figure 1.1: NREL graph of solar cell efficiencies from 1976-2023.

Because solar panels are typically located outside, the effects of weather and nature are extremely significant. PVs are subject to wind, rain and a varying range of moderate to extreme weather phenomena; this can lead to damaging of the system and a loss in efficiency of the system [3]. It is difficult to predict how solar panels will perform based upon these fickle outdoor

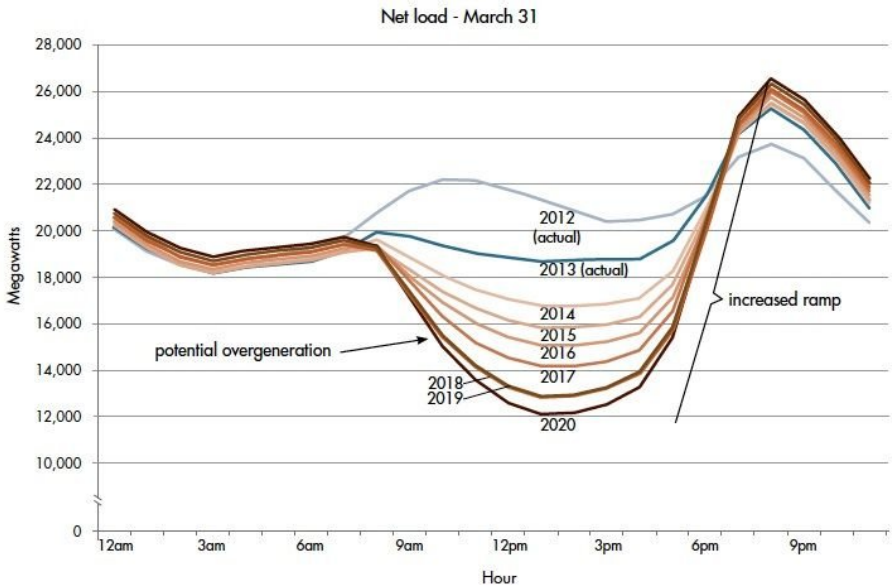

Figure 1.2: Duck curve.

conditions. One important example of some of the problems with photovoltaics is called the Duck Curve, shown in Figure 1.2. The Duck Curve is an example of photovoltaic over-generation, specifically measured in California. It measures the hour of the day versus solar panel power output in Megawatts [4]. The curve resembles the shape of a duck, hence the name. The Duck Curve represents several problems with solar panels. What happens is that PV's generate a lot of power during the peak hours of about 12 pm to 3 pm [4]. However, this also happens to be when people require less electricity because it is sunny outside. This leads to an over generation of power, reducing the environmental and economical benefits of solar cell systems. Also, there is a steep demand for generators to produce a lot of electricity later in the day and the power output produced from PV systems is less [4]. The Duck Curve is just one example of the problems of inefficiency that exist within the PV world.

1.2 Existing Methods to Address PV Inefficiency

There are several current methods to address this problem of PV inefficiency. This paper will explore three of them in the following sections. These methods typically relate to improving the environment of the solar panel after it is constructed and implemented. Although these methods work well in improving efficiency, there is still room for improvement.

1.2.1 Solar Tracking System

Having the PV track the sun is a popular method of increasing solar panel power output. By tracking the sun, the PV can ensure that it is consistently exposed to the maximum amount of solar energy possible. Tracking the position of the sun is relatively simple. It requires a stepper motor and light sensor [1]. This method increases the efficiency of the power collection because it allows the PV to always be at a right angle to the sun's rays. The angle of the PV is essential as

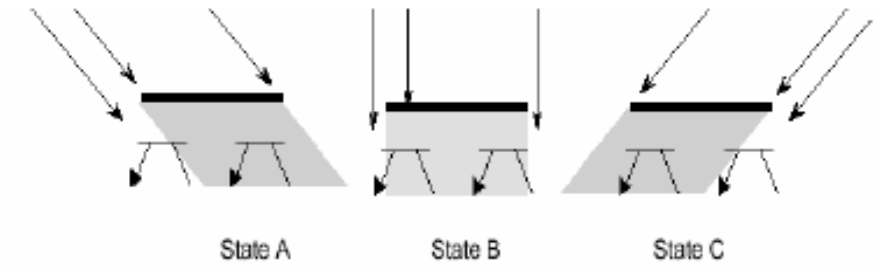

Figure 1.3: LDR approach to solar tracking.

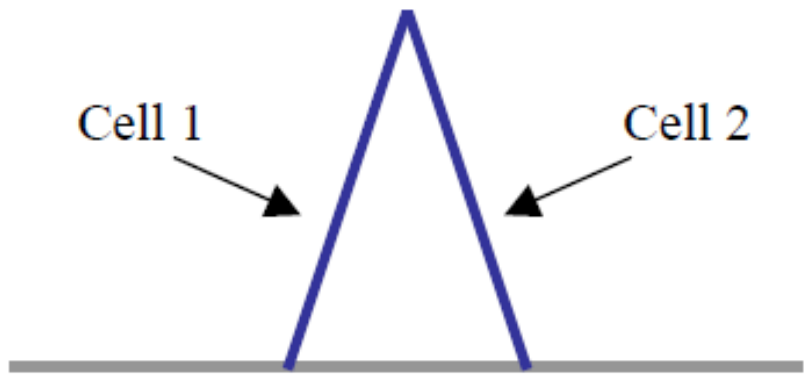

Figure 1.4: Triangular set up of PV systems to obtain equal amounts of sunlight.

it determines the amount of sunlight absorbed. To maintain maximum power, the PV is tilted to maintain an angle of incidence as close to 0° as possible. There are many different methods that achieve this. One of the most typical is done with a Light Dependent Resistor (LDR) [1]. The LDR simply detects when there are changes in light intensity on the surface of the resistor, signaling to the motor that it needs to tilt the panel. Another method includes two phototransistors. Figure 1.3 shows the setup of this method. It starts at the beginning of the day; in State A, the tracker remains in the state it was in from the previous day. Throughout the day, the phototransistors read the levels of light and rotate the panel accordingly, until the panel reaches State C, signaling the end of the day [1].

One problem with solar tracking system is that, specifically for the phototransistor system, it has a narrow range of sensitivity and only operates under the conditions it has been initialized with [1]. As a result of this, Figure 1.4 shows a proposed triangular set up with solar cells facing opposite directions, allowing the system to receive equal amounts of sunlight regardless of the sun's position. The angle of incidence is equal in this case. Thus, unexpected weather conditions can prove a fatal error for these systems, as they are ill-equipped to handle these sudden changes. Additionally, adding a mechanical component will ultimately lead to maintenance problems compared to mounted systems. Having a moving component leads to errors in machinery that can negatively impact the efficiency of the PV system [1]. Finally, solar trackers are often extremely large and are nearly impossible to install on rooftop locations. Therefore, these PV systems will have to be installed in ground-mounted solar farms, leading to a increase in land use and resources.

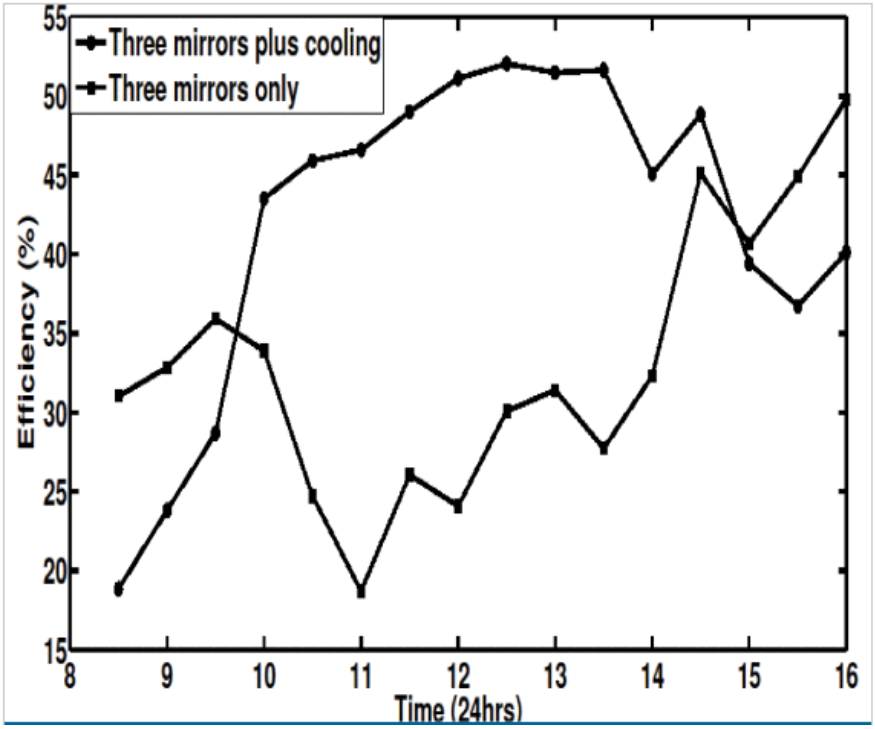

Figure 1.5: Improvement in solar efficiency for CPV system.

1.2.2 Solar Concentration

Another solution to inefficiency of PV systems is solar concentration with simple mirrors and cooling. Concentrated photovoltaic technology (CPV) revolves around optical lenses and mirrors to focus beams of sunlight straight onto the PV [5]. Similar to the solar tracking system, this increases the concentration of solar energy applied to the PVs. One advantage of CPV is that it does not require as many solar cells as non-concentrated systems to produce the same power output. This is because it increases the amount of energy applied to a single cell [5]. Another problem that CPV addresses is temperature. The heat from the sun is a major factor in affecting the operational capacity of PVs. High temperatures decrease the power output of PVs and this is a significant problem. CPV attempts to fix this issue by implementing mirrors and cooling systems. Figure 1.5 shows a graph of the improvement in efficiency when using three mirrors without cooling and three mirrors with cooling throughout a 24 hour period. During hours 12-13, the efficiency of the solar cells is maximum with around 50% [5]. These reflectors are cheap and easy to handle. As long as the temperature of the solar cells are kept down by the cooling mechanisms, the increase of concentration of sunlight on the PV by the CPV system can operate efficiently. Despite these advantages, CPV is extremely costly for installation and maintenance is expensive compared to traditional PV systems. Additionally, CPV systems can only be used in high direct normal irradiance environments, limiting its range of ability. Direct normal irradiance is the amount of solar energy received by the surface of the panel when it is directly held perpendicularly to the rays of sunlight. So keeping this perpendicular state and straight lines of rays is essential to CPV and misses all the other times when the panel is not directly perpendicular [5].

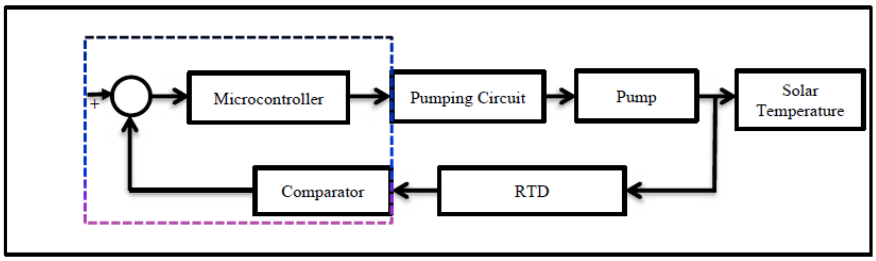

Figure 1.6: Block diagram schematic for water immersion method.

1.2.3 Water Immersion Method

Similar to the CPV, the cooling of solar panel cells is critical to improving and maintaining efficiency. The water immersion method involves submerging the photovoltaic system into distilled water at various degrees of depth [6]. Figure 1.6 shows a typical block schematic for a water immersion of a solar panel. There are several different components including an electronic resistance temperature detector (RTD) which senses the temperature of the solar panel. The figure also shows a microcontroller, pumping circuit, comparator, and pump. This system is attached to the top of the PV and releases water to cool it [6]. The comparator is used to track if the required temperature is exceeded. If this temperature is exceeded, the microcontroller directs the pump circuit to release water until the proper temperature is achieved. Because the solar panel only converts less than 20% of the irradiance into electrical energy, most of it is translated into heat within the cell [6]. As temperature of the cell increases, the efficiency decreases; therefore, the water immersion method seeks to decrease this temperature and increase efficiency. Typical losses from heat vary for each cell, but this number averages around 0.5%. The water immersion method has shown to increase PV efficiency with increasing water depth, the maximum being an 11% increase at a water depth of 6 cm. This method also improves efficiency by reducing the reflection of light because the refraction index within the water is lower [6]. It also leads to an absence of thermal drift, which is a change in the operation of the solar panels due to a sudden change in temperature.

1.3 Machine Learning for PV Outcomes

Although the three above mentioned methods have been shown to successfully improve solar panel function, there is one method that may surpass them all in efficiency. This method involves artificial intelligence (AI), a promising and intriguing field that will certainly play a key role in technological development. Specifically within the AI realm, this paper will specifically focus on machine learning (ML), which is a sub-field of AI. ML focuses on the use of algorithms and data to predict outcomes and mimic the way that humans learn, overtime improving its accuracy of said predictions [7].

When it comes to PV systems, ML can be used to monitor for possible faults in operation. If a fault occurs, due to weather, temperature, or some other factor, the quickness of the detection of the problem is vital to maintaining operation of the system and the production of power [8]. Figure 1.7 shows an example of a common issue that can affect solar cell efficiency: dirtiness. Sand or mud can cover up the solar cells and lead to loss of energy of capture. Additionally, solar farms can be

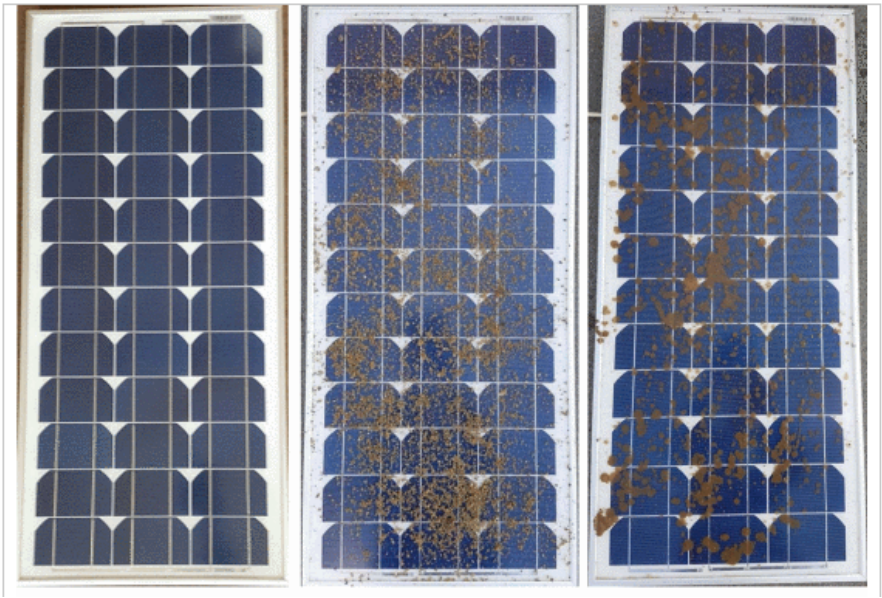

Figure 1.7: Photovoltaic shown in three different ways: clean, sandy, muddy (from left to right).

quite large; this mass of scale can lead to more lag in fixing the fault. These two examples show where ML techniques can be extremely helpful. ML can predict when said faults will occur and inform workers to preemptively fix this problem [8]. If the data looks wrong or askew from mud, the system can inform the worker to clean it. Or if there is an issue at a far end of the solar farm, the use of ML techniques can tell this disruption to a worker to go fix the issue. Additionally, patterns can be identified of the use of solar energy for distribution systems, helping power companies to adjust the power load. With greater development of AI systems, the cost of solar panel installation and maintenance may see a decrease as well. With more powerful and detailed models, efficiency of PV systems may see a rapid advancement of information analysis [8].

There are several models that exist to predict the outcome of solar panels using ML techniques. Because the ultimate goal of ML is to imitate behavior, it can be used to perform complex tasks. The objective of machine learning is to predict more accurate outcomes based upon collected data [9]. This sample data (training data) is used to make decisions based upon the best possible outcome. Developing systems to predict outcomes can help to model and design technologies, for further efficiency and dynamics [8]. Figure 1.8 is a flowchart of the photovoltaic system with active ML learning, monitoring, and analysis. As seen in the figure, the solar PV is connected to the typical hardware units, including inverter, storage, power distribution network, and utility. These allow for power distribution; however, one difference when it comes to ML is the alternative connections to the PV. This mainly includes the use of the Internet for external data. The use of the internet provides a better baseline data for the algorithm to create predictions. Machine learning is a field with exciting applications. It can be used for a variety of purposes including, medicine, agriculture, computers, and in the case of this thesis, renewable energy [9].

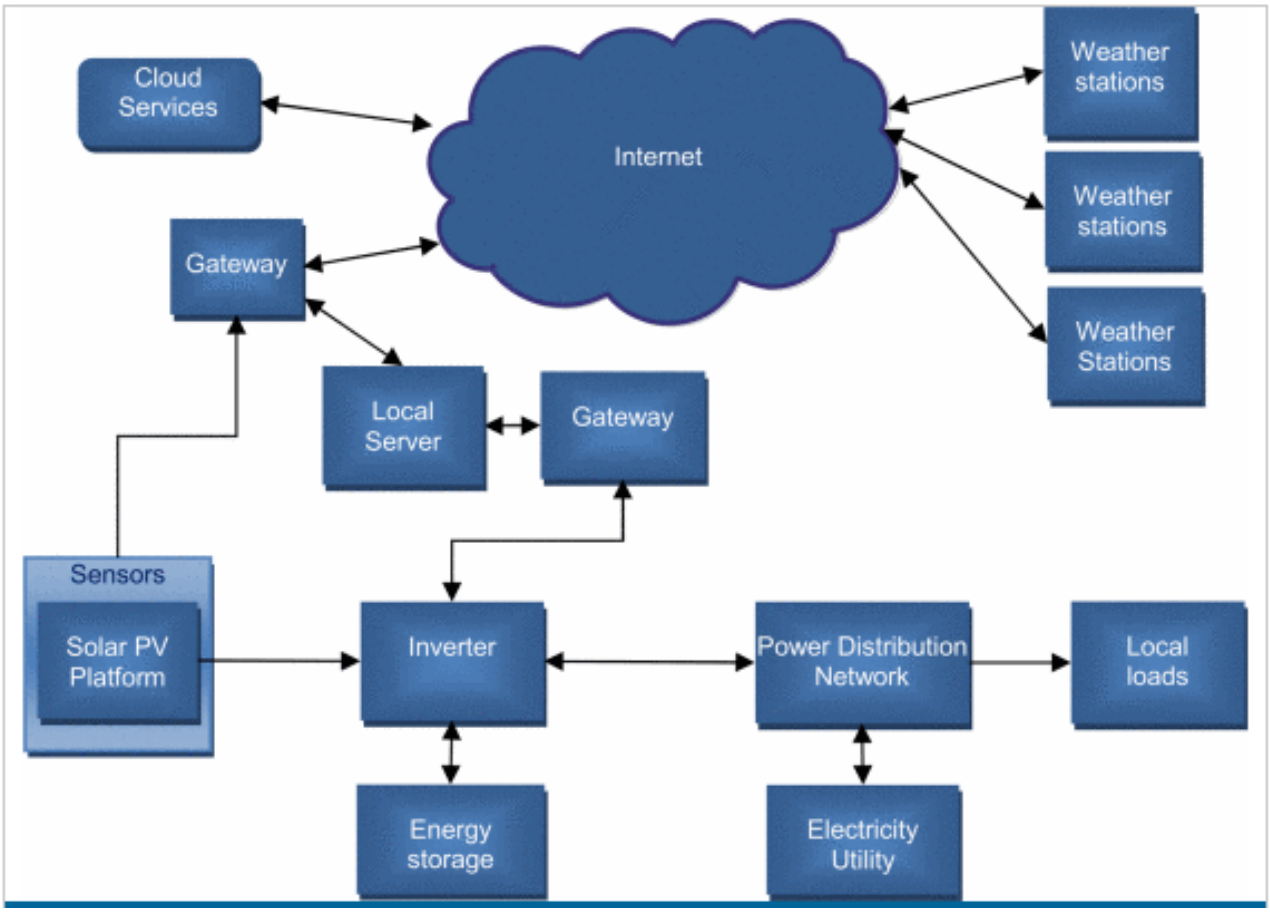

Figure 1.8: Flowchart of PV system with machine learning integration.

Chapter 2

Next Generation Reservoir Computing

2.1 Machine Learning Techniques

To understand the applications for PV systems,it is essential to recognize how machine learning works and can be used. Over the past two decades, ML has seen significant progress in development. ML is the choice branch of AI to deal with computer and software development because the research has shown that it is easier to train a computer by providing it examples of desired behavior rather than manually programming it [7]. Many ML algorithms are so widely developed that they can cover a wide range of data and problem types that a system could possibly encounter. ”Conceptually, machine-learning algorithms can be viewed as searching through a large space of candidate programs, guided by training experience, to find a program that optimizes the performance metric” [10]. This means that ML algorithms differ greatly from each other because they have to represent a large array of systems. Algorithms have to sift through two categories to accurately predict data outcomes: representative candidate programs(mathematical functions, programming languages, decision tree diagrams), and the space of programs, which means how the algorithm is optimized with well developed search methods [10]. The learning algorithm of machine learning can be broken up into three parts as shown in Figure 2.1.

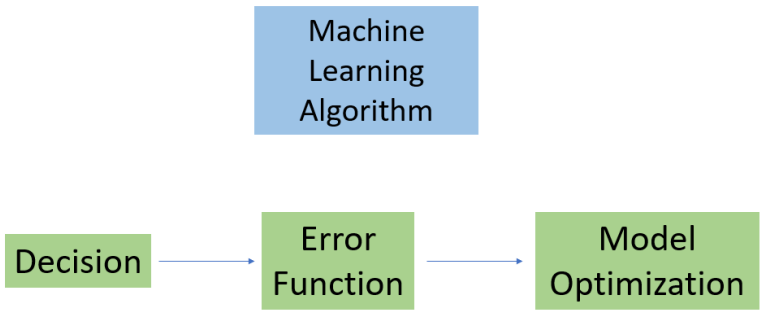

Figure 2.1: Flowchart for machine learning process.

In part one, the decision, given an inputted data, the algorithm can create an estimated pattern from the data. Next, the error function, the prediction from the decision process is evaluated. Given any provided examples, the machine can compare and contrast the data and assess for any possible errors. Finally, the model optimization, the machine checks to see if any adjustments can be made. This includes determining if the model fits better to the training data. This data is adjusted to compensate for any inconsistencies between the estimate and the given. This process will be repeated in the form of ”evaluate and optimize” [10], updating the outcome as time progresses. This will occur until the accuracy threshold is achieved [11].

2.2 Reservoir Computing

One type of machine learning is called reservoir computing (RC). This algorithm is a highly advanced, best-in-class system that processes information created by dynamical systems from observed discrete data. RC is relatively easy to use as it requires a small amount of training data and

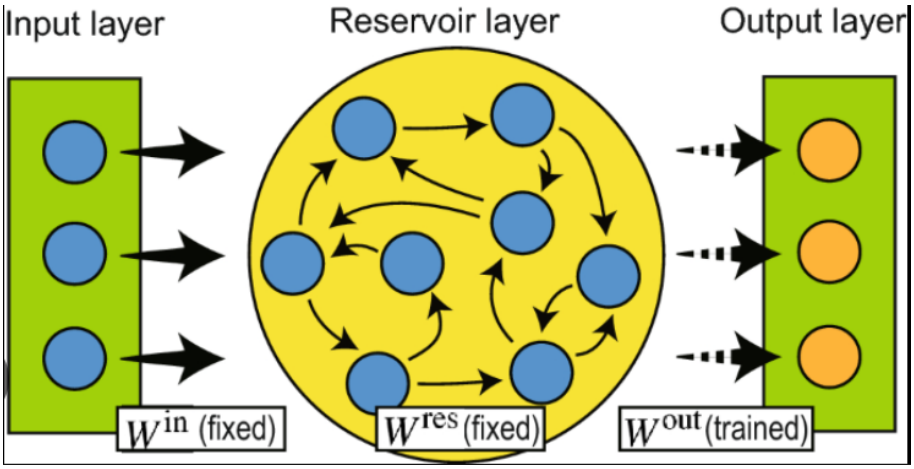

Figure 2.2: Reservoir computing flow chart.

uses linear optimization [11]. Figure 2.2 shows a simplified diagram of an RC system. Here there is an input, reservoir, and output layer. Each layer consists of its own algorithm, whereas the major computational work occurs in the reservoir layer. The input layer typically consists of the training data, and the output layer is the predictions. RC is particularly well suited for dynamic machine learning; it can handle systems with complex problems when efficiently optimized. However, despite these advantages, there are some drawbacks to reservoir computing. For example, reservoir computing uses a set of randomly sampled matrices which define the underlying network, ultimately leading to more optimization of certain parameters. Therefore, the next generation of reservoir computing (NGRC) is now gaining more traction, as it consists of nonlinear vector auto-regression; this requires no random matrices, produces results equivalent to reservoir computing, and has less parameters [11]. Another key advantage, specifically for machine learning, from next generation reservoir computing, is that the training data required for producing accurate results is even smaller than just reservoir computing. The development of models is necessary to forecast behavior. Recent advancements made within the ML paradigm have generated algorithms that require a lot of data; this can be an issue when it comes to complexity of the system [11].

An RC is composed of a reservoir which feeds data into the network and an output layer displaying the state of the network. Figure 2.3 shows this arrangement. In the figure, both the processes for an RC and NGRC are displayed. Focusing first on the RC, the sampled data is inputted into a matrix and reservoir, ultimately displaying a forecasted dynamic and predicted outcome.

However, despite these advantages, there are some drawbacks to reservoir computing. For example, reservoir computing uses a set of randomly sampled matrices which define the underlying network, ultimately leading to more optimization of certain parameters. Therefore, the next generation of reservoir computing is now gaining more traction, as it consists of nonlinear vector auto-regression (NVAR); as mentioned earlier, the use of this NVAR leads to no requirements for random matrices, produces results equivalent to reservoir computing, and has less parameters for optimization [11]. Another key advantage, specifically for machine learning, from next generation reservoir computing, is that the training data required for producing accurate results is even smaller than just reservoir computing. This fact proves that a NVAR system can exist and perform as well as an RC optimized system; this suggests that within an NVAR, RC is implicitly defined. As a

result, it is generally simple to create an NVAR system for three of the most common RC challenges: predicting short-term dynamics, reproducing the long-term outcomes of a chaotic system, and interpreting any unseen data within a dynamic system [11]. NVAR is used within the NGRC system, giving it these advantages to overcome those three key issues found within RC. It is essential to have forecasting with high accuracy and to not overlook the data that is unseen. The NVAR makes up for these issues by using extremely specific data sets and overcomes any difficulties with parameters that arise from the installation of an RC system. Recent advancements have focused upon the next generation of reservoir computing (NGRC) to make up for any disadvantages. The NGRC is more advanced than a traditional RC system because it incorporates an implicit RC and is much more efficient than RC.

It is essential to truly understand the diagrams within Figure 2.3 because it shows the process and differences of both RC and NGRC systems. In the top of Figure 2.3, the process for the traditional RC is shown. This is times-series data that is associated with a strange attractor [11]. A strange attractor revolves around algorithms. It is the state of a mathematical system that is chaotic, or moves to disorder, instead of towards the trends. This strange attractor is indicated by the blue graph located within the x,y, and z planes in the left/middle of Figure 2.3. The RC network uses an artificial recurrent neural network, which is a type of network that is commonly utilized with time series data [11]. This is important because time series systems take measurements at increments of time for a certain period. The predicted strange attractor, as indicated by the red 3D graph on the right/middle, shows the weight of all the reservoir states in a linear relationship. The ground truth dynamics indicates the data that is being input into the algorithm as training data, to teach the algorithm how to make predictions based upon past principles; the data goes through the RC system is displayed as the forecasted dynamics, or expected outcome [11]. The actual RC system will be explored further in more detail later in this paper.

Also in Figure 2.3 is the diagram for the NGRC system. One difference between NGRC and RC is that NGRC makes predictions based upon a linear weight of the delayed time states of the time-series data as well as the nonlinear portions of the data [11]. This means that the NGRC is using more input data points, as indicated by the extra dots on the ground-truth dynamics graph for the NGRC system. Like the RC system, this data is fed through the NGRC algorithm and outputted as the forecasted dynamics, with an extra data point, with comparison to the traditional RC system [11]. This extra data and more advanced system makes the NGRC system more powerful than RC. The actual NGRC system will also be explored further in more detail later in this paper.

As shown in Figure 2.3, RC systems work through the following process: inputted data X is broadcasted into the next dimension of the reservoir network, which consists of N nodes connected internally. These two are connected to an output, Y, as a reservoir state, which is very similar to the desired output Y_d [11]. Matrix A shows the node-to-node connections and their strength; these are randomly chosen and kept fixed. The input data, X is processed through the reservoir of an input layer that has an assortment of fixed coefficients, W, in Figure 2.3 [11]. The equation of the reservoir system is represented by

$$r_{i+1} = (1-\gamma)r_i + \gamma f(Ar_i + WX_i + b) \tag{2.1}$$

where r_i is an vector with N-dimensions, or $r_i = [r_{1,i}, r_{2,i}, ..., r_{N,j}]^T$. In this vector is the component r_{ji}, which, at time t, represents the state of the jth node. i is the ith time step. γ represents the nodal rate of decay and is the same for every node, f represents the applied activation function

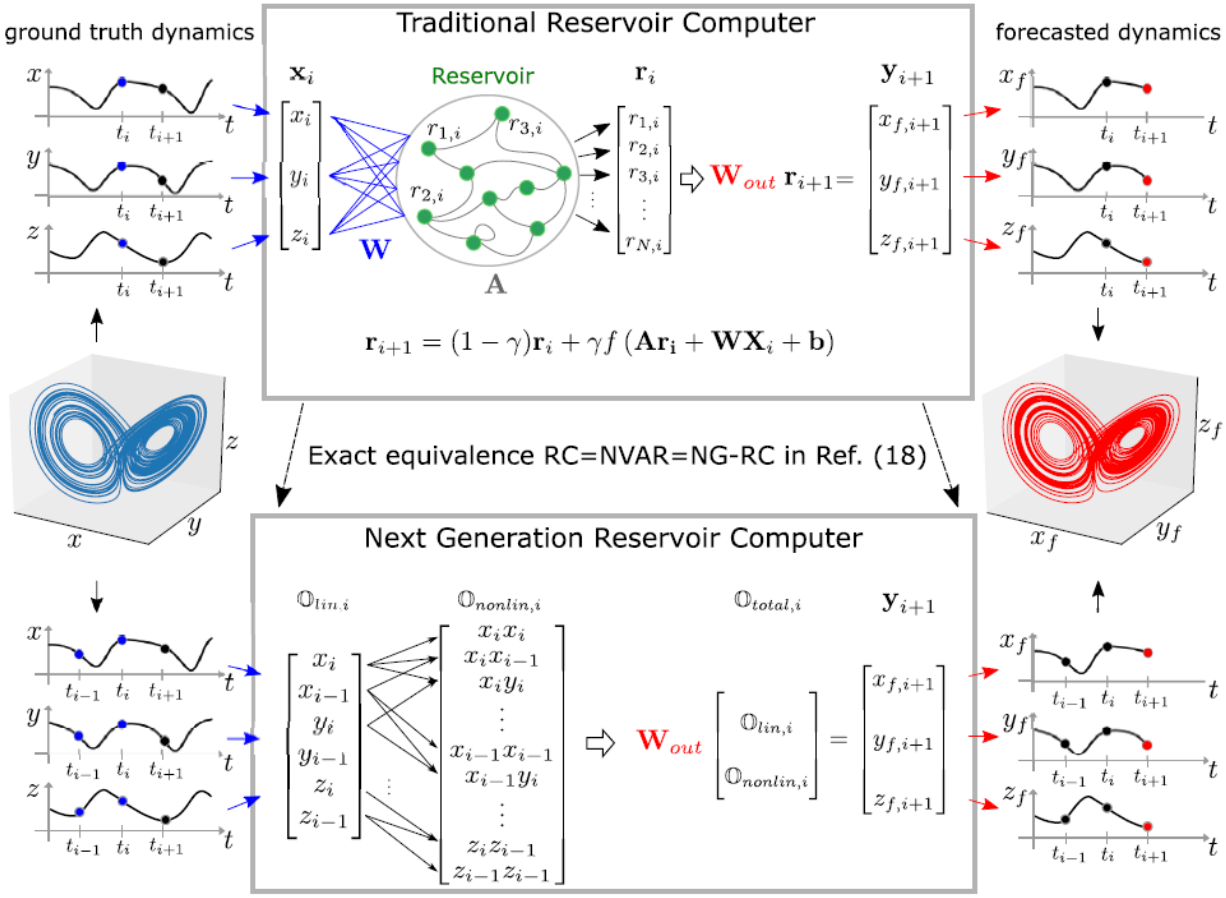

Figure 2.3: (top) Traditional RC process and (bottom) New NGRC process.

that is given to each component of the vector, and b is the node bias vector and is also the same for every node. Time is discrete at sample dt, where $dt = t_{i+1} - t_i$. When r is incremented by r_{i+1}, this implies consecutive time steps within the reservoir state [11].

As for the output expressed through the RC, this can be seen as the equation Y_{i+1}. This output is expressed as a linear transformation of the feature vector $\mathbb{O}_{total,i+1}$. This vector is constructed from the aforementioned reservoir state through the following equation:

$$Y_{i+1} = W_{out} + \mathbb{O}_{total,i+1} \tag{2.2}$$

Here, W_{out} is the weight of the output matrix. The subscript 'out' implies that the variable can be composed of either non-linear, linear, or constant terms. Typically, in an RC system, the terms for the nodes would be chosen to be in a non-linear form. The output layer is represented by

$$\mathbb{O}_{total,i+1} = \mathbb{O}_{lin,j+1} = r_{i+1} \tag{2.3}$$

And for the output layer, a linear feature vector would be chosen. Using a concept called regularized least-squares regression, the RC is trained and supervised. This process generates training data points and matches Y to the desired Y_d output, using the Tikhonov regularization (a strategy to regularize ill-formed data) to get the following equation for W_{out}:

$$W_{out} = Y_d {\mathbb{O}_{total}}^T (\mathbb{O}_{total} {\mathbb{O}_{total}}^T + \alpha I)^{-1} \tag{2.4}$$

Here, I is the identity matrix and α is the regularization parameter and is used to prevent the over fitting of the sampling data.

Another approach to RC is to change the location of the nonlinear data set from the reservoir to the output layer of the system. When this is the case, the nodes are given the function $f(r) = r$, which induces linear activation [11]. Because of this, the feature vector $\mathbb{O}_{total}$ becomes nonlinear. This function is given below.

$$\mathbb{O}_{total} = r \oplus (r \odot r) = [r_1, r_2, ..., r_N, r_1^2, r_2^2, ...r_N^2]^T \tag{2.5}$$

In equation 2.5, the Hadamard product is given by

$$(r \odot r) = [{r_1}^2, {r_2}^2, ..., r_N]^T \tag{2.6}$$

In equation 2.5, $\oplus$ represents the operation of vector concatenation. This alternative approach to the traditional RC approach is an equivalent method with a linear reservoir with a nonlinear output.

On the other hand, the NGRC system also has a vector that comes directly from the inputted sample data with no requirement for a neural network, where $\mathbb{O} = c \oplus \mathbb{O}_{;on} \mathbb{O}_{nonlin}$. Here, c is a constant and $\mathbb{O}_{nonlin}$ is the nonlinear part of the vector. Similar to the RC system, for the NGRC system, the output is gathered from the same components found in equation 2.4. $\mathbb{O}_{lin}$ is a set of observations of input vector X spaced apart by the previous $k-1$ steps from consecutive data observations. When $X_i = [x_{1,i}, x_{2,i}, ..., x_{d,i}]^T$, $\mathbb{O}_{lin,i}$ can be expressed by

$$\mathbb{O}_{lin,i} = X_i \oplus X_{i-s} \oplus X_{i-2s} \oplus ... \oplus X_{i-(k-1)s} \tag{2.7}$$

where s is the space of the time steps. K is taken to be relatively large due to the theory of universal approximation, which states that networks can approximate functions by increasing width.

One important advantage of the NGRC as opposed to the traditional RC system is that it has a warm up period that only takes a specific amount of time steps (sk) needed to create the necessary vector for processing [11]. The RC system takes much longer to warm up. A shorter warm up time is even more important when there it is particularly difficult to obtain data. For example, set $s = 1$ and $k = 2$, and only two warm up data points are required. In a traditional RC system, a common warm-up time can be within the upper range of 10^3 to 10^5 data points, which is substantially higher than the NGRC's two data points [11].

$\mathbb{O}_{nonlin}$, as the subscript suggests, is the nonlinear portion of $\mathbb{O}_{lin}$. It is most typical and practiced to use polynomials for the nonlinear function as they provide solid predictability, specifically low order polynomials [11]. An example of such polynomial could be composed of $(dk)(dk+1)/2$ molynomials when $\mathbb{O}_{lin} \otimes \mathbb{O}_{lin}$. $\otimes$ can be defined as the collector of unique molynomials within a vector. Thus, the nonlinear portion of can be found by the following equation:

$$\mathbb{O}_{nonlinear} = \mathbb{O}_{lin} \lceil \otimes \rceil \mathbb{O}_{lin} \lceil \otimes \rceil ... \lceil \otimes \rceil \mathbb{O}_{lin} \tag{2.8}$$

Recent methodologies have proven that it has been shown that the linear RC model with a nonlinear output results in the same outcomes as the NVAR method; however, the RC method requires much more computational effort and requires parameter optimization. The NGRC method, which utilizes NVAR, is much simpler and more efficient. The NGRC does the same work as the RC network, just without much of the complicated computational efforts [11].

2.3 NGRC Performance Modeling

There are several models used for showing the performance of the NGRC system. The training data used in this model comes from a weather system model developed by Lorenz, represented by the equation,

$$\dot{x} = 10(y - x), \dot{y} = x(28 - z) - y, \dot{z} = xy - 8z/3 \tag{2.9}$$

Here, the state $X(t) \equiv [x(t), y(t), z(t)]^T$ is a vector of the Rayleigh-Benard convection, which establishes principles of chaos and sensitivity dependence to initial conditions. This system consists of three differential and coupled nonlinear equations. These equations show several characteristics, including sensitivity of the system to initial conditions [11].

The NGRC is used to predict the behavior of a double-scroll electrical circuit through the following equations

$$\dot{V}_1 = V_1/R_1 - \Delta V/R_2 - 2I_r sinh(\beta \Delta V), \tag{2.10}$$

$$\dot{V}_2 = \Delta V/R_2 + 2I_r sinh(\beta \Delta V) - I, \tag{2.11}$$

$$\dot{I} = V_2 - R_4 I \tag{2.12}$$

where $\Delta V = V_1 - V_2, R_1 = 1.2, R = 3.44, R_4 = 0.193, \beta = 11.6$ and $I_r = 2.25 * 10^{-5}$. These numbers were specifically chosen to satisfy the Lyapunov time condition. This means that the vector field is not in polynomial form and it proves that ΔV will be large enough to ensure that the series expansion of the system will provide large differences within the expected attractor. To use the NGRC system, a listening phase is created to achieve the outcome $X(t + dt) = W_{out}\mathbb{O}_{total}(t)$. From this outcome, the NGRC system is finally an autonomous and dynamical system which successfully predicts the system's outcome because the predicted outcome is fed back to the input, updating the training data [11].

The double scroll system has an odd symmetry and a zero mean for every parameter. These are specific characteristics that are represented by

$$\mathbb{O}_{total} = \mathbb{O}_{lin} \oplus \mathbb{O}_{nonlinear} \tag{2.13}$$

The NGRC adapts simultaneously with its new knowledge from the vector and instantiates an integrator that is one step ahead of the data to create accurate predictions, creating a mapping of the data. This is helpful because it allows for next steps predictions without having to learn each vector field separately, also known as a flow dynamical system [11]. The NGRC learns the flow of the data from this system. A Euler integration is used to teach the NGRC the simpler and smaller steps of this integration by modifying Eq. 2.2 to get

$$X_{i+1} = X_i + W_{out}\mathbb{O}_{total,i} \tag{2.14}$$

The double scroll NVAR is an attractor made of 4 linear circuit elements - 2 capacitors, 1 inductor, and 1 resistor [12]. This circuit can be applied to work as non polynomial vector field instead of a polynomial form.

In the methods used to predict outcomes, the NGRC anticipates the dynamics of the Lorentz and the double scroll systems. During the testing phase, the components of the input are not fed back to the NGRC and the predicted output is provided back to the input. This is essential because it indicates that the NGRC is an autonomous and dynamical system which can predict the system's outputs [11].

One key aspect of the NGRC is that it simultaneously learns both the vector field and the mapping from one state to the next as an integrator that is one-step-ahead [11]. This ultimately means that NGRC is learning the flow of the dynamical system. To ensure that the NGRC can still focus on the smaller and more subtle details of the flow, a Euler-like step for integration is included as a low order approximation. This allows the NGRC to learn the difference between the state of the order, i.e. a current or future step.

As seen in Eq. 2.14, there are three variables account for, x,y and z. However, during testing of this Lorenz equation, only x and y are used and the variable z is inferred. This is for when it is possible to obtain quality data in a laboratory setting, but not in an actual real world environment. In a field setting, the sensory information is used to infer the missing data (z) [11].

2.4 Results

The Lorenz attractor is shown in Figure 2.4; this shows the forecasting of the dynamical system using the NGRC system. This figure shows the true Lorenz strange attractor, or the reference. For

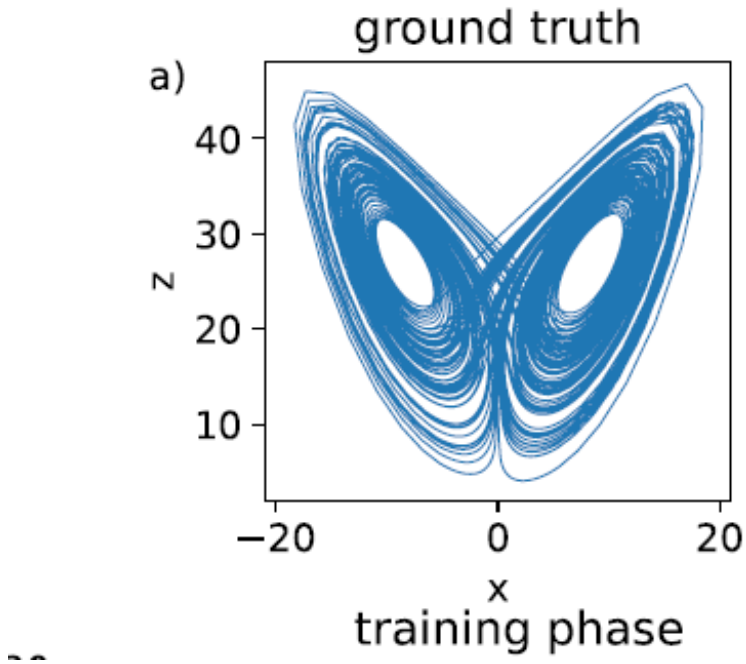

Figure 2.4: True Lorenz strange attractor.

training, there is a specific array of data used, graphs b-d in Figure 2.5. These graphs show the training data with predicted behavior overlapped. This figure also shows the various phases of data prediction. Each graph consists of 400 data points with a time interval (dt) of 0.025, where $k = 2$ and $s = 1$. Additionally, $\alpha = 2.5 * 10^{-6}$ [11]. The normalized root mean square error (NRMSE) which compares differences between various number of models, is $1.06 \pm 0.01 * 10^{-4}$. These numbers were chosen, as alluded to before earlier in this paper, for optimization. The graphs b-d are all within the training phase with a computational time of less than 10 ms. Running on a Python based system, the values for $\mathbb{O}_{total}$ and W_{out} are 28 and $3x28$, respectively [11]. This dimension is necessary to ensure that the trajectory is large enough to examine both sides of the graph in Figure 2.4.

After these ground truths are established, the prediction phase is next. The NGRC is moved to the prediction phase. Figure 2.6 shows the NGRC outcome for the predicted data. The graphs of both Figure 2.4, the basis, and Figure 2.6, the prediction are almost identical. This proves that in the long run, the NGRC replicates the Lorenz system. The NGRC system's ability to make these predictions indicates that it is a useful algorithm for increasing a machine's efficiency [11]. Figure 2.7 shows the predicted data of the NGRC system in red, overlapping the true data in blue in graphs f-h. As seen from these graphs, the NGRC almost identically imitates the true data during the forecasting phase. Here, the NRMSE is $2.40 \pm 0.53 * 10^{-3}$. This data is comparable to a traditional optimized RC system, once again proving that NGRC is more efficient and effective for prediction models [11].

After examining the dynamical NGRC system, the next phase is the double-scroll system. For the double-scroll, the Lyapunov is used instead of the Lorenz63 system. Therefore, the training time for the NGRC is extended from 10 to 100 units. After this, the NGRC is put through a

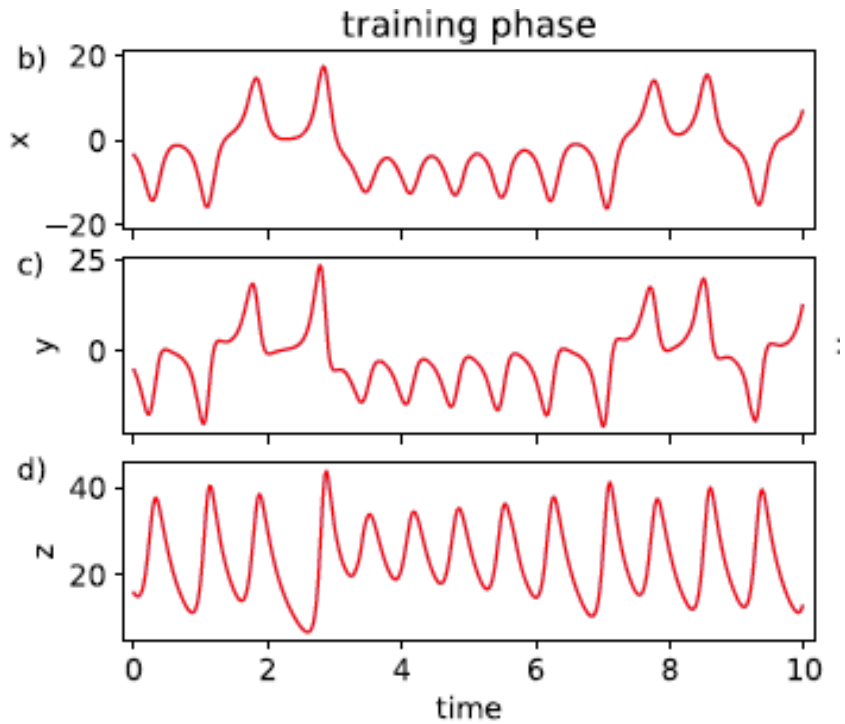

Figure 2.5: Training set of data for NGRC.

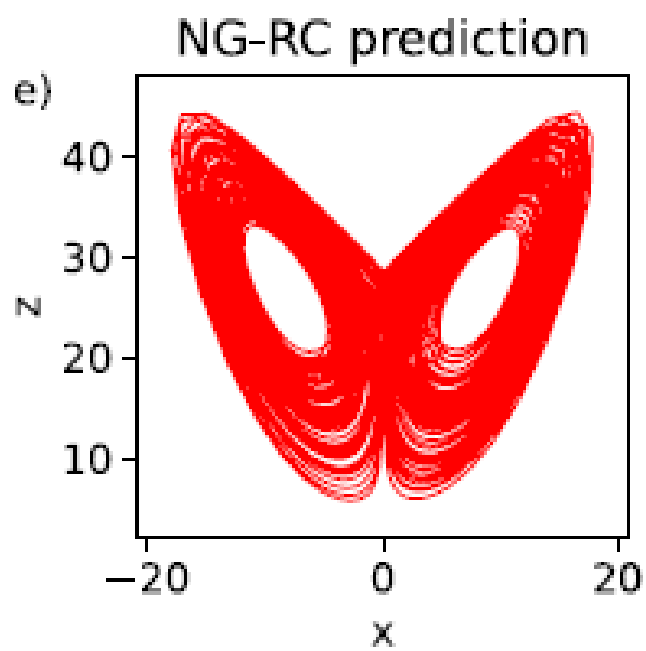

Figure 2.6: Testing data set for NGRC Lorenz strange attractor.

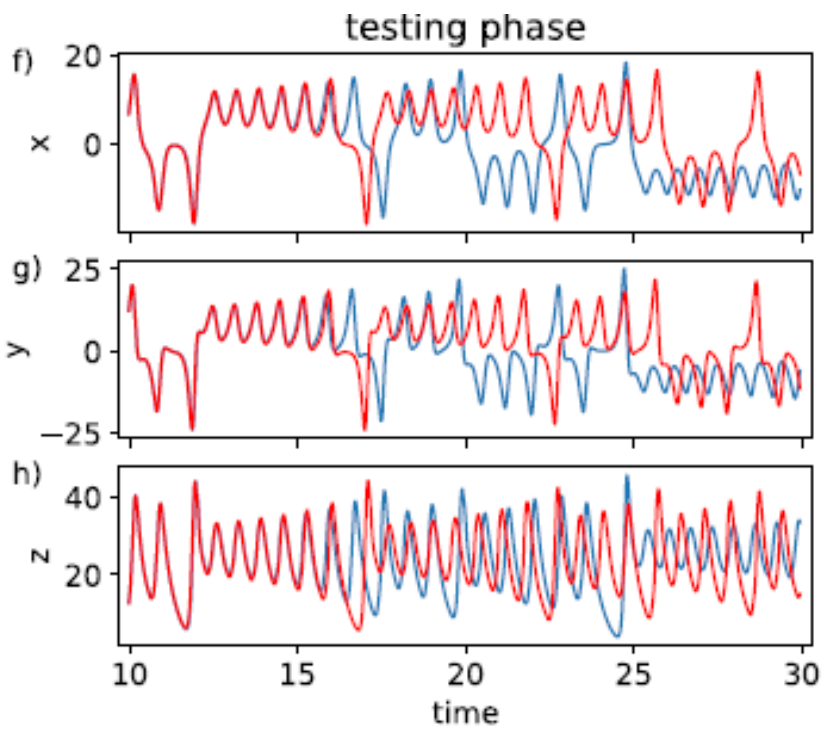

Figure 2.7: Predicted data outcomes for Lorenz strange attractor.

prediction phase, in which it can inspect the quality of the data predicted. One timing note is that the Lyapunov system for the double scroll is much longer in duration than the Lorenz system [11]; thus, the the training time of the NGRC system is lengthened to ensure that this is not an issue. This leads to an equal comparison time of both cases. In the double scroll system, the NGRC system uses 400 data points, similar to the Lorenz system. The timing interval is set to 0.25, and $d = 3, k = 2$, and $s = 1$ to make up for this timing difference. This leads to a total of 62 components in $\mathbb{O}_{total}$. The results of this can be seen in Figure 2.12, showing that ultimately the NGRC system can predict similar outcomes to the double scroll system within the Lorenz system [11].

Finally, the last component of the system is to test and infer the dynamics that were not explicitly shown in the NGRC within the testing phase described above. The variables x,y, and z are provided during training and then a short set of training data is observed of 400 points. This is a good performance and can be seen in Figure 2.7, where the training data is compared to the NGRC data predictions. This inference of data is specifically for the z component, as x and y are provided to the NGRC system during the testing phase [11].

Besides the variable modifications as indicated above, the forecasting of the double scroll system is almost identical to the Lorenz63 system. These results can be seen in the Figures 2.8 - 2.12. As seen from these graphs, the results are pretty identical to the Lorenz63 results. There are differences in the true and predicted strange attractor graphs for the Lorenz and the double scroll; but as seen in Figures 2.8 and 2.10, the true and predicted outcomes for the strange attractor are similar for the double scroll [11].

The last major component of this testing system is to predict any data or dynamics that are not explicitly shown in the NGRC system. In this example, $k = 4$, $s = 5$, and $dt = 0.05$. These param-

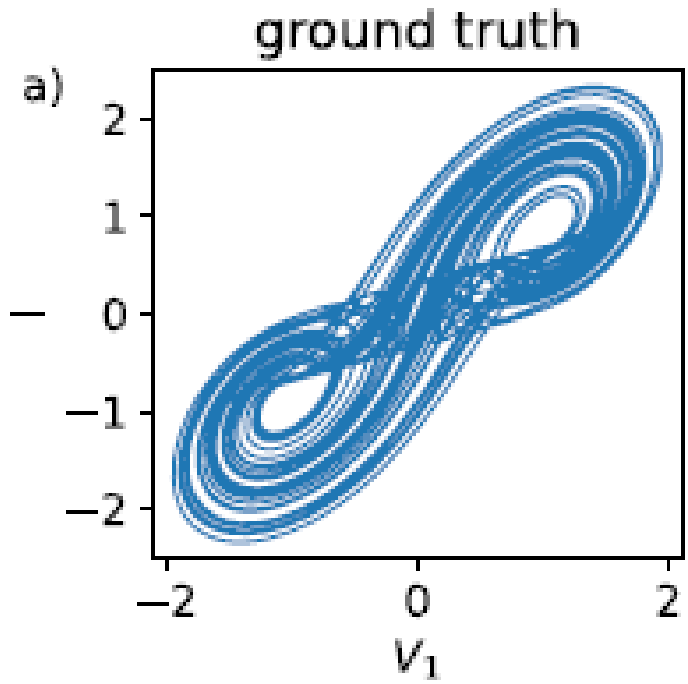

Figure 2.8: True double-scroll strange attractor.

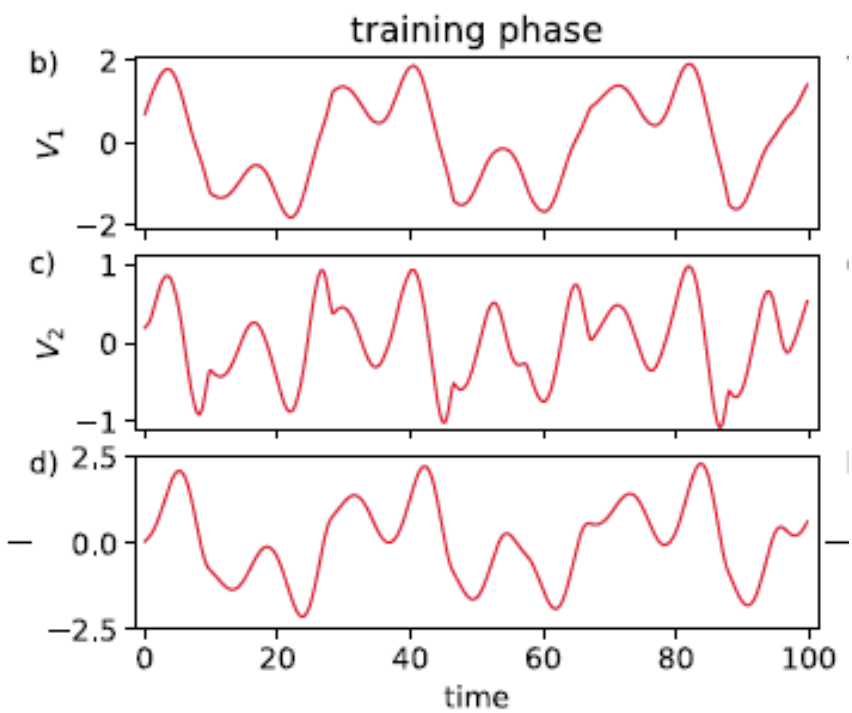

Figure 2.9: Training set of data for NGRC for double scroll.

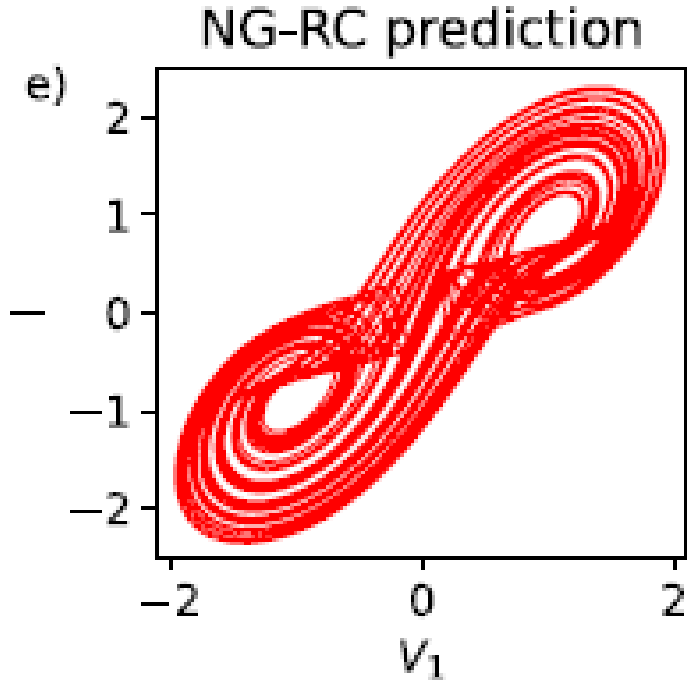

Figure 2.10: Predicted data set for double scroll strange attractor.

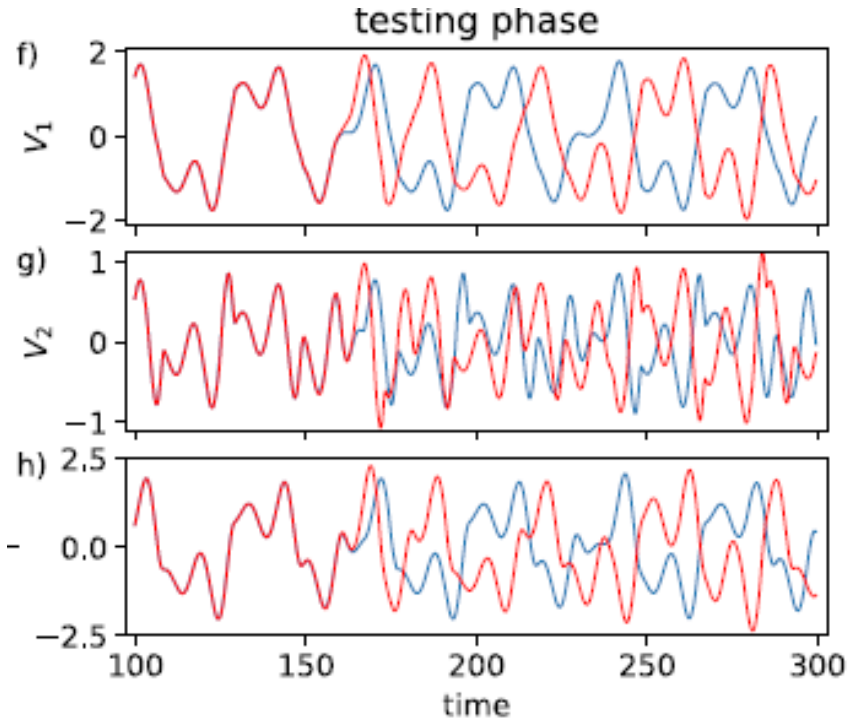

Figure 2.11: Predicted data outcomes for double scroll strange attractor.

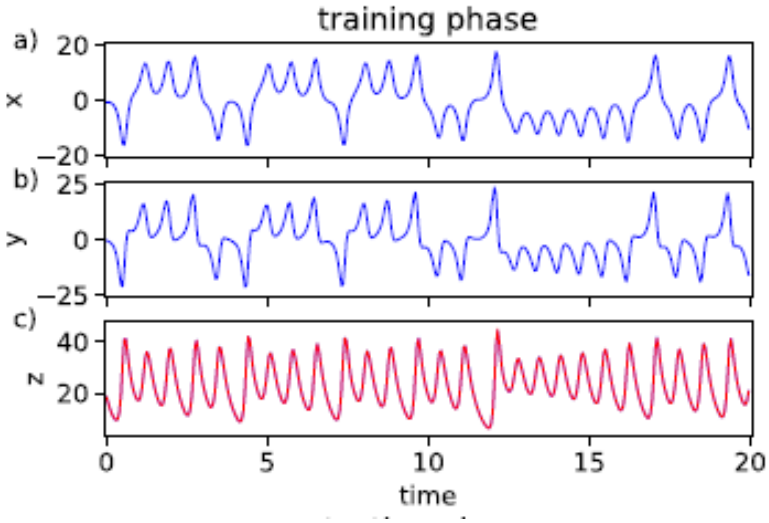

Figure 2.12: Training data for inferred dynamics.

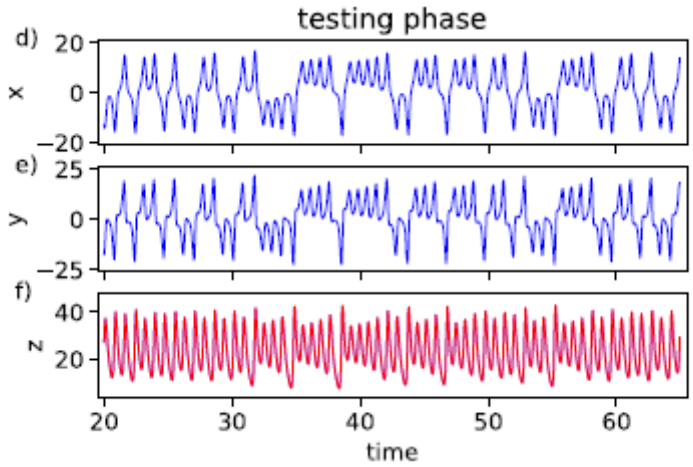

Figure 2.13: Predicted data outcomes for inferred dynamics.

eters are set to ensure that a full attractor is generated for comparison to other components [11]. Once again, the variables x,y, and z are used for observation and the training data consists of 400 points. Figure 2.12 shows the initial data for these inferred dynamics. The training data is overlapped with the NGRC data [11].

Figure 2.13 shows the predicted data outcomes. And as seen in comparison to the previous figures established in the NGRC Lorenz and double scroll data sets, the predictions are almost identical in the testing phase.

Overall, the NGRC is much faster at computational efforts than a traditional RC method. This is due to several reasons but most importantly due to a smaller vector size. A smaller vector size allows for fewer parameters, which allows the algorithm to learn with a smaller training data set. This increases efficiency and lowers the learning curve for the AI. Additionally, the training time and the warm-up for the NGRC is significantly shorter than the traditional RC system, shortening computational efforts again [11]. Finally, the NGRC system does not have as many meta parame-

ters to optimize, again shortening the computational time. The NGRC is a product of the complex history of AI work. It comes from previous work on nonlinear systems. The NGRC system is closely related to a multiple input and output system with a nonlinear auto-regression, also known as NARX. The NGRC combines the best of the NARX methodology along with modern regression theories to create the successful outcomes described in this paper [11].

Chapter 3

Photovoltaic Modeling

3.1 System Modeling

To understand the conclusions that this thesis is testing for, it is necessary to understand modeling of photovoltaics. PVs generate DC power instead of AC power and a power electronic interface is then used to convert this DC to AC power. Figure 3.3 shows the mathematical model of a singular PV cell [13].

Using Kirchhoff's Current Law at the location of the red oval, an equation for the output current (I) can be established as the following:

$$I = I_{ph} - I_d - I_p \tag{3.1}$$

where I_{ph} is the photon current that depends upon the light intensity and its wavelength, I_d is the Shockley temperature-dependent diode current and I_p is the PV cell leakage current [13].

To determine the values of I_{ph}, I_d, and I_p, there are also several equations and parameters that must be established. I_{ph} is dependent on light intensity and wavelength and can be determined from the following equation,

$$I_{ph} = \frac{S}{S_{ref}}[I_{phref} + C_T(T - T_{ref})] \tag{3.2}$$

where S is the value of irradiance, S_{ref} is the irradiance reference value (typically 1000 W/m^2), C_T is the temperature coefficient, T is the temperature (in Kelvin), and T_{ref} is the temperature reference value (typically 298 K) [13].

As seen from this model, a feature of the photon current is that it shares a positively linear relationship with irradiance, meaning as I_{ph} increases, so does S. This is because of the principle of conservation of energy, as the kinetic energy of the electrons in the cell are linearly proportional to the frequency of the radiation of light. When the light that is being received is very weak, there is no output for the photovoltaic [13]. This intensity of the light equates to the number of photons available. To produce the necessary electrons, the highest wavelength for when the photon energy is still large enough is 1.15 μ. The kinetic energy is also independent of the intensity of the radiation [13]. Figure 3.1 shows the relationship between kinetic energy and frequency and Figure 3.2 shows the relationship between kinetic energy and light intensity. Frequency and kinetic energy share a linear relationship, while kinetic energy and light intensity are independent variables [13]. Similarly, as temperature increases I_{ph} also increases due to the relationship in equation 3.2.

To further develop the model shown in Figure 3.3, it is necessary to include the addition of the Shockley diode, which is a semiconductor that helps control the system. In Figure 3.3 the diode is added to the original diagram in parallel to the other components. The Shockley temperature-dependent diode current can be obtained from the equation,

$$I_d = I_s(e^{\frac{q(U+IR_s)}{\eta kT}} - 1) = I_s(e^{\frac{q(U+IR_s)}{\eta V_T}} - 1) \tag{3.3}$$

where I_s is the diode reverse saturated current (typically 100 pA), k is the Boltzmann constant ($1.38047 * 10^{-23}\frac{J}{K}$), q is the electron charge ($1.60201 * 10^{-19}C$), η is the empirical ideal constant for silicon (around 1.2 and 1.8), and V_T is the equivalent temperature where, $V_T = \frac{kT}{q}$.

I_s can be found from the following equation,

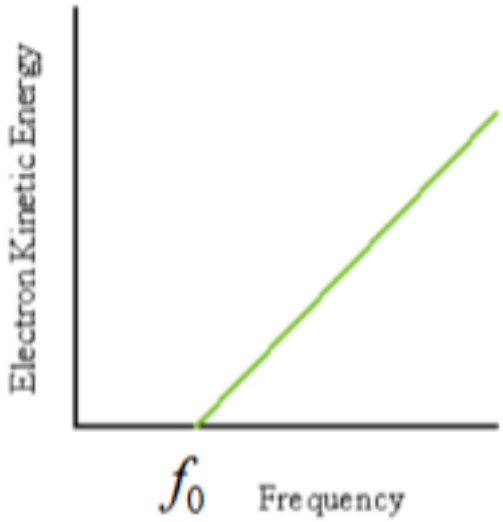

Figure 3.1: Graphical relationships between kinetic energy and frequency.

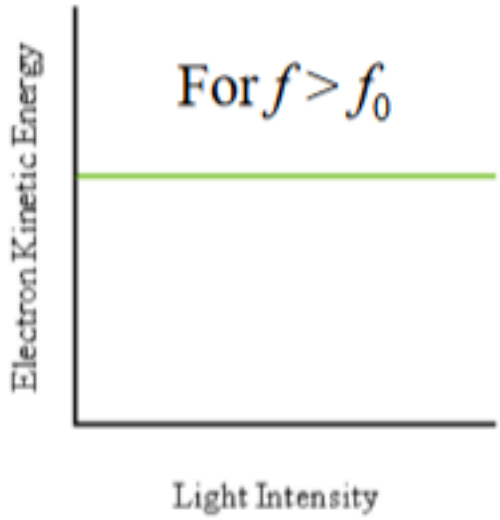

Figure 3.2: Graphical relationships between kinetic energy and light intensity.

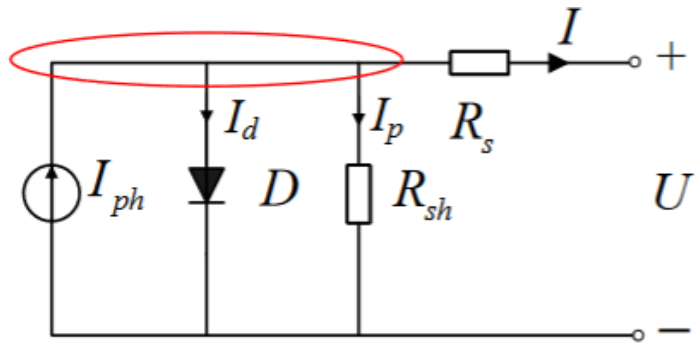

Figure 3.3: Mathematical Model of PV Cell

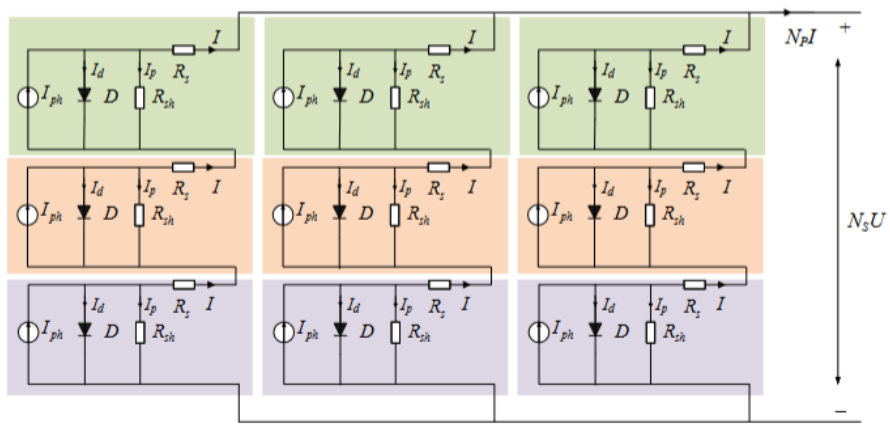

Figure 3.4: Series and parallel connections of singular PV cells to form entire solar panel

$$I_s = I_{sref}(\frac{T}{T_{ref}})^3 e^{\frac{qE_g}{\eta k}(\frac{1}{T_{ref}-\frac{1}{T}})} \tag{3.4}$$

The last component of the output current equation is the PV cell leakage current, I_p, which can be modeled from the equation,

$$I_p = \frac{U + IR_s}{R_{sh}} \tag{3.5}$$

In an ideal case, I_p would be zero, but in the case of this thesis it will have a value.

Figure 3.3 shows the mathematical model of a singular PV cell. Each solar panel is composed of many PV cells, so to understand the model of the entire system, Figure 3.4 shows the mathematical model of the whole solar panel. Each solar cell is connected in series with the next; in the case of Figure 3.4 there are three cells connected together. The number of cells connected in series is represented by N_s. These three cells are then connected to two other series connected groups of three in parallel for a total of nine connected PV cells. The number of PV cells in parallel is represented by N_p.

Figure 3.5 is a generalized representation of this mathematical model, showing that the number of PV cells connected in series and parallel can vary for the size of the solar panel [13]. To determine the mathematical model of the PV array the following equation can be applied:

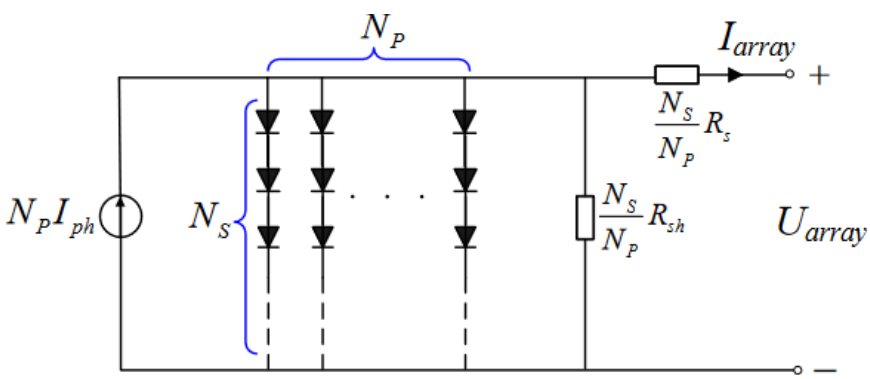

Figure 3.5: Generalized mathematical model of solar panel systems

$$I_{array} = I_{ph} - I_d - I_p \tag{3.6}$$

where

$$I_{ph} = N_p * I_{ph} \tag{3.7}$$

$$I_d = N_p I_s (e^{\frac{1}{\eta V_T}(\frac{U_{array}}{N_s} + \frac{I_{array}}{N_p} R_s)} - 1 \tag{3.8}$$

$$I_p = \frac{N_p}{R_{sh}}(\frac{U_{array}}{N_s} + \frac{I_{array} R_s}{N_p}) \tag{3.9}$$

When equations 3.7, 3.8,and 3.9 are plugged into equation 3.6, the resulting mathematical model for the PV array is

$$I_{array} = N_p I_{ph} - N_p I_s (e^{\frac{1}{\eta V_T}(\frac{U_{array}}{N_s} + \frac{I_{array}}{N_p} R_s)} - 1 - \frac{N_p}{R_{sh}}(\frac{U_{array}}{N_s} + \frac{I_{array} R_s}{N_p}) \tag{3.10}$$

Similar to a single PV cell, in an ideal case I_p will be zero in an ideal case.

The output characteristics of a solar panel can be determined by examining different circuit cases. When the voltage of the array (U_{array})is 0, the circuit is open and the characteristics of the output can be seen in Figures 3.6 and 3.7.Figure 3.6 shows the current/voltage relationship for several different irradiances (900 $\frac{W}{m^2}$, 1000 $\frac{W}{m^2}$, and 1100 $\frac{W}{m^2}$). Meanwhile, Figure 3.7 shows the active power/voltage relationship for those same irradiances. From these graphs it can be determined that as the irradiance increases, the power output of the photovoltaic array also increases.

It can be seen from these two graphs that as the irradiance increases, the current of the short circuit increases a significant amount. On the contrary, as the irradiance increases, the open circuit voltage also increases, but not as much or as rapidly as the short circuit current [13]. To represent this within a mathematical model, equation 3.10 is set equal to 0, to solve for U_{array} in the following way:

$$I_{array} = 0; U_{array} = \eta V_T N_s ln(\frac{I_{ph}}{I_s} + 1) \tag{3.11}$$

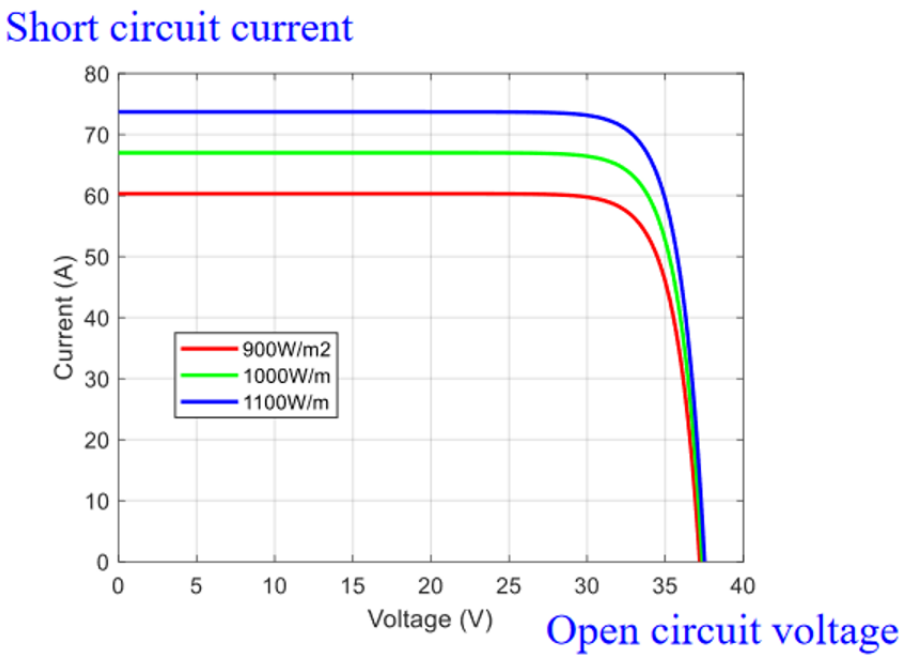

Figure 3.6: Current/Voltage relationship with short circuit current and open circuit voltage labeled

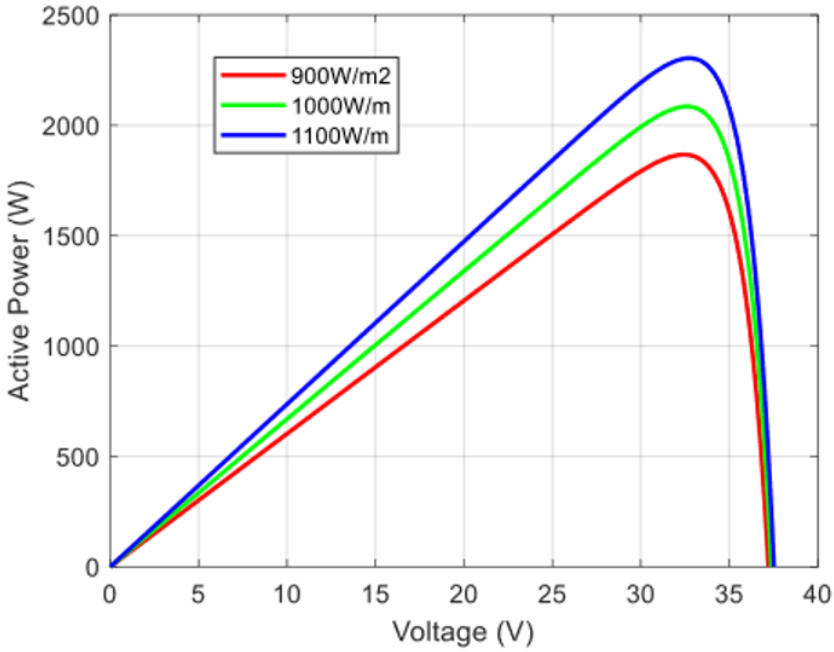

Figure 3.7: Active Power/Voltage relationship

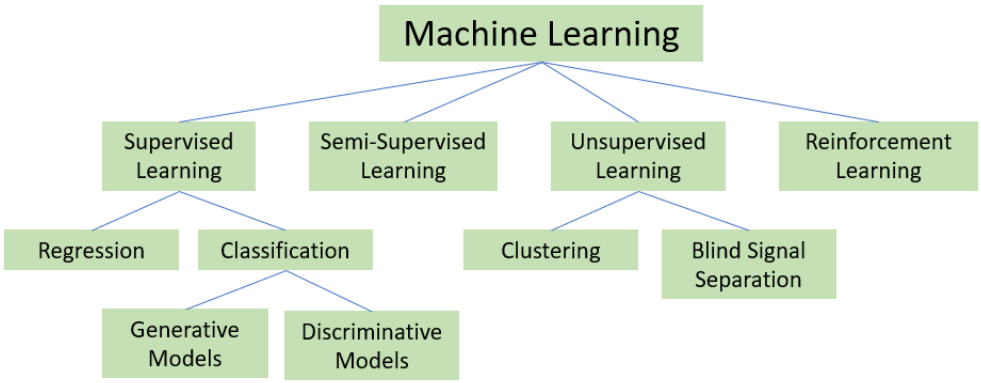

Figure 3.8: Tree diagram of branches of machine learning.

Similarly, as the irradiance increases, the open circuit voltage also increases but not by a lot. This can be seen in Figure 3.6 Another characteristic of photovoltaic output is that as temperature increases, the open circuit voltage decreases, as shown in Figure 3.7; however, as the temperature increases, the short circuit current does not increase by a lot [13].

3.2 Machine Learning for Mathematical Prediction of PV Output

The mathematical prediction of these models can be represented and established through machine learning, which is the point of this thesis. Machine learning provides the system with the ability to learn automatically from experience and data sets to improve without having to be explicitly programmed to do so [13]. Figure 3.8 is a visual tree diagram of the possibilities of ML techniques for PV power output array. In this figure, there are many different types of machine learning and subsets, including a range from completely supervised learning to unsupervised learning. Machine learning itself is a very broad category.

There are many different approaches to photovoltaic modeling, but Figure 3.9 shows two: a data driven model (a), and a predicted outputs model (b) [13]. In Figure 3.9(a), historical irradiance, historical temperature, and historical outputs are programmed for the system to predict future outcomes based upon the past. In Figure 3.9(b), the predictions for the irradiance and temperature are applied into the data driven model from Fig 3.9(a) to ultimately be able to predict PV outputs without needing explicit programming [13].

Photovoltaic modeling can be performed through Simulink, an environment of Matlab. It allows the user to simulate hardware without having to write code, as block diagrams display the necessary information. For example, Figure 3.10 shows an model of a photovoltaic. Figure 3.10 is a portion of a greater model, but it shows the general idea and visual understanding of photovoltaic modeling. The various logic and hardware instruments are connected via wires to the PV system, as well as any other algorithms necessary for solar panel function. Oscilloscopes are set up for certain measurements. The point of this modeling is to predict the outcomes of solar panels, and

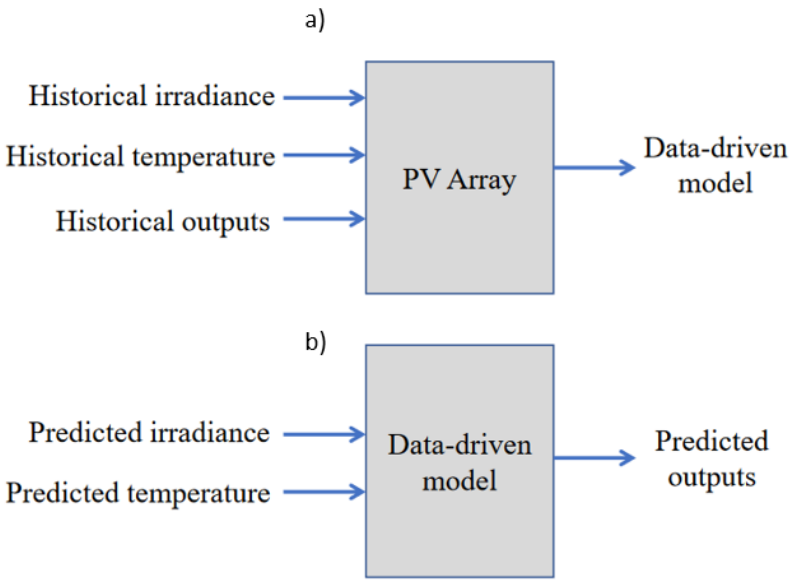

Figure 3.9: Machine learning application for PV systems.

Simulink is just one tool to do this. Using machine learning could make this process more efficient and economical.

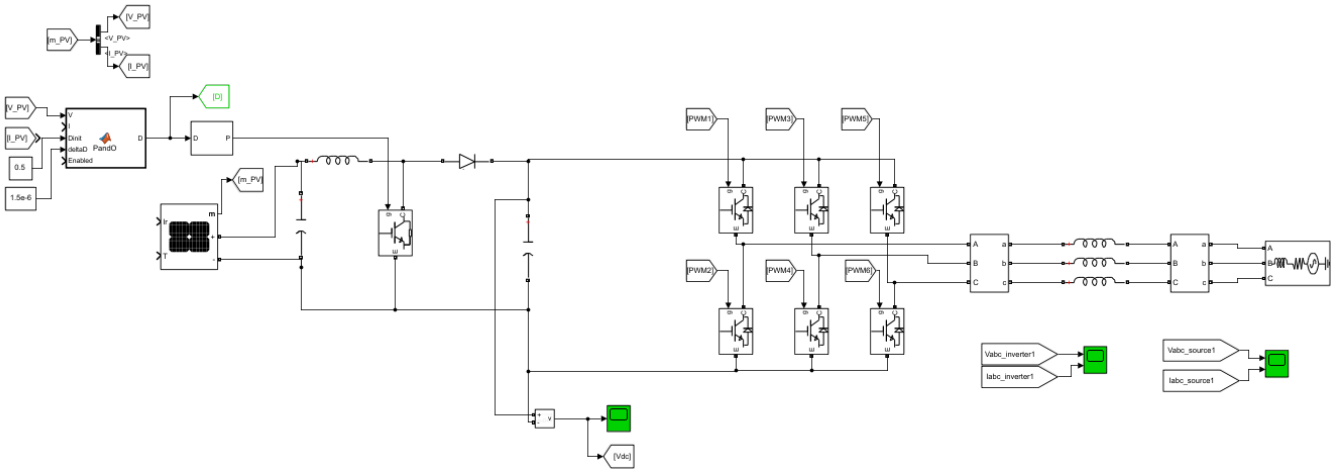

Figure 3.10: Simulink model of photovoltaic.

Chapter 4

Simulation Results

4.1 Code Results

The algorithm described in Chapter 2 above is created through a lengthy coding block. This code is written in the Python language and ran through Anaconda. This thesis was written using the Spyder 5.2.2 console (Python 3.9) through the Anaconda Navigator. This code specifically requires certain libraries, including NumPy SciPy, and Matplotlib. A Python environment like Anaconda already has these libraries installed. If not using Anaconda, it still is possible to run the code, but the creation of a virtual environment using a command will install the most recent requirements to make the libraries available [14]. Then, individual scripts can be run. This code is the results based upon the Next Generation Reservoir Computing paper. They are complements. It is sectioned into 10 different parts, each of which simulates a part of the algorithm. It is necessary to understand the code to understand the algorithm and ultimately the machine learning. There are sections describing the double scroll, NVAR, time delay, Lorenz63, Lyapunov, training data, and prediction data. The code is necessary to truly understand the details and methodology described in Chapter 2 of this paper. Three of the most important sections will be detailed and described in sections 4.1.1 and 4.1.2 of this paper. Each of these parts will be analyze and broken down with provided examples of code.

4.1.1 NVAR Double Scroll

First is the Double Scroll for the NVAR with incorporated time delays for forecasting. All three of the aforementioned libraries — NumPy, SciPy, and Matplotlib — are imported. The NumPy library supports large arrays and matrices. SciPy solves initial value problems for ordinary differential equations (ODEs); it also takes the min/max of objective functions and solves for nonlinear programming. Matplotlib is a state based interface that provides an implicit way of plotting functions. After calling in these function, the first trial of the NVAR is run, with an allotted given warm-up time. There are certain parameters that are established early on, including the time step (dt), training time, testing time, and the time of the Lyapunov function for the double-scroll system [14]. After these variables are declared, they are transformed into discrete time versions of themselves, by dividing by the specific time step, dt. The dimensions of the system are declared and the vector is created. Figure 4.1 shows a code snippet of the function used to establish the double scroll system. The equations are initialized as dV, dy0, dy1, and dy2. These equations depend on variables, r and g that are previously established within the double scroll metric. The double scroll system is then established using several of the previously mentioned parameters. The NVAR is synthesized for three different RC problems: ”forecasting the short term dynamics, reproducing the long term climate of a chaotic system, and inferring the behavior of any unseen data of a dynamical system” [14]. An array is created to hold the linear part of the feature vector and the linear parts are filled in correspondingly at all times. The non-linear part is then filled in. Next, a place is generated to store feature vectors for predictions. Finally, a prediction is performed using a for loop function. Figure 4.2 shows the data outcome for the NVAR double scroll.

The NVAR is ran as many times times as necessary to calculate the NRMSE at fixed points. There are three training phases and a prediction attractor that works to establish the data found in Figure 4.2.

```python
def doublescroll(t, y):
    # y[0] = V1, y[1] = V2, y[2] = I
    dV = y[0]-y[1] # V1-V2
    g = (dV/r2)+ir*np.sinh(alpha*dV)
    dy0 = (y[0]/r1)-g
    dy1 = g-y[2]
    dy2 = y[1]-r4*y[2]

    return [dy0, dy1, dy2]
```

Figure 4.1: Double scroll function.

```
 ridge regression parameter: 0.001

mean, meanerr, train nrmse: 0.0011818339720932798 3.0171091678616985e-05
mean, meanerr, test nrmse: 0.0045213330702967725 0.0010099275730963088

mean, meanerr, fp1 nL2 distance: 0.0020844669532041013 0.00022771659282137637
mean, meanerr, fp2 nL2 distance: 0.0020844669532041247 0.00022771659282137558
mean, meanerr, fp0 nL2 distance: 0.0 0.0

mean, meanerr, fp1 [-0.00163635 -0.00010137 -0.00128574] [1.82548015e-04 1.47496869e-05 1.36781322e-04]
mean, meanerr, fp2 [0.00163635 0.00010137 0.00128574] [1.82548015e-04 1.47496869e-05 1.36781322e-04]
mean, meanerr, fp0 [0. 0. 0.] [0. 0. 0.]

nL2 distance to mean, meanerr, fp1 0.0020835177791417624 0.00022858359728159268
nL2 distance to mean, meanerr, fp2 0.0020835177791417785 0.00022858359728159279
nL2 distance to mean, meanerr, fp0 0.0 0.0
```

Figure 4.2: Outcome for NVAR double scroll.

```
# number of NRMSE trials
npts=10
# how far in to Lorenz solution to start
start=5.
# units of time to train for
traintime=10.
# ridge parameter for regression
ridge_param = 2.5e-6
```

Figure 4.3: Parameters for NVAR with time delays.

```
dy0 = sigma * (y[1] - y[0])
dy1 = y[0] * (rho - y[2]) - y[1]
dy2 = y[0] * y[1] - beta * y[2]
```

Figure 4.4: Portion of Lorenz63 algorithm.

4.1.2 NVAR with Time Delays for Lorenz Forecasting

The next significant piece of code written was regarding the installment of the NVAR with the time delays for Lorenz forecasting. First, a certain number of parameters are established. Figure 4.3 shows a code snippet for these parameters, where 'npts' is the number of NRMSE trials, the start time, units for time, and ridge parameter for regression are all set [14].

After this parameter establishment, a warm up trial is run given the initial warm up time, 'start'. This is done through a function which is looking for any errors in the system. The time step is set to 0.25, and the Lyapunov time is established for the Lorenz system [14]. After these variables are the discrete-time variables versions, which is done by dividing the variables by the time step, 'dt'. Next, a 3x2 matrix is created and both the linear and nonlinear portions of the feature vector are distinguished. As mentioned in Chapter 2, the both the linear and nonlinear portions of the feature vector as important. Figure 4.4 shows a code snippet of the Lorenz 63 algorithm. Lorenz is a 3 dimensional system, as shown by dy0, dy1, and dy2. These three variables are formed into an array, and then the maximum, minimum and mean values are found for all three of the components using a for loop.

The NVAR algorithm is next tested. An array is created that holds the linear portion of the feature vector. For all times, t, the linear part of the vector is filled in and this data is put into an array for holding. The nonlinear part is done in a similar fashion, using a for loop for a range of the linear data. The ridge regression is then conducted, where W_{out} is trained to map out the array for the NVAR linear full feature vector to the Lorenz63 system. W_{out} is then applied to to train the feature vector, and NRMSE is calculated from the true Lorenz output and the training data [14].

The next key is step is actually performing the predictions that have been mentioned in this

i	int	1	9
n_fp0_diff_v	Array of float64	(10,)	[0.00115807 0.00194233 0.00182655 0.00374413 0.00259939 0.0066567 0.0 ...
n_fp1_diff_v	Array of float64	(10,)	[0.00153701 0.000684 0.00382102 0.00103462 0.00067719 0.00079683 0. ...
n_fp2_diff_v	Array of float64	(10,)	[0.00079129 0.00174694 0.0005527 0.00076525 0.00089662 0.00072351 0. ...
npts	int	1	10
p_fp0_norm_v	Array of float64	(10, 3)	[[6.94093328e-06 -1.12574992e-03 2.71612688e-04] [1.07290687e-04 . ...
p_fp1_norm_v	Array of float64	(10, 3)	[[0.00093853 0.00060176 0.00105804] [0.00059512 -0.0002203 0.00 ...
p_fp2_norm_v	Array of float64	(10, 3)	[[-6.69169710e-04 1.48046335e-04 3.95518978e-04] [-1.00899500e-03 - ...
ridge_param	float	1	2.5e-06
start	float	1	5.0
test_nrmse_v	Array of float64	(10,)	[0.00269091 0.00137871 0.00249468 0.00587657 0.00033801 0.00170193 0. ...
train_nrmse_v	Array of float64	(10,)	[0.00011149 0.00010687 0.00011563 0.0001043 0.00010146 0.00010444 0. ...
traintime	float	1	10.0
warmup_v	Array of float64	(10,)	[5. 15. 25. 35. 45. 55. 65. 75. 85. 95.]

Figure 4.5: NVAR time delay Lorenz forecasting outcome.

```
 ridge regression parameter: 2.5e-06

mean, meanerr, train nrmse: 0.00010633758520008862 1.4888232948528239e-06
mean, meanerr, test nrmse: 0.0024014044808879963 0.0005282065290949091

mean, meanerr, fp1 nL2 distance: 0.0012695163350966007 0.000290965428129616
mean, meanerr, fp2 nL2 distance: 0.0008296874505165322 0.00010390222557622253
mean, meanerr, fp0 nL2 distance: 0.0036877582254422905 0.00092221917624813 57

mean, meanerr, fp1 [0.00082101 0.00030571 0.00082964] [0.00014696 0.0001872  0.00020881]
mean, meanerr, fp2 [-6.08177656e-04  1.21427361e-06  4.47689062e-04] [4.65791233e-05 1.08477015e-04
9.30925265e-05]
mean, meanerr, fp0 [0.00017831 0.0010225  0.0005605 ] [9.66607770e-05 5.53970869e-04 1.32479084e-03]

nL2 distance to mean, meanerr, fp1 0.0012065721087182947 0.00031661716239122275
nL2 distance to mean, meanerr, fp2 0.0007551867527157848 0.00015034326019617214
nL2 distance to mean, meanerr, fp0 0.0011796036362699263 0.0014392004058161579
```

Figure 4.6: Alternative NVAR time delay Lorenz forecasting outcome.

paper. This prediction is performed through a for loop within a specific range. The linear data is copied into the whole feature vector and the nonlinear part of the data is filled in. The delay taps are also filled in for the next state of the algorithm, and then the prediction is performed [14]. Finally, in the NVAR with time delays for Lorenz forecasting section, a function is created to perform a single step NVAR prediction for a fixed point. Figure 4.5 shows the outcome of this code.

The left hand column indicates which variables are being printed, including, start time, ridge regression parameter, warmup time, and the normalized distance from the real to the predicted fixed point data. The type of integer is shown as well as the size of the variable. Finally, the values are displayed. As seen from the value column, the ridge parameter is sized at $2.5 * 10^{-6}$. Figure 4.6 also shows another view of the outcome of the NVAR. The mean for the training NRMSE is given at about 0.00106 and the mean for the testing NRMSE is found at about 0.002401. Some other significant data points to note are the normalized differences between the true and predicted data,represented by the fp0-2 variables. These have a range from 0.00127 - 0.00369 [14].

4.1.3 NRMSE vs Training Time

The next significant piece of coding revolves around NRMSE in comparison to training time. The training time is very important regarding the performance of the machine. A quicker training time means that the algorithm can learn quicker and is ultimately more efficient. To accurately

```
traintime_v=np.arange(start_train,end_train+step_train,step_train)

warmup_v=np.empty((npts_train,npts))
for i in range(npts_train):
    warmup_v[i,:]=np.arange(start,traintime_v[i]*npts+start,traintime_v[i])

testNRMSE=np.empty(npts_train)
testNRMSEerr=np.empty(npts_train)
```

Figure 4.7: Establishment of NRMSE vector.

assess the efficiency of this training time, a comparison between the NRMSE and the training time is performed. The code is similar to the previous sections, 4.1.1 and 4.2.2, with some specific modifications [14]. After establishing the number of NRMSE trials as a variable at 20, as well as NRMSE training time at 21, the algorithm for this comparison is created. A vector is created using both the warmup and training times. This vector is then divided into different spaces of the original number of NRMSE trials (20) segments of length, set to the variable traintime. Figure 4.7 shows an example of this code. In the figure, the vector is created and represented by variable traintime_v. The np.arrange formats the array with evenly spaced intervals per the interval [14]. The warmup data is then initialized to return a new array without any entries given the command, np.empty. A for loop is created to test the range of the training NRMSE trials, in which the vector is then arranged in the aforementioned way. Finally, the testing data and any errors that might have occurred are put into an np.empty vector.

4.1.4 Results

The graphical outcomes of the described outcome can be seen in Chapter 2 of this paper. For example, shown in Figures 2.12 and 2.13 is the graphical outcome of the Lorenz63 system for the inferred dynamics. The variables written in the coded section directly create the metrics seen within these graphs. The algorithm written, the machine learning system, creates this data which suggests possible outcomes for a PV system.

Figure 4.8 shows the outcome of the NRMSE vs the training data points. The NRMSE shows the differences between models at different points. It measures the model performance using the mean of the data. As shown from the figure, as the training data set size in increased, the NRMSE decreases. A lower NRMSE indicates a better fit, suggesting that as the training data set size increases in size, the predictions of the algorithm are more accurate.

Figure 4.9 is another outcome of the algorithm described in the sections above, where it shows the inferred Lorenz63 data of W_{out}. The figure is a graphical representation of W_{out} at different time intervals. The blue bars represent the useful data points. Any noise has been filtered out of this graph, as it is just focusing on the relevant data. As seen from the graph, W_{out} ranges from -0.2 to 0.2, and 0 to 2 [14]. The y interval is the various time intervals that are being tested. This data shows how different intervals effect W_{out}. There are two different cases being tested in this figure, as indicated by the two plots.

For comparison to Figure 4.9, 4.10 shows the predicted outcomes for W_{out}. This graph has three separate outcomes for W_{out}, as indicated by the three plots. Additionally, unlike Figure 4.9, 4.10 has noisy data. This noise is indicated by the red sections. This figure shows the test in all three dimensions, x,y, and z. The data shows the various outcomes for W_{out} as predicted when put

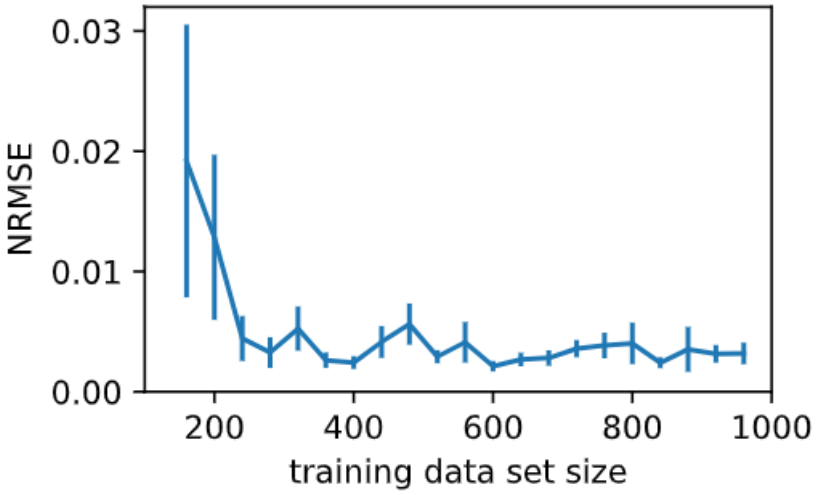

Figure 4.8: NRMSE vs training data.

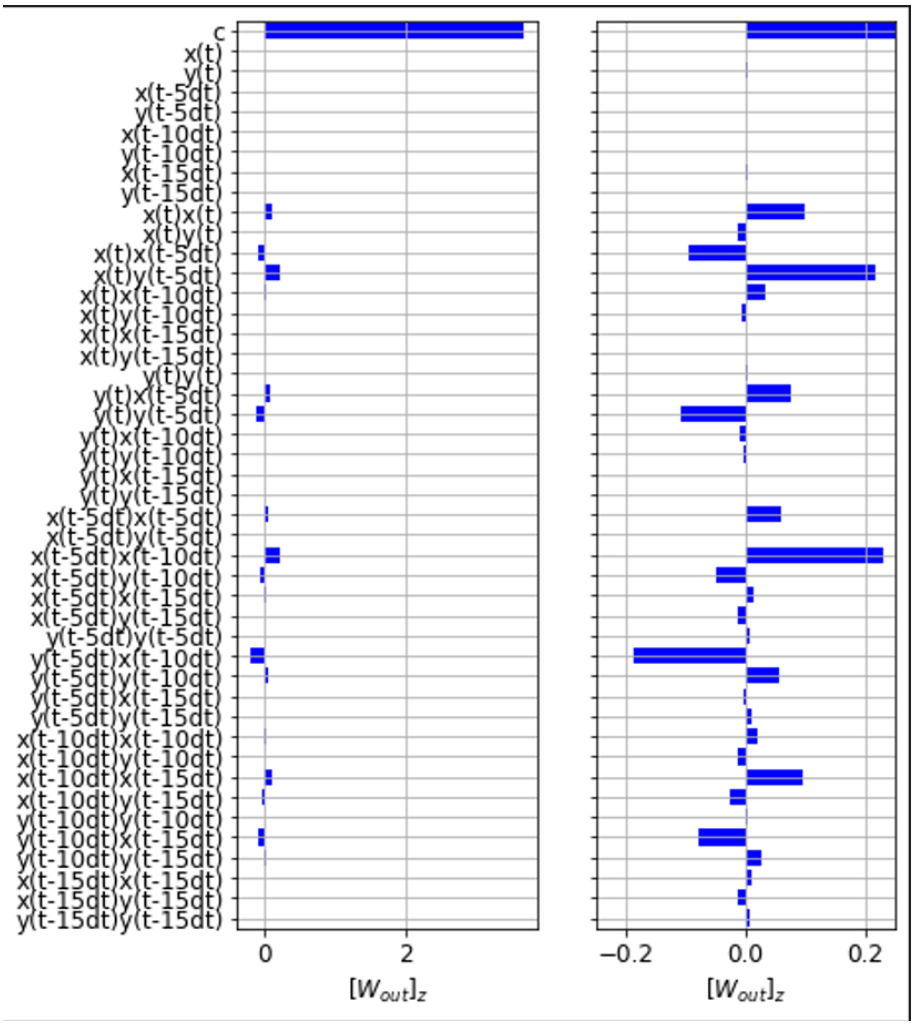

Figure 4.9: Inferred Lorenz data.

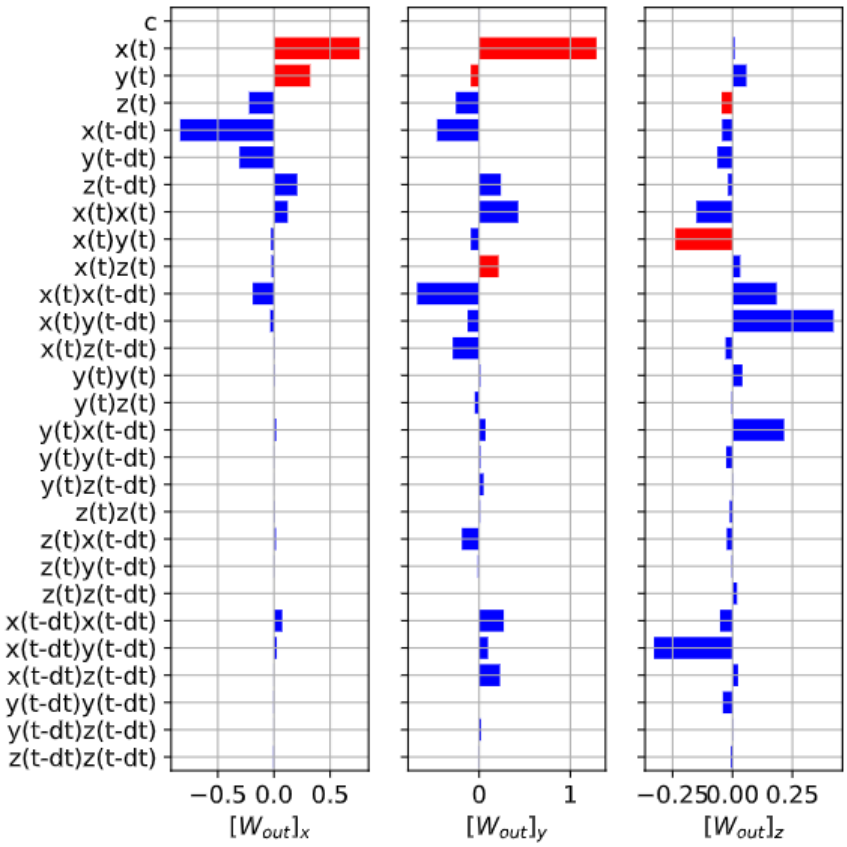

Figure 4.10: Predicted Lorenz data.

through the NGRC algorithm [14]. The noise can be filtered out, but it was kept in for comparison to Figure 4.9, which does not have any noise.

4.2 Trajectory

There are still many considerations necessary to truly understand reservoir computing and next generation reservoir computing. The possibilities surrounding NGRC and RC are endless and there is still much to be learned. For example, in related research, data drive linearization methods have been on the rise. These methods suggest the use of a vector field with projection on a linear yet finite space, represented through simple monomial functions [11]. Contrasting to NGRC, this type of modeling aims to create a model of the vector field directly from the data, ultimately learning the dynamical flow over a finite period of time. Some of the larger components of W_{out} can be found from the vector field, suggesting that the NGRC predictions and flow are different from the vector field. Another possibility comes from the fact that a few components of W_{out} are small, indicating that features can be removed without harming the integrity of the test [11].

There are also some limitations of this thesis that must be further explored. First, this study considers only noise-free generated data. The noisy data is disregarded for ease of interpretation. The use of regularized regression allows for this noise tolerance, as the best model is chosen regardless of any noise or uncertainty. Another limitation is that only low-dimensional dynamics were considered. There has been previous work done using a traditional RC system, but this paper only uses low dimensional dynamics [11].

Chapter 5

Conclusion

5.1 Further Direction and Conclusion

There are so many possibilities when it comes to machine learning. It truly is the future. There is a lot of fear and misinformation surrounding artificial intelligence, but there are so many opportunities and advancements that can be made from machine learning, this thesis highlighting one of them. NGRC has many important applications for the learning process of dynamical systems. It is an ideal system for these systems because it requires the optimization of fewer parameters and therefore has a quicker learning time compared to traditional RC systems. Having a shorter learning time means that the algorithm can identify patterns quicker within the data. The NGRC has underlying and hidden implicit components that allow it to perform metrics that the traditional RC system would be unable to do. This suggests that the possibilities and applications of NGRC are plentiful, in comparison to the traditional system. For example, the NGRC could be applied with only observed data to create a digital twin for the systems, limiting the amount and type of training data needed to create accurate predictions [11]. NGRC could also be used for nonlinear control; this is important for making up for any losses that come from the nonlinear portion of the featured vectors.

To truly understand the context of the topics described in this paper, an astronomical amount of research and base comprehension was necessary. NGRC and machine learning are both complex and diverse topics that still require a lot of research and learning. Given more time and more resources, and now with this baseline understanding of artificial intelligence, this would be an interesting project to continuing researching. Considering the prevalence of AI for the future, understanding how it works is truly an advantage. Through the work that I have done in this thesis, I now have a greater understanding and appreciation for the processes of machine learning.

When it comes to energy technological advancement, as described earlier in this paper, the application of machine learning technologies towards renewable energy resources, specifically photovoltaic systems is an essential step forward. Currently, photovoltaic systems lack a high level of efficiency. With better data analysis, this efficiency of energy conversion will rise. NGRC can increase efficiency of PV systems because with more testing and trials, the training data will see a decrease in the learning curve for the algorithm. This research cannot come soon enough however, as the need for renewable energy is ever growing. Rising global temperatures are leading to a greater demand for alternative energy sources and this demand will lead to greater allocation of resources and research to make alternative sources as efficient as fossil fuels. This increase in research will allow methodologies like machine learning, water immersion, or solar concentration to improve the power production of PV systems. It is clear that these advancements are the future and continuing research within these fields will continue to improve the power output of photovoltaic system, ultimately benefiting the world in dire need of a climate crisis solution.

Bibliography

[1] JCAY Rizk and Y Chaiko. Solar tracking system: more efficient use of solar panels. *World Academy of Science, Engineering and Technology*, 41(2008):313–315, 2008.

[2] Best research-cell efficiency chart, 2023.

[3] M Rajvikram and S Leoponraj. A method to attain power optimality and efficiency in solar panel. *Beni-Suef University journal of basic and applied sciences*, 7(4):705–708, 2018.

[4] Office of Energy Efficiency Renewable Energy. Confronting the duck curve: How to address over-generation of solar energy. `https://www.energy.gov/eere/articles/confronting-duck-curve-how-address-over-generation-solar-energy`, October 12 2017.

[5] Rizwan Arshad, Salman Tariq, Muhammad Umair Niaz, and Mohsin Jamil. Improvement in solar panel efficiency using solar concentration by simple mirrors and by cooling. In *2014 international conference on robotics and emerging allied technologies in engineering (iCREATE)*, pages 292–295. IEEE, 2014.

[6] Sayran A Abdulgafar, Omar S Omar, and Kamil M Yousif. Improving the efficiency of polycrystalline solar panel via water immersion method. *International Journal of Innovative Research in Science, Engineering and Technology*, 3(1):8127–8132, 2014.

[7] Jorge Felipe Gaviria, Gabriel Narváez, Camilo Guillen, Luis Felipe Giraldo, and Michael Bressan. Machine learning in photovoltaic systems: A review. *Renewable Energy*, 196:298–318, 2022.

[8] Cristian-Gvőző Haba. Monitoring solar panels using machine learning techniques. In *2019 8th international conference on modern power systems (MPS)*, pages 1–6. IEEE, 2019.

[9] Tom Michael Mitchell et al. *Machine learning*, volume 1. McGraw-hill New York, 2007.

[10] M.I. Jordan and T.M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science Mag*, July 17 2015.

[11] Daniel J Gauthier, Erik Bollt, Aaron Griffith, and Wendson AS Barbosa. Next generation reservoir computing. *Nature communications*, 12(1):5564, 2021.

[12] L Chua T Matsumoto et al. The double scroll.

[13] Yan Li. *Cyber-Physical Microgrids*. Springer, 2021.

[14] Daniel J Gauthier, Erik Bollt, Aaron Griffith, and Wendson AS Barbosa. Next generation reservoir computing. `https://github.com/quantinfo/ng-rc-paper-code`, August 18 2021.