Beiträge zum Verständnis von minimalem Reservoir Computing

Masterarbeit - KI in der Physik (M.Sc.Physics)

Fakultät für Physik - Ludwig-Maximilians-Universität München

Deutsches Zentrum für Luft- und Raumfahrt (DLR) - Institut für Materialphysik im Weltraum

Betreuung:

PD Dr. Christoph Räth, Davide Prosperino

Masterarbeit eingereicht von:

Vincent Groß

10. Februar 2026

Towards Understanding of Minimal Reservoir Computing

Master's thesis - AI in Physics (M.Sc.Physics)

Faculty of Physics - Ludwig-Maximilians-University Munich

German Aerospace Center (DLR) - Institute for Materials Physics in Space

Supervision:

PD Dr. Christoph Räth, Davide Prosperino

Master's thesis submitted by:

Vincent Groß

February 10, 2026

Acknowledgements

First and foremost, I would like to thank my supervisors, PD Dr. Christoph Räth and Davide Prosperino, for their invaluable guidance and support throughout this process.

I would also like to thank PD Dr. Christoph Räth for proposing the topic of this thesis.

Additionally, I am grateful to the Deutsches Luft- und Raumfahrtzentrum for providing the resources and opportunity to conduct this research.

Also I would like to thank Albert Fitz for supporting me with some of the analysis done in this thesis.

Grammarly was used as a final editing tool to check for grammatical errors, spelling mistakes, and punctuation issues, with all suggestions critically reviewed and accepted or rejected by the author.

The codes used to obtain the results presented in this thesis are included in the digital version.

Contents

Chapter 1

Introduction

The prediction of complex dynamic systems presents a significant challenge across numerous disciplines, ranging from scientific applications such as climate modeling to economic domains like financial market analysis. A defining property of such systems is their sensitive dependence on initial conditions: even a slight variation in a system parameter can result in dramatically different system evolutions and outcomes. These minor differences in initial conditions amplify exponentially over time, a phenomenon commonly attributed to nonlinearities in the governing equations [1]. Consequently, achieving accurate predictions in complex systems remains a notoriously difficult task.

Recent advances in artificial intelligence have substantially enhanced our ability to forecast the behavior of complex systems. Among the promising developments is "Minimal Reservoir Computing" (Minimal-RC) [2], an architecture specifically tailored for time series prediction and the central focus of this thesis. The objective of this work is to advance the explainability and understanding of Minimal-RC, providing new insights into its mechanisms and potential applications.

The motivation for this research arises from a striking phenomenon observed in the prediction of the Lorenz system: a sharp improvement in predictive performance occurs when the nonlinearity order in the generalized reservoir state is increased from one to two [3]. The goal is to explain what is happening in between those two nonlinearities, that enables good predictions. The observation made, that the nonlinearities in the system equations need to closely match the nonlinearities in the reservoir state, suggests a deeper connection between the nonlinear structure of the system and the output matrix, raising fundamental questions about the conditions required for successful prediction in complex dynamical systems.

This thesis is structured to address these questions systematically. The theory section begins with a foundational overview of chaos theory and complex systems. It then presents reservoir computing as a specialized type of recurrent neural network, with a focused introduction to Minimal-RC. The numerical methods employed, particularly the fourth-order Runge-Kutta method for solving dynamical equations, are outlined next. Subsequently, the principal measures for estimating prediction quality—forecast horizon, lyapunov exponent, and correlation dimension—are introduced and explained.

The results section provides a comprehensive exploration of Minimal-RC in practice. It opens with a synthesis of previous knowledge, including detailed analyses of fractional nonlinearities, the Lorenz peak phenomenon, and variations in the Halvorsen system. In these analyses, a key finding is the match between the maximum nonlinearity present in the system equations and the order of nonlinearity required in the reservoir state for optimal predictions. The thesis then investigates the mixing of different dynamical systems, focusing on the emergence of the "Valley of Unpredictability," where predictive quality unexpectedly degrades. This issue is eventually resolved by using time delay embedding. A substantial portion of the results is dedicated to the complex extension of Minimal-RC, presenting and comparing several approaches to incorporating complex-valued states and outputs.

Further chapters discuss how defining system-specific functions for the generalized reservoir state can significantly enhance prediction quality, and systematically analyze how different input combinations and their alignment with system nonlinearities affect performance. m,kiooTheo key finding is that knowledge of the underlying equations enables better predictions—and, in some cases, makes prediction possible at

all. However, this presents a Catch-22 paradox: once the equations are known, there is no longer a need to apply machine learning. The influence of the spectral radius on the memory and predictive capacity of the reservoir is explored in depth, and the section concludes with an examination of fixed points and their implications for prediction stability.

Chapter 2

Complex Systems

2.1 The Logistic Map

In order to understand chaotic behavior one begins by considering a simple system: the Logistic map, which is defined by the following equation [4]:

$$x_{n+1} = r x_n (1 - x_n) \tag{2.1}$$

The Logistic map is a one-dimensional discrete-time system. Here, x_{n+1} denotes the value of the system at the next time step and depends solely on the value at the previous time step, x_n. The parameter r is a control parameter that can be varied. The Logistic map possesses two fixed points, given by:

$$f(x^*) = x^* \tag{2.2}$$

This means that, once the system has reached the fixed point x^*, the value of x remains constant over time. Fixed points are of particular interest because, when the system settles at such a point, all parameters remain unchanged and the process of making predictions becomes trivial. The fixed points of the Logistic map are:

$$x_1^* = 0, \; x_2^* = 1 - 1/r \tag{2.3}$$

Take a point x_n in the vicinity of a fixed point, represented as $x_n = x^* + \eta_n$, where η_n denotes a small deviation from the fixed point. It is of interest to determine, whether this deviation grows or decays over time. To investigate this, examine the evolution of x_n, which is given by:

$$x_{n+1} = x^* + \eta_{n+1} = f(x^* + \eta_n) \tag{2.4}$$

Since η_n is a small quantity, the expression $f(x^* + \eta_n)$ can be expanded using a Taylor series:

$$f(x^* + \eta_n) = f(x^*) + f'(x^*)\eta_n + \mathcal{O}(\eta_n^2) \tag{2.5}$$

Since $f(x^*) = x^*$ one arrives at the following equation:

$$\eta_{n+1} = f'(x^*)\eta_n \tag{2.6}$$

Set $\lambda = f'(x^*)$ and conclude:

$$\eta_n = \lambda^n \eta_0 \tag{2.7}$$

Depending on the magnitude of the multiplier λ, the deviation η_n will either grow or decay over time. Consequently, the fixed point x^* is either linearly stable or unstable. If $|\lambda| < 1$, the fixed point is linearly stable and the system will evolve towards this point. Here, the quantity of interest is the absolute value of λ, as the focus lies on whether multiples of λ^n grow or decay over time.

Examination of the fixed points of the Logistic map in greater detail gives more insight. The first derivative of the system equation evaluated at the fixed point is given by:

$$|\lambda| = |f'(x^*)| = |r - 2rx^*| \tag{2.8}$$

For $x_1^* = 0$ the multiplier will be $|\lambda| = |f'(x_1^*)| = |r|$. This means $x_1^* = 0$ will be stable in the range $0 < r < 1$. For $r > 1$, $x_1^* = 0$ becomes unstable. For $x_2^* = 1 - 1/r$ the multiplier becomes $|\lambda| = |f'(x_2^*)| = |2 - r|$. Therefore x_2^* will be stable in the range $1 < r < 3$. One has found, that for $r = 3$ the system will evolve on a two cycle, which means $f(q) = p$ and $f(p) = q$. There will be successive period doublings for $r > 3$ until $r = 3.56994$. For $r > 3.56994$ the Logistic map will behave completely chaotic. [5]

Figure 2.1 shows the bifurcation diagram of the Logistic map. The x-axis shows the parameter r and the y-axis the possible long term values of x. In a definite range of r ($r \in [1, 3]$) one can see the fix points of the system and for $r > 3$ the period doubling can be seen, up to the chaotic behavior. In the chaotic regime nearby initial conditions will separate.

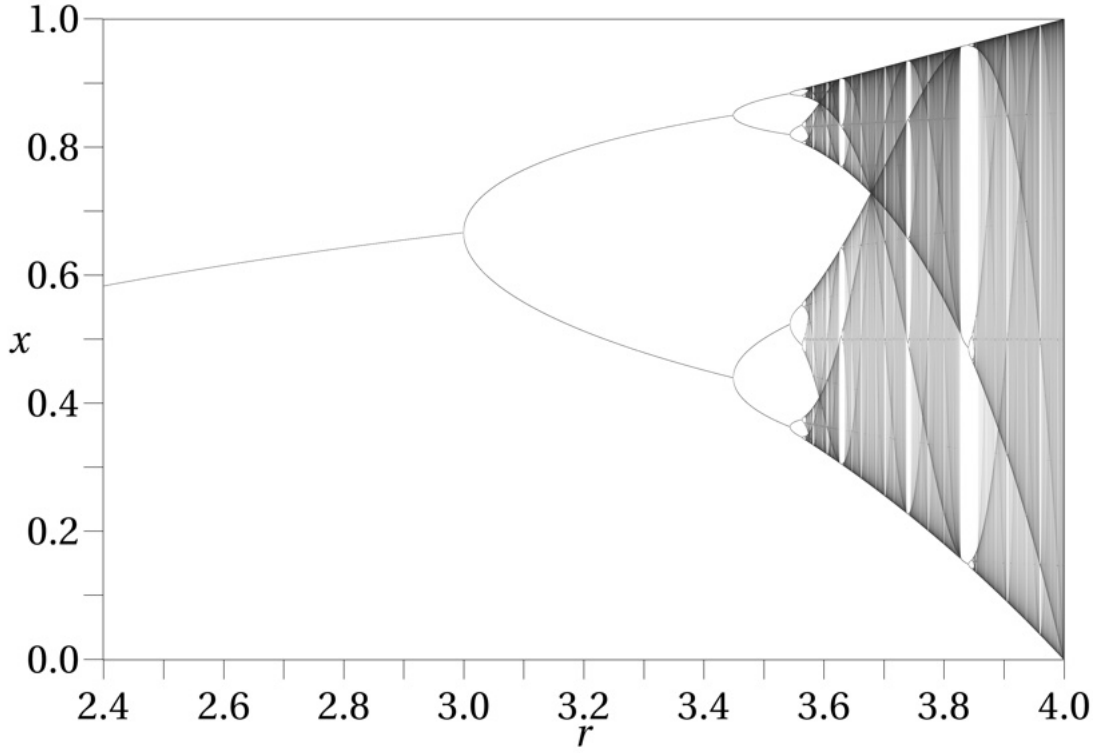


Figure 2.1: Bifurcation diagram of the Logistic map [6]

2.2 Dynamical Systems

In the first section, a system described by a discrete equation was examined, meaning that the evolution of the system occurs in discrete steps. In the previous example, each subsequent point depended only on the preceding point.

In general, d-dimensional dynamical systems are described by continuous equations of the form:

$$\dot{x}_i = f_i(t, x_1, x_2 ... x_d) \tag{2.9}$$

This can be expressed in vector form:

$$\dot{\vec{x}}(t) = \vec{f}(\vec{x}(t)) \tag{2.10}$$

A fixed point $\vec{x}^*$ in this case is defined by:

$$\vec{f}(\vec{x}^*) = \vec{0} \tag{2.11}$$

Once again, a small deviation $\vec{\eta}(t)$ from the fixed point is considered:

$$\vec{\eta}(t) = \vec{x}(t) - \vec{x}^* \tag{2.12}$$

For the ith component, one can write:

$$\dot{\eta}^i(t) = \dot{x}^i(t) = f^i(\vec{x}^* + \vec{\eta}(t)) \tag{2.13}$$

and perform a Taylor expansion:

$$f^i(\vec{x}^* + \vec{\eta}(t)) = f^i(\vec{x}^*) + \vec{\eta}^j(t)\frac{\partial f^i(\vec{x}^*)}{\partial x^j} + \mathcal{O}(\eta^2) \tag{2.14}$$

A sum over all $j \in \{1, \ldots, d\}$ is implied by the Einstein summation convention. By using equation (1.11) one obtains:

$$\dot{\eta}^i(t) = \frac{\partial f^i(\vec{x}^*)}{\partial x^j}\vec{\eta}^j(t) \tag{2.15}$$

The derivatives $\frac{\partial f^i(\vec{x}^*)}{\partial x^j}$ are associated with the matrix elements A^i_j of the Jacobian matrix, resulting in a homogeneous differential equation with constant coefficients:

$$\dot{\vec{\eta}}(t) = \boldsymbol{A}\vec{\eta}(t) \tag{2.16}$$

The general solution for such an equation is given by [7]:

$$\vec{\eta}(t) = \sum_j \vec{v_j} e^{\lambda_j t} c_0^j \tag{2.17}$$

Where $j \in \{1, \ldots, d\}$. The λ_j are the eigenvalues of the matrix $A^j_i = \frac{\partial f^i(\vec{x}^*)}{\partial x^j}$. The vectors $\vec{v_j}$ are the corresponding eigenvectors, and c_0^j are constant coefficients. Only when all the real parts of the λ_j are negative will the fixed point be stable, as the deviation $\vec{\eta}(t)$ approaches $\vec{0}$. The vectors $\vec{v_j}$ indicate the directions of the contributions. If $\lambda_j < 0$, the direction $\vec{v_j}$ will be stable. Conversely, if $\lambda_j > 0$, the direction $\vec{v_j}$ will be unstable. [8]

2.3 Fix Points in Two Dimensions

A two-dimensional system is considered, given by:

$$\dot{x} = f(x, y) \tag{2.18}$$

$$\dot{y} = g(x, y) \tag{2.19}$$

Let (x^*, y^*) be a fixed point, such that $f(x^*, y^*) = 0$, $g(x^*, y^*) = 0$. The Jacobian Matrix is given by:

$$\boldsymbol{A} = \begin{pmatrix} \frac{\partial f((x^*,y^*))}{\partial x} & \frac{\partial f((x^*,y^*))}{\partial y} \\ \frac{\partial g((x^*,y^*))}{\partial x} & \frac{\partial g((x^*,y^*))}{\partial y} \end{pmatrix}$$

With the characteristic polynomial equation, one can determine the eigenvalues using $(\det(A - \lambda I) = 0)$.

By defining:

$$\tau = trace(A) = \frac{\partial f}{\partial x}|_{x^*,y^*} + \frac{\partial g}{\partial y}|_{x^*,y^*} \tag{2.20}$$

$$\Delta = det(A) = \frac{\partial f}{\partial x}|_{x^*,y^*}\frac{\partial g}{\partial y}|_{x^*,y^*} - \frac{\partial g}{\partial x}|_{x^*,y^*}\frac{\partial f}{\partial y}|_{x^*,y^*} \tag{2.21}$$

and inserting into the characteristic polynomial, the eigenvalue equation is given by:

$$\lambda^2 - \tau\lambda + \Delta = 0 \tag{2.22}$$

with λ being an eigenvalue of the Jacobian matrix. Hence, the eigenvalues, which are important for the analysis of the fixed points, are given by:

$$\lambda_{1,2} = \frac{\tau}{2} \pm \frac{1}{2}\sqrt{\tau^2 - 4\Delta} \tag{2.23}$$

For imaginary λ, the flow of the differential equation will be rotational, depending on the value of the real part. A positive real part results in an unstable (repelling) spiral that moves away from the fixed point, whereas a negative real part leads to an attracting spiral that approaches the fixed point. If λ is purely imaginary, the trajectories will be circular.

Figure 2.2 shows a two-dimensional plane (the Δ–τ plane), which classifies fixed points according to the values of their imaginary and real parts. [9]

For $\Delta < 0$, λ_1 will be positive and λ_2 will be negative, as the root term will always be greater than $\tau/2$. Consequently, one direction will be stable and the other unstable, resulting in a saddle point.
For $\Delta > 0$, $\tau > 0$, and a positive root ($\tau^2 > 4\Delta$), the fixed points (nodes) will be unstable, since both λ are positive and, therefore, nearby initial conditions will diverge.
For $\Delta > 0$, $\tau > 0$, and a negative root, unstable spirals will occur, as the eigenvalues have a nonzero imaginary part and positive real parts.
For $\Delta > 0$, $\tau < 0$, and a negative root, the spirals will be stable, since the real parts are negative, causing nearby initial conditions to converge.
For $\Delta > 0$, $\tau < 0$, and a positive root, the nodes will be stable, since both λ are negative and, therefore, the separation of nearby initial conditions decreases over time.

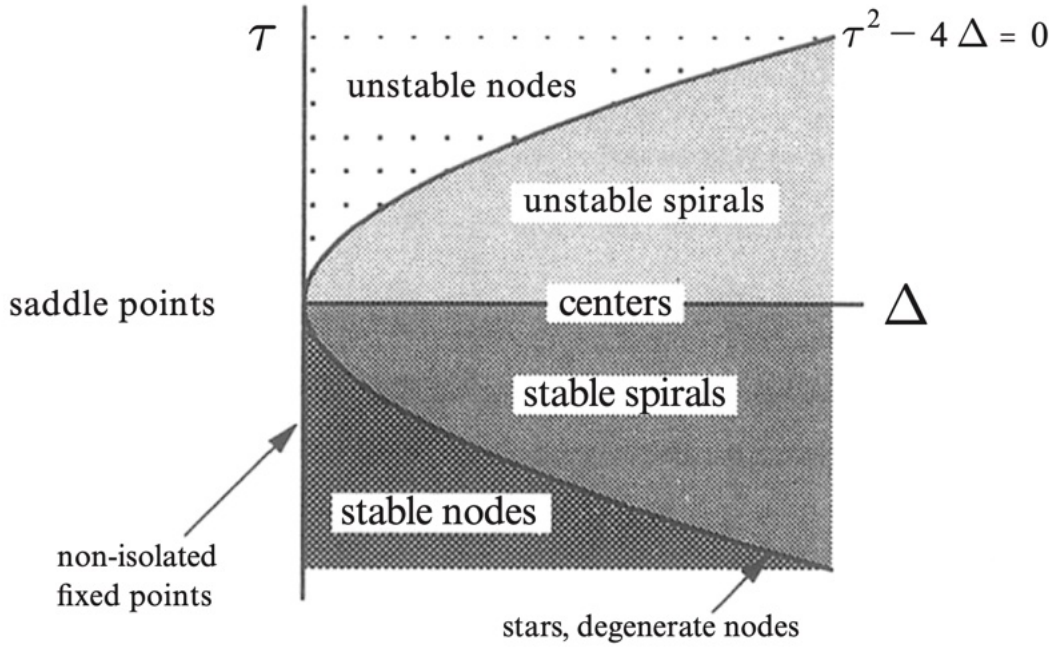


Figure 2.2: Classification of fixed points in two dimensions [9]

The same analysis can be applied to systems with three or more dimensions, allowing for even more complex flows and fixed points, such as spiraling saddle points.

2.4 Poincaré Map

Some systems do not possess fixed points, meaning their flow in phase space is not attracted to any particular location. Especially in higher dimensions, the flow of the differential equations may instead follow a periodic or nearly periodic orbit. It is possible to project onto a lower-dimensional hyperplane to identify stable and unstable orbits. One considers an d-dimensional system and an $(d-1)$-dimensional hyperplane S, as illustrated in Figure 2.3.

The surface of section S is required to be transverse to the flow, ensuring that trajectories pierce through

it. A Poincaré map P is a mapping from S onto itself, obtained by following the flow from one intersection to the next. Let $\vec{x}_k$ denote the k-th intersection. The Poincaré map is then defined by:

$$\vec{x}_{k+1} = P(\vec{x}_k) \tag{2.24}$$

Let $\vec{x}^*$ be a fixed point of the Poincaré map P ($P(\vec{x}^*) = \vec{x}^*$), then a trajectory starting at $\vec{x}^*$ will return to $\vec{x}^*$ after some time t and is therefore a closed orbit of the original system.
Hence, the Poincaré map transforms questions about closed orbits into questions about fixed points of a mapping. [10]

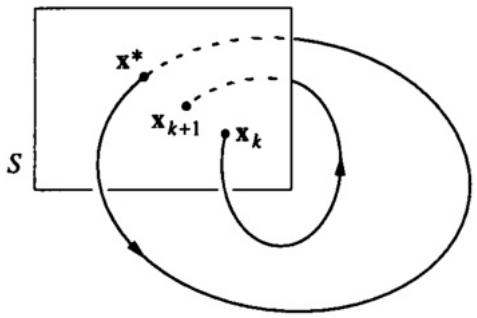


Figure 2.3: Example of a Poincaré section [9]

In general, dynamical systems do not settle on a stable trajectory. Many chaotic systems, however, are attracted to specific regions in phase space, known as "strange attractors." Regardless of the initial conditions, after a certain period of time, the flow of the system will always be found within this region of phase space. Within such regions, the behavior is typically completely chaotic. This thesis focuses on systems that occupy strange attractors. Systems that fill the entire phase space are not considered, as they offer limited possibilities for analysis.

2.5 The Lyapunov Exponent

A measure for chaos must be defined, indicating how long it takes for initially close trajectories to separate by a certain distance δ. The longer it takes for the trajectories to diverge, the less chaotic the system is considered to be. The Lyapunov exponent provides such a measure, and will be discussed in detail.

Let δ_n denote the separation of two trajectories after n iterations, and let δ_0 be the initial separation. δ_n grows exponentially, which is justified by the previous sections in equation (2.7) and (2.17):

$$|\delta_n| = |\delta_0| e^{n\lambda} \tag{2.25}$$

For d-dimensional flows that are continuous in time, one obtains:

$$||\vec{\delta}(t)|| = ||\vec{\delta}(0)|| e^{t\lambda} \tag{2.26}$$

For each dimension, there is a deviation that changes in magnitude over time:

$$\vec{\delta}(t) = (\delta_1(t), \delta_2(t), ..., \delta_d(t)) \tag{2.27}$$

λ is defined as the largest Lyapunov exponent. There is a Lyapunov exponent associated with each system dimension, however, the largest exponent will dominate the system's behavior. A positive largest Lyapunov exponent indicates chaotic behavior, whereas a negative largest Lyapunov exponent corresponds to non-chaotic behavior, since for $\lambda < 0$ the initial difference decreases exponentially.

The so-called Lyapunov time is introduced to facilitate the comparison of different chaotic systems. It is defined as:

$$T_L = \frac{1}{\lambda} \tag{2.28}$$

If T_L is large, the system will remain predictable for an extended period of time. [11]

2.6 Time Delay Embedding

The typical approach in scientific fields such as weather prediction is to construct a model that incorporates all relevant data and generates predictions based on initial conditions. However, creating such a model is not always straightforward. For example, it is known that the variability of the tropical oceans and changes in land surface influence the atmospheric state, but the precise nature of this influence is not yet fully understood. The number of parameters involved is so large that identifying all the necessary variables for accurate prediction becomes nearly impossible [12]. In other cases, it is not feasible to measure all parameters independently. For instance, in many chemical reactions, multi-dimensional measurements with high temporal resolution of both thermochemical states and fluid dynamic quantities are required. Attaining simultaneous precise multi-parameter measurements is a notoriously difficult task [13]. When proper data or a suitable model is unavailable, mathematical approaches can be employed to reconstruct the dynamical system from an incomplete sequence of observations.

Suppose the full dynamics of a system are described by a d-dimensional state vector $\vec{x}_t$ that evolves deterministically and continuously in time. Let y_t be one component of $\vec{x}_t$. According to Takens' theorem, by considering y_t and its values at earlier times—specifically, by shifting the coordinate y_t by a time lag τ for k times (the embedding dimension)—it is possible to reconstruct the phase space of the attractor. A generic map from a d-manifold to k-dimensional euclidean space is called an embedding. The set $\{y_t, y_{t+\tau}, y_{t+2\tau}, y_{t+3\tau}, \ldots, y_{t+k\tau}\}$ forms the k-dimensional embedded space. [14]

One might assume that taking $k \to \infty$ would yield the best results. However, Takens embedding theorem states, that a d-dimensional time-delayed version of one signal coordinate ($k = d$) suffices to embed a d-dimensional manifold, thereby reconstructing the phase space $\vec{x}_t$. [14][15]

Chapter 3

Reservoir Computing

To understand reservoir computing, one must first examine graphs and recurrent neural networks in general, since reservoir computing (RC) is a specialized form of recurrent neural networks.

3.1 Recurrent Neural Networks

Unlike conventional neural networks, which receive input data, process it through an arbitrarily complex network, and map it to a desired output by adjusting matrix weights to minimize a defined loss function [16], recurrent neural networks utilize a basic unit $\vec{h}_t$ (Figure 3.1), that is repeated throughout the network. Recurrent neural networks (RNNs) incorporate a form of memory known as the internal state [17]. This architecture is particularly well-suited for time series prediction, as it accounts for the temporal nature of data by using information from previously observed data points when evaluating new inputs.

Figure 3.1 illustrates the basic unit of an RNN. Suppose d quantities are measured at each time point in an experiment, resulting in a d-dimensional vector $\vec{x}_t$ at each time point, which serves as the input to the RNN (input layer). An RNN maintains a hidden state at each time point, represented by a p-dimensional vector $\vec{h}_t = f(\vec{x}_t, \vec{h}_{t-1})$, which is a function of both $\vec{x}_t$ and the previous hidden state $\vec{h}_{t-1}$ (hidden layer). Finally, the output is given by $\vec{y}_t = g(\vec{h}_t)$, which depends solely on the hidden state $\vec{h}_t$ (output layer). However, the hidden state itself depends on all the previous hidden states and the past data points, which means, that the output also indirectly depends on all previous data points: $\vec{y}_t = g(\vec{h}_t) = g(f(\vec{x}_t, \vec{h}_{t-1})) = g(f(\vec{x}_t, f(\vec{x}_{t-1}, \vec{h}_{t-2})))$. The output could, for example, be a prediction $\vec{x}_{t+1}$ of the time series provided to the network. In such a case, one would aim to train the RNN to minimize an objective function similar to:

$$L = ||\vec{y}_t - \vec{x}_{t+1}||^2 \tag{3.1}$$

In this approach, the goal is to map the current output state to the next state in the time series. With sufficient data and the presence of deterministic correlations—such as recurring patterns or other forms of temporal dependence—it is possible to train the network to make accurate predictions of future states.

All of this can be formulated within a mathematical framework as follows:

$$\vec{h}_t = tanh(\boldsymbol{W}_{hx}\vec{x}_t + \boldsymbol{W}_{hh}\vec{h}_{t-1} + \vec{b}_h) \tag{3.2}$$

$$\vec{y}_t = \sigma(\boldsymbol{W}_{yh}\vec{h}_t + \vec{b}_y) \tag{3.3}$$

The hidden vector $\vec{h}_t$ is updated whenever a new data point is introduced, according to the equation above. Here, $\boldsymbol{W}_{hx}$ and $\boldsymbol{W}_{hh}$ are trainable matrices, and tanh is an activation function that has proven effective in practice. The vector $\vec{b}_h$ serves as a bias term added to the hidden state. The matrix $\boldsymbol{W}_{yh}$ is also trainable and is used to map the hidden state to the output state. Additionally, a bias term $\vec{b}_y$ is added to the output state. The function σ represents another activation function, which can be chosen depending on the application.

$\boldsymbol{W}_{hx}$ is of dimension $(p \times d)$, $\boldsymbol{W}_{hh}$ is $(p \times p)$-dimensional, and $\boldsymbol{W}_{yh}$ is $(n \times p)$-dimensional. The

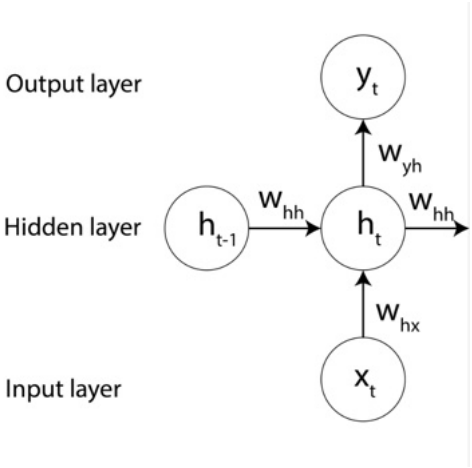


Figure 3.1: Basic unit of a recurrent neural network

input vector $\vec{x}_t$ interacts with the hidden state from the previous time step, $\vec{h}_{t-1}$, and thus indirectly incorporates information from all preceding time steps:

$$\vec{h}_t = f(\vec{x}_t, \vec{h}_{t-1}) = f(\vec{x}_t, f(\vec{x}_{t-1}, \vec{h}_{t-2})) = u(\vec{x}_t, \vec{x}_{t-1}, \vec{x}_{t-2}, ...) \tag{3.4}$$

This property makes the RNN architecture particularly well-suited for data exhibiting deterministic relations. Due to the recursive nature of the core building block of an RNN (the recurrent unit, which is reused at every time step with shared parameters) and because information from the hidden state is passed forward as additional input from one time step to the next, this process can be represented as a self-loop, as illustrated in Figure 3.2.

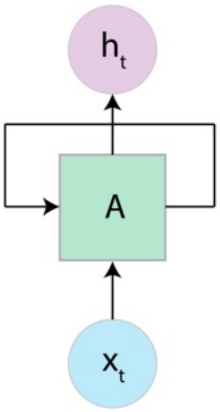


Figure 3.2: Core building block of a recurrent neural network [18]

An RNN can essentially be viewed as a timeline when the horizontal axis is unrolled, as illustrated in Figure 3.3.

Regular neural networks are trained using backpropagation, which involves calculating the gradient of the loss function and updating the matrix weights in the direction of the gradient to find the optimal solution. For RNNs, a specialized method known as backpropagation through time has been developed [19]. This algorithm operates on the unrolled version of the RNN, taking into account that all layers share the same parameters. The gradients for the network parameters are accumulated over all time steps before being propagated backward through the network.

In practice, significant numerical instabilities can arise due to the vanishing gradient problem. This occurs because of the repeated multiplication of $\boldsymbol{W}_{hh}$ with itself over N time points in a sequence. As a result, critical temporal correlations may be lost during the learning process of basic RNNs, leading

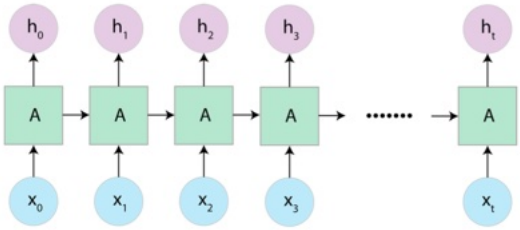


Figure 3.3: Timeline character of a recurrent neural network [18]

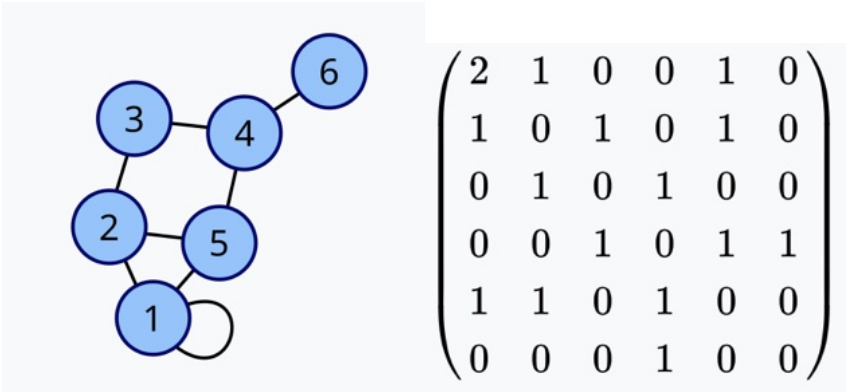
to poor long-term memory. Several methods have been introduced to address this issue, such as Long Short-Term Memory (LSTM) networks [20] and Gated Recurrent Units (GRUs) [21], although these will not be discussed here. Reservoir Computing provides a different solution to avoid the vanishing gradient problem by not relying on gradient calculations at all, as will be discussed in the following sections. [18]

3.2 Graphs

A graph is a collection of objects that are related to each other. The objects are represented by vertices (also called nodes or points), which are connected by edges (or links) whenever a relationship exists between the vertices. Edges can be directed, as in directed graphs, or undirected, as in undirected graphs. Additionally, edges can carry weights that reflect the strength of the relationship between vertices.

Mathematically, a graph is defined as a pair $G = (V, E)$, where V is a set containing the vertices and E is a set of unordered pairs $\{v_1, v_2\}$ of vertices, representing the edges. A graph can be represented by the so-called adjacency matrix $\boldsymbol{A}$, which is a square matrix. The elements of this matrix indicate which pairs of vertices are adjacent (neighboring) and the strength of the relationship between them. An example of a graph and its corresponding adjacency matrix is shown in Figure 3.4. Entries on the diagonal indicate a connection of a vertex to itself. For this reason, all diagonal elements except for A_{11} are zero. A_{11} equals two in this case, since that node has two connections with itself. Because no direction is implied, both directions can be interpreted as input and output. A nonzero entry in the off-diagonal elements A_{ij} signifies a connection between the ith and jth vertices. For instance, since there is a connection between vertex 6 and vertex 4, there is a value other than zero in A_{64}. [22]

In this case, the graph is undirected, meaning that $A_{ij} = A_{ji}$. A connection between vertex i and vertex j implies a connection from vertex j to vertex i. Thus, for undirected graphs, the adjacency matrix is symmetric. The value of A_{ij} indicates the strength of the connection, with larger values representing stronger connections. In this particular example, the graph is unweighted.



(a) Visual representation of a graph (b) Adjacency matrix of the graph

Figure 3.4: Graph and its corresponding adjacency matrix [23]

Random graphs are a class of graphs in which one begins with N vertices and connects each pair of vertices with probability p. Most nodes have roughly the same number of connections (degree). The

node degrees follow a Poisson distribution. These networks are also called Erdős–Rényi random graphs. [24]

In real-world networks, connections often follow a so-called preferential attachment model [25]. In these networks, vertex connectivities follow a scale-free power-law distribution. As networks grow through the addition of new vertices, these new vertices tend to attach preferentially to vertices that are already highly connected. For instance, this phenomenon can be observed in social networks, where popular individuals have more connections and are more likely to be introduced to new people [26]. Social dynamics also encourage people to form friendships with those who are already well connected.

Regardless of the specific system, the probability $p(k)$ that a vertex in the network interacts with k other vertices decays according to a power law, following $p(k) \propto k^{-\gamma}$. To create a network of this type, one begins with a small number of vertices. At each time step, a new vertex is added. To incorporate preferential attachment, it is assumed that the probability p that a new vertex will connect to vertex i depends on the connectivity k_i of that vertex, such that $p(k_i) = \frac{k_i}{\sum_j k_j}$, where the sum over j includes all existing vertices. [25]

3.3 Reservoir Computing

Reservoir Computing is a specialized form of a recurrent neural network. It maps input signals into higher-dimensional computational spaces using the dynamics of a fixed, nonlinear system referred to as the reservoir. The input signal is fed into the reservoir, which functions as a "black box." A simple readout mechanism is then trained to interpret the state of the reservoir and map it to the desired output. One advantage of this approach is that training is required only at the readout stage, since the reservoir dynamics remain fixed throughout. [27]

The framework was first introduced in 2001 by Herbert Jäger in the work "The "echo state" approach to analysing and training recurrent neural networks" and has since been further developed. [28]

Figure 3.5 presents a visualization of the reservoir computer (RC) architecture. The input vector $\vec{u}(t)$ enters the reservoir, is processed by its dynamics, and the output vector $\vec{s}(t)$ is subsequently extracted from the reservoir. No memory is required for introducing or extracting data from the network. Instead, the memory of past data points is encoded exclusively within the so-called reservoir states themselves. The reservoir receives the input at time step t as follows:

$$\boldsymbol{W}_{in}\vec{u}(t) \tag{3.5}$$

$\boldsymbol{W}_{in}$ is the input matrix. It is an $(N \times d)$ matrix, which is generated using uniformly distributed random numbers within a certain range $[-a, a]$. This matrix maps the d-dimensional system data onto the N-dimensional reservoir space. The reservoir is typically represented by a large random network with N nodes, encoded in an $(N \times N)$ adjacency matrix $\boldsymbol{A}$, which is characterized by its spectral radius ρ. [29]

The reservoir state at time t is described by the vector $\vec{r}(t) = (r_1(t), r_2(t), ...r_N(t))^T$. The next state of the reservoir, $\vec{r}(t+\Delta t)$, is determined by combining the input $\boldsymbol{W}_{in}\vec{u}(t)$ with the previous reservoir state, multiplied with the adjacency matrix $\boldsymbol{A}\vec{r}(t)$:

$$\vec{r}(t + \Delta t) = tanh\left(\boldsymbol{A}\vec{r}(t) + \boldsymbol{W}_{in}\vec{u}(t)\right) \tag{3.6}$$

The tanh function is used as the activation function. However, using tanh can introduce issues such as mirror attractors, although various solutions have been proposed to address this problem. [30]

The output $\vec{s}(t + \Delta t)$ is obtained by mapping the typically higher-dimensional reservoir state $\vec{r}(t + \Delta t)$ onto the usually lower-dimensional output $\vec{s}(t + \Delta t)$ as follows:

$$\vec{s}(t + \Delta t) = \boldsymbol{W}_{out}\vec{r}(t + \Delta t) \tag{3.7}$$

The readout matrix $\boldsymbol{W}_{out}$ $(d \times N)$ depends on a large number of adjustable parameters. These matrix weights are the only parameters that are trained in RC.

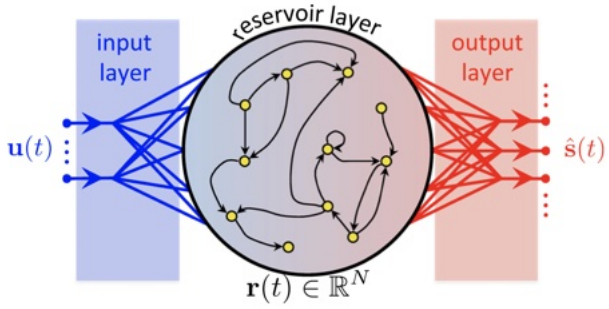


Figure 3.5: Reservoir computing architecture [31]

The key aspect of this method is the training procedure. The weights of $\boldsymbol{W}_{out}$ must be adjusted to achieve the desired outcome. In the context of time series prediction, the objective is for the outputs $\vec{s}(t)$ to closely approximate the target outputs $\vec{s}_d(t)$ that correspond to the input sequences $\vec{u}(t)$, shifted by one time step.

The training process spans T time steps. At each time step, the reservoir state $\vec{r}(t)$ and the data point $\vec{u}(t)$ are stored. The parameters of $\boldsymbol{W}_{out}$ are then optimized to minimize the mean squared difference between $\vec{s}(t)$ and $\vec{s}_d(t)$:

$$\sum_{0\leq t\leq T} ||\boldsymbol{W}_{out}\vec{r}(t) - \vec{s}_d(t)||^2 + \beta||\boldsymbol{W}_{out}||^2 \tag{3.8}$$

Again, $\vec{s}_d(t)$ is chosen to be the input $\vec{u}(t)$ at one time step in the future. The term $\beta||\boldsymbol{W}_{out}||^2$ helps to prevent overfitting and is known as Tikhonov regularization. [32]

Minimizing equation (1.8) to approximate the desired outcome as closely as possible yields:

$$\boldsymbol{W}_{out} = \vec{r}^T\vec{s}_d(\vec{r}^T\vec{r} + \beta\mathbf{1})^{-1} \tag{3.9}$$

The above equation constitutes a ridge regression problem, for which well-established methods exist [33]. A linear fit is performed, which eliminates the need to compute gradients—a process that often leads to the vanishing gradient problem in RNNs. This approach maps the data points to the next data points in time, that can subsequently be used for prediction on new datasets. Once training is complete, predictions can be made as follows [29][31][34]:

$$\vec{r}(t+\Delta t) = tanh\left(\boldsymbol{A}\vec{r}(t) + \boldsymbol{W}_{in}\boldsymbol{W}_{out}\vec{r}(t)\right) \tag{3.10}$$

$$\vec{s}(t+\Delta t) = \boldsymbol{W}_{out}\vec{r}(t+\Delta t) \tag{3.11}$$

Now consider the Lorenz system as an example. The equations are given by [1]:

$$\dot{x} = a(y-x) \tag{3.12}$$

$$\dot{y} = x(b-z) - y \tag{3.13}$$

$$\dot{z} = xy - cz \tag{3.14}$$

A typical choice for the parameters is $a = 10$, $b = 28$, and $c = 8/3$. Equations of this type can be integrated using numerical methods, which are described in the next section.

In this case, the input vector $\vec{u} = (x, y, z)^T \in \mathbb{R}^3$ and the output vector $\vec{s} \in \mathbb{R}^3$ are both three-dimensional. The reservoir is implemented as a random Erdős-Rényi network with an average degree $\langle d\rangle = 6$ and $N = 300$ nodes and is then scaled to the spectral radius ρ (the absolute value of the largest eigenvalue). The reservoir computer is trained for $T = 100$ time units.

Figure 3.6 presents a comparison between the prediction obtained by the reservoir computer and the actual data for $0 < t < 25$ for two different spectral radii ($\rho_a = 1.2, \rho_b = 1.45$). [34]

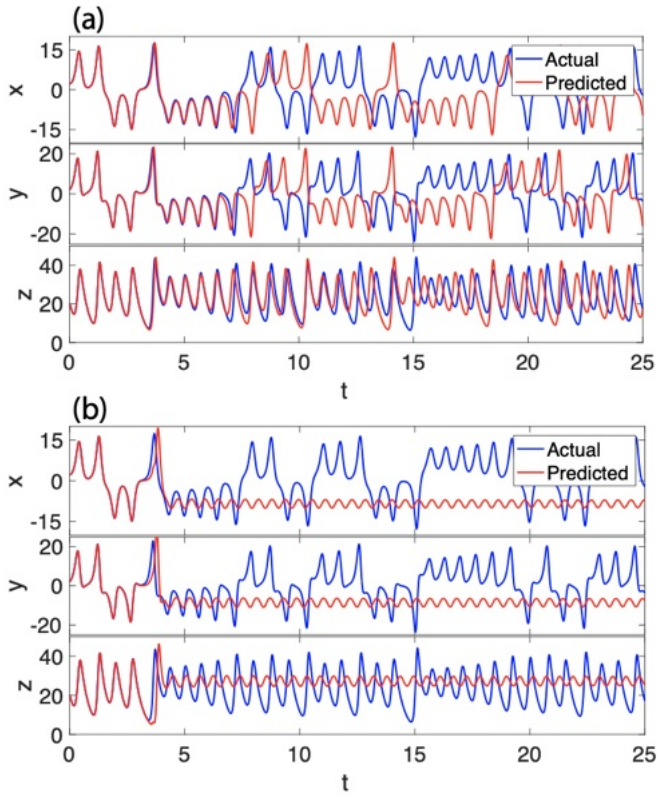


Figure 3.6: Comparison between prediction and actual data of the Lorenz system for $t = 25$; (a): spectral radius of 1.2; (b): spectral radius of 1.45 [34]

In this example, the spectral radius appears to influence the long-term accuracy of the prediction. Both cases yield good short-term predictions, which gradually deviate from the original trajectory—an expected outcome, as small errors accumulate over time. While the long-term dynamics for $\rho_a = 1.2$ seem to resemble those of the original Lorenz system, this is clearly not the case for $\rho_b = 1.45$.

Various reservoir computing architectures have been developed beyond the standard RC framework, as traditional RC can be computationally expensive, does not incorporate prior knowledge about the underlying model, and is often difficult to analyze. For example, Hybrid-RC integrates partial knowledge of underlying models [35], and Next-generation reservoir computing makes the feature space deterministic and interpretable [36]. Minimal-RC—which is the focus of this thesis—emphasizes simplicity and interpretability [2]. The aim of this work is to analyze Minimal-RC and advance its explainability by providing insights into the processes occurring during training.

3.4 Minimal Reservoir Computing

Regular reservoir computing requires the optimization of numerous hyperparameters. However, a few simple modifications to the traditional reservoir computing architecture can reduce computational resource requirements and result in significant and robust improvements in both short- and long-term predictive performance. [2]

Input weights are restructured so that all coordinate combinations are fed into the reservoir separately. Additionally, the randomness of the reservoir is eliminated by replacing it with a block-diagonal matrix composed of blocks of ones, which is subsequently scaled by the spectral radius ρ, as in regular RC. Instead of introducing nonlinearity through the activation function, higher-order terms of the reservoir states are incorporated in the readout by forming the generalized reservoir state, which is constructed by concatenating higher powers of the reservoir state with itself.

In traditional reservoir computing, the elements of $\boldsymbol{W}_{in}$ are selected as uniformly distributed random numbers within the interval $[-a, a]$. In Minimal-RC, however, the elements are not chosen randomly but are instead constructed in a structured manner. Specifically, in Minimal-RC, $\boldsymbol{W}_{in}$ is defined such that different combinations of the input data—referred to as features—are fed into the reservoir separately. Each feature is introduced into the reservoir multiple times (denoted by b), each time multiplied by a different prefactor, so that blocks of size b are formed, containing equally spaced values between $[1, 0]$.

$$\boldsymbol{w} = (w_1, w_2, ..., w_b)^T = (1, \frac{b-2}{b-1}, ..., \frac{1}{b-1}, 0)^T \tag{3.15}$$

The square roots of all weight values are taken to avoid non-invertible matrices in ridge regression: $\boldsymbol{w} = (\sqrt{w_1}, \sqrt{w_2}, \ldots, \sqrt{w_b})^T$. This results in a specifically structured input matrix. For example, if the input vector $\vec{u}(t) = (x, y, z)^T(t)$ is three-dimensional and the nonlinearity input order $\alpha = 3$, the input matrix $\boldsymbol{W}_{in}$ would be constructed as follows:

$$\boldsymbol{W}_{in}\vec{u}(t) = \begin{pmatrix} w & 0 & 0 \\ 0 & w & 0 \\ 0 & 0 & w \\ w & w & 0 \\ w & 0 & w \\ 0 & w & w \\ w & w & w \end{pmatrix} \vec{u}(t) = \begin{pmatrix} x \\ y \\ z \\ x+y \\ x+z \\ y+z \\ x+y+z \end{pmatrix} (t) \otimes w$$

$\otimes$ denotes the tensor product, meaning that each block represents one feature f. In contrast to the random Erdős-Rényi network used in regular Reservoir Computing, Minimal-RC employs a Reservoir that keeps the features separate. A block-diagonal matrix $\boldsymbol{J}$, composed of blocks of ones with block size b, is utilized. Each block $\boldsymbol{J}_i$ can thus be directly associated with a particular feature:

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{J}_1 & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{J}_2 & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{J}_{n_f} \end{pmatrix} \longmapsto \begin{pmatrix} \boldsymbol{J}_x & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{J}_y & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{J}_{x+y+z} \end{pmatrix}$$

In Minimal-RC, the spectral radius $\rho(\boldsymbol{J})$ is also scaled to a target spectral radius ρ^*. For a block-diagonal matrix of ones, the eigenvalues are equal to the block size b, so the rescaled Reservoir can be computed as follows:

$$\boldsymbol{A} = \frac{\rho^*}{\rho(\boldsymbol{J})}\boldsymbol{J} = \frac{\rho^*}{b}\boldsymbol{J} \tag{3.16}$$

In regular reservoir computing, updating the reservoir state requires a bounded nonlinear activation function, such as the hyperbolic tangent, to capture the nonlinear properties of the data. In Minimal-RC, however, the nonlinearity is entirely shifted to the readout layer, like in regular-RC. The time evolution of the reservoir states in Minimal-RC is calculated as follows:

$$\vec{r}(t+1) = g\left(\boldsymbol{A}\vec{r}(t) + \boldsymbol{W}_{in}\vec{u}(t)\right) = \boldsymbol{A}\vec{r}(t) + \boldsymbol{W}_{in}\vec{u}(t) \tag{3.17}$$

g is the activation function, which is not required in Minimal-RC. Because of the architectural design, the reservoir state for each feature can be obtained separately:

$$\vec{r}(t) = \begin{pmatrix} \vec{r}_x \\ \vec{r}_y \\ \vec{r}_z \\ \vec{r}_{x+y} \\ \vec{r}_{x+z} \\ \vec{r}_{y+z} \\ \vec{r}_{x+y+z} \end{pmatrix}(t) = \begin{pmatrix} r_{x,1} \\ r_{x,2} \\ r_{x,3} \\ \vdots \\ r_{x,b} \\ r_{y,1} \\ r_{y,2} \\ r_{y,3} \\ \vdots \\ r_{y,b} \\ r_{z,1} \\ \vdots \\ r_{x+y+z,b-1} \\ r_{x+y+z,b} \end{pmatrix}(t)$$

The segments of the reservoir states associated with a single feature are referred to as feature states and can be analyzed independently. The reservoir functions as an averaging operator on these feature states:

$$\boldsymbol{A}_f \vec{r}_f(t) = \left(\frac{\rho^*}{b} \sum_{i=1}^{b} \vec{r}_{f,i}(t) \right) \mathbf{1}_b = \rho^* \overline{r}_f(t) \tag{3.18}$$

$\mathbf{1}_b$ is a vector of ones with length b. At each iteration step, the feature states are normalized to the average of the past feature states, $\overline{r}_f(t)$, and a variable contribution—determined by the input weight—is added to the new feature data $f(t)$ [2]:

$$\overline{r}_f(t+1) = \rho^* \overline{r}_f(t) + w f(t) \tag{3.19}$$

The last row of the feature state for each feature tracks the average, or memory, since $w_b = 0$. The target spectral radius ρ^* determines how much of the memory of past data is retained at each iteration step. The nonlinearities of the system are captured exclusively in the readout layer. Generalized reservoir states $\vec{r}_g$ are constructed, and for a nonlinearity order η, they take the following form:

$$\vec{r}_g = (\vec{r}, \vec{r}^2, ..., \vec{r}^{\eta-1}, \vec{r}^{\eta})^T \tag{3.20}$$

These generalized reservoir states are then fitted to the input data using ridge regression, as in the standard reservoir computing architecture. [2]

3.5 Considerations on Minimal Reservoir Computing

Since the goal of this thesis is to analyze the Minimal-RC architecture and advance its explainability, a few considerations are outlined here.

In Minimal-RC, as in regular-RC, the entire learning process is shifted to the output matrices, making their analysis essential. For a d-dimensional system and a nonlinearity order η, the reservoir state is represented as a vector of size $l = b\alpha_n$, where b is the block size and α_n is the number of input combinations, which depends on the nonlinearity input order α and the system dimension d. For $\alpha = d$, the number of input combinations is $\alpha_n = 2^d - 1$, since the total number of elements in the power set is 2^d, but the empty set must be excluded. The output matrix is therefore expected to be a $(d \times l)$-dimensional matrix, as it must map from the generalized reservoir state to the desired output, which is a system state.

Figure 3.7 presents a visualization of an output matrix. This matrix was trained on a three-dimensional system and therefore contains three rows. Since the input nonlinearity order is $\alpha = 3$, there are seven different input combinations. With a block size of $b = 3$ and a readout nonlinearity order of $\eta = 2$, the total number of columns is $l = \eta\alpha b = 3 \times 7 \times 2 = 42$. In the plot, different matrix values are represented by different colors.

The first thing to notice is that the first row of the matrix maps to the x-coordinate of the output

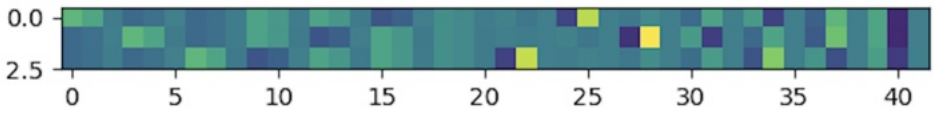


Figure 3.7: Output Matrix of three dimensional system with input nonlinearity order $\alpha = 3$ and readout nonlinearity order $\eta = 2$

state, the second row to the y-coordinate, and so on. When referring to the x part of the output matrix, this always corresponds to the first row of the output matrix. Examination of the update of the reservoir state $\vec{r}(t+1)$ in more detail for the three-dimensional case should clarify things:

$$\vec{r}(t+1) = \boldsymbol{A}\vec{r}(t) + \boldsymbol{W}_{in}\vec{u}(t) = \frac{\rho^*}{3} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} r_{x,1} \\ r_{x,2} \\ r_{x,3} \\ r_{y,1} \\ r_{y,2} \\ r_{y,3} \\ \vdots \\ r_{x+y+z,3} \end{pmatrix} (t) + \boldsymbol{W}_{in}\vec{u}(t)$$

It becomes clear that the three entries in each feature state share the same memory and differ only by the new input multiplied by w_i. When mapping the generalized reservoir state to the desired outcome, it is possible to identify which column of the output matrix corresponds to which feature space, as well as the amount of memory it utilizes by taking the absolute value as a proxy variable. For example, the first column corresponds to the feature space x in the above example and uses the entire value of the new input, since it is multiplied by $w_1 = 1$. In this case, the first 21 columns belong to the linear part of the reservoir state, while the last 21 columns correspond to the squared reservoir state.

An interesting question is whether it makes a difference to first perform the entire calculation and then extend the reservoir state, or to first extend the reservoir state (as well as the adjacency matrix and $\boldsymbol{W}_{in}\vec{u}(t)$) and then carry out the calculation:

$$\vec{r}_m(t+1) = \boldsymbol{A}_m\vec{r}_m(t) + \left(\boldsymbol{W}_{in}\vec{u}(t), (\boldsymbol{W}_{in}\vec{u}(t))^2\right) \tag{3.21}$$

Here, m stands for "modified". In the second approach, the linear part of the extended reservoir state is obviously the same. For a three-dimensional system with a block size of $b = 3$, the nonlinear part of the modified reservoir state in the second approach would be as follows:

$$\vec{r}_n(t+1) =$$

$$\boldsymbol{A}_n\vec{r}_n(t) + (\boldsymbol{W}_{in}\vec{u}(t))^2 = \frac{\rho^*}{3} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} r^2_{x,1} \\ r^2_{x,2} \\ r^2_{x,3} \\ r^2_{y,1} \\ r^2_{y,2} \\ r^2_{y,3} \\ \vdots \\ r^2_{x+y+z,3} \end{pmatrix} (t) + \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ (x+y+z)_3 \end{pmatrix}^2 (t)$$

In the original approach, one would square the entire expression $(\boldsymbol{A}\vec{r}(t) + \boldsymbol{W}_{in}\vec{u}(t))^2$, which results in cross terms between $\boldsymbol{A}\vec{r}(t)$ and $\boldsymbol{W}_{in}\vec{u}(t)$. In contrast, it becomes clear that such cross terms do not appear in the second approach.

Figure 3.8 presents a prediction of the lorenz data (10.000 steps) using the modified version of Minimal-RC. The Minimal-RC was trained for 10.000 steps, using a nonlinearity order in the generalized reservoir state of $\eta = 2$, a nonlinearity input order of $\alpha = 3$, a regularization parameter of $\beta = 1 \times 10^{-6}$, a block

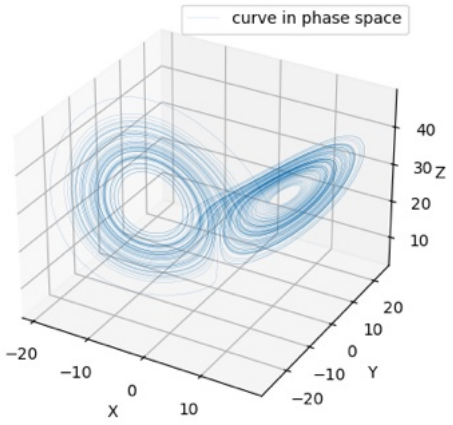


Figure 3.8: Prediction of lorenz data for the modified version of Minimal-RC

size of $b = 3$, and a spectral radius of $\rho = 1 \times 10^{-9}$.

Although this method has been shown to work, further analysis is needed to determine whether this subtle difference in the architecture affects prediction performance.

Chapter 4

Measures and Numeric Calculations

To evaluate the quality of predictions made by the reservoir computer, it is necessary to define measures that compare the actual data from the system equations with the data predicted by the machine. Additionally, differential equations must be solved to generate the time series data used for training the machine.

4.1 Runge-Kutta Method

To obtain input data for Minimal-RC, it is necessary to integrate the chosen system equations. This is achieved using an approximation technique known as the Runge-Kutta method [37][38]. The Runge-Kutta method is an iterative numerical approach for approximating the solutions of differential equations. While several variations exist, the most widely used is the RK45 method, which combines the fourth- and fifth-order Runge-Kutta approximations. This method, is also the one applied in this thesis, which is implemented in SciPy. Consider the following initial value problem to be solved:

$$\frac{dy}{dt} = f(y, t) \tag{4.1}$$

$$y(t_0) = y_0 \tag{4.2}$$

Now define:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.3}$$

$$t_{n+1} = t_n + h \tag{4.4}$$

Here, $h > 0$ represents the time step size, and k_1, k_2, k_3, and k_4 are defined as follows:

$$k_1 = f(t_n, y_n) \tag{4.5}$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right) \tag{4.6}$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right) \tag{4.7}$$

$$k_4 = f(t_n + h, y_n + hk_3) \tag{4.8}$$

Here, y_{n+1} is the RK4 approximation of $y(t_{n+1})$. The next value y_{n+1} is determined by adding the present value y_n to the weighted average of four increments. Each increment is the product of the time step size h and an estimated slope, given by the function f on the right-hand side of the differential equation. Runge and Kutta showed that their method provides an accurate approximation to the solutions of ordinary differential equations. [38]

Figure 4.1 provides a visualization of the entire procedure. k_1 is the slope at the beginning of the interval, calculated using y. k_2 is the slope at the midpoint of the interval, using y and k_1. k_3 is again the slope at the midpoint, but this time using y and k_2. k_4 is the slope at the end of the interval, using y and k_3. By averaging these four slopes, the method advances iteratively from one point on the trajectory to the next, with greater weight assigned to the slopes at the midpoint. [39]

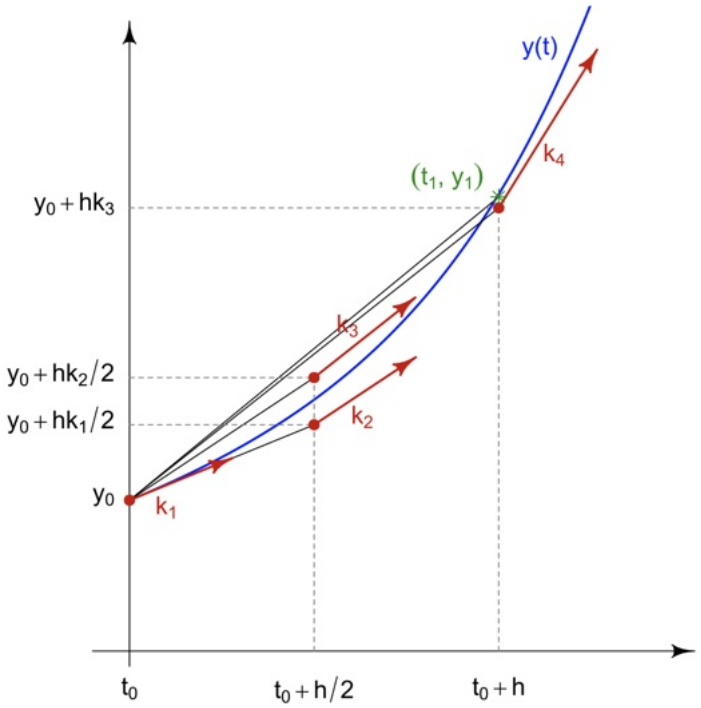


Figure 4.1: Visualization of the RK4 method [39]

The classical fourth-order Runge–Kutta (RK4) method uses a fixed step size h, which can lead to significant estimation errors if the step size is not chosen appropriately. In contrast, the Runge–Kutta–Fehlberg (RK45) method simultaneously computes fourth- and fifth-order approximations—requiring six function evaluations per step—and uses their difference to estimate the local truncation error. This error estimate enables adaptive step-size control, making RK45 generally more robust and efficient than RK4. [40]

4.2 Forecast Horizon

One important quantity for evaluating prediction quality is the so-called forecast horizon (FH) used as in [41][42]. The forecast horizon serves as a metric for comparing the original trajectory with the predicted time series. To calculate the FH, it is necessary to define a distance threshold ϵ, above which two time series are considered to have diverged.

In this context, the two time series being compared are the actual data—beginning from the moment the machine starts making predictions—and the predicted time series. Both series start from the same initial point. However, since the reservoir computer is not yet capable of perfect predictions, the two trajectories will eventually diverge after some time. Additionally, even if perfect predictions were possible, the precision of floating point numbers and their implementation in a computer can lead to divergence over time, especially given the chaotic nature of the system [43]. The FH is therefore a measure of the short-term accuracy of a prediction. When the distance between the two trajectories exceeds ϵ, they are considered divergent.

In this work, ϵ is defined as the standard deviation of the actual time series. The standard deviation is calculated separately for each dimension, and time series of each dimension is compared to its respective standard deviation. The Forecast Horizon is reached at the time when the first coordinate exceeds its threshold. This choice ensures that the threshold is adaptive: if the system occupies a larger phase space volume and exhibits greater fluctuations, the standard deviation (and thus ϵ) will be larger. Consequently, small deviations that do not significantly impact predictive quality will fall below ϵ, providing a system-appropriate tolerance. Therefore, ϵ is sensitive to the particular system and is not simply a fixed value. For multidimensional systems, each system coordinate has its own ϵ_i.

Mathematically, the Forecast Horizon can be expressed as follows. At each time step, the difference between the actual time series $\vec{u}(t)$ and the predicted time series $\vec{s}(t)$ is calculated:

$$\vec{\Delta}(t) = |\vec{u}(t) - \vec{s}(t)| \tag{4.9}$$

The forecast horizon is defined as the first time step at which, for any of the system dimensions, $\Delta_i(t) > \epsilon_i$.

The FH can be expressed either in terms of prediction steps or in Lyapunov times (1.28). When reported in Lyapunov times, the measure becomes more meaningful, as systems with higher levels of chaos are inherently more difficult to predict, resulting in a shorter forecast horizon in absolute prediction steps. By normalizing to Lyapunov times, the metric accounts for the degree of chaos in the system, providing a more system-independent assessment of predictive performance.

4.3 Correlation Dimension

Another measure for comparing the two trajectories is the correlation dimension. The correlation dimension quantifies the fractal dimension of the time series, providing a measure of complexity that characterizes the geometry and shape of a strange attractor. The correlation sum is used to estimate the correlation dimension. For a collection of points $\vec{s}_n$, the correlation sum is defined as the fraction of all possible pairs of points that are separated by less than a given distance ϵ.

The method involves centering a hypersphere on a point in phase space, allowing the radius of the hypersphere to grow until all points are enclosed, and keeping track of the number of data points contained within the hypersphere, followed by normalization. [44][45]

Similarly, the correlation sum can be interpreted as the probability that two randomly chosen points are closer than the distance ϵ. For a chaotic system, it is expected that $C(0) = 0$, since the points do not repeat in a non-periodic system. The correlation sum is calculated as follows:

$$C(\epsilon) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \theta(\epsilon - ||\vec{s}_i - \vec{s}_j||) \tag{4.10}$$

θ is the Heaviside theta function, which equals one for inputs $x > 0$ and zero for inputs $x < 0$. In the limit of an infinite amount of data ($N \to \infty$) and for small ϵ, it was found empirically, that C scales according to a power law [9]:

$$C(\epsilon) \propto \epsilon^D \tag{4.11}$$

D is the correlation dimension and can therefore be estimated by:

$$D(\epsilon) = \lim_{\epsilon \to 0} \lim_{N \to \infty} \frac{\log C(N, \epsilon)}{\log \epsilon} \tag{4.12}$$

A double logarithmic plot can be performed for various values of ϵ, and the slope of the resulting graph can be used to determine the correlation dimension. This procedure was introduced by Grassberger and Procaccia [44][45]. Figure 4.2 shows a double logarithmic plot for the logistic map at $r = 3.669\ldots$, which corresponds to the onset of chaos [4]. A single trajectory of 30,000 points was generated, starting from $x_0 = \frac{1}{2}$. A linear fit has been applied to extract the slope of the graph ($D = 0.500 \pm 0.005$), which corresponds to the correlation dimension.

In our analysis values of $\epsilon \in [1.5, 5]$ are used to calculate the correlation sum and thereby estimate the correlation dimension. These are default parameters that have been shown to work effectively for a broad range of chaotic systems.

4.4 Lyapunov Exponent

The largest Lyapunov exponent is numerically computed in our analysis. There are methods to calculate the Lyapunov exponent directly from the system equations, which is relatively straightforward. However,

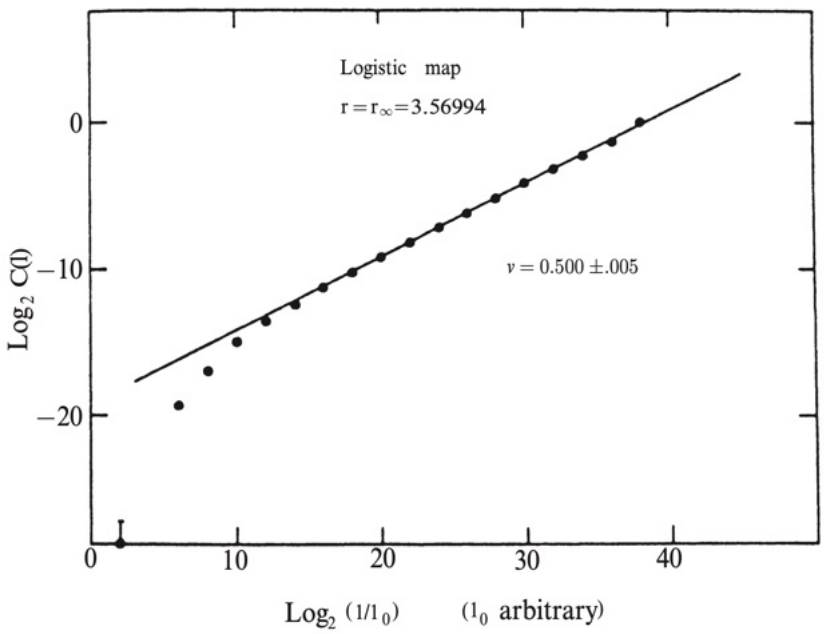


Figure 4.2: Double logarithimc plot for the Logistic map at $r = 3.669...$ [44]

since we aim to make predictions using a machine learning method that outputs a dataset, it is necessary to compute the Lyapunov exponent purely from the given data. For this purpose, the Rosenstein method is employed. [46][47]

Recall that a d-dimensional system has a distinct Lyapunov exponent for each dimension. The largest of these Lyapunov exponents determines whether the system's behavior is chaotic. The Rosenstein method estimates the largest Lyapunov exponent.

Consider a one-dimensional time series of length N, $\vec{x} = [x_1, x_2, \ldots, x_N]$. For each element of $\vec{x}$, the nearest neighbor is determined by:

$$d_i(t = 0) = \min_{x_j} ||x_i - x_j|| \tag{4.13}$$

$d_i(0)$ is the initial distance between the i^{th} point and its nearest neighbor. The closest eligible neighbor must have a temporal separation greater than the mean period of the time series, ensuring that the points are close in phase space but not too close in time. The mean period of the time series, μ, can be calculated as $1/m$, where m is the mean frequency of the power spectrum, obtained via Fourier analysis. Such an analysis is only meaningful when the data are correlated. [48]

Only the closest eligible neighbors are used in the following steps, as the focus is on how quickly nearby trajectories diverge. Points that are both close in value and in time will follow the same trajectory and will not diverge, making them irrelevant for this calculation. The Lyapunov exponent for a discrete time series is given by:

$$d(n) = d(n = 0)e^{n\lambda\delta t} \tag{4.14}$$

This can be written as:

$$\ln(d(n)) = n\lambda\delta t + \ln(d(n = 0)) \tag{4.15}$$

The separation $d(n)$ after n time steps, where $\Delta t = n\delta t$, for a pair of points is calculated as follows:

$$d_i(n) = ||x_i(n) - x_j(n)|| \tag{4.16}$$

This distance after n time steps is calculated for all eligible nearest neighbours and is than averaged. The largest Lyapunov exponent is than calculated using a least-squares fit to the "average" line defined

by [46]:

$$\lambda = \frac{1}{\Delta t} \langle \ln(d_i) \rangle_i \tag{4.17}$$

This method is easy to implement and does not require large data sets.

Chapter 5

Results

The goal of this thesis is to develop a deeper understanding of the processes happening during the training of Minimal-RC. By explaining why the architecture works—rather than simply observing that it works—the aim is to further enhance prediction accuracy and address any potential weaknesses that may still exist.

Figure 5.1 displays 30 different squares, with each tile within a square representing a distinct configuration on which the Minimal-RC has been trained for the Lorenz system. [3]

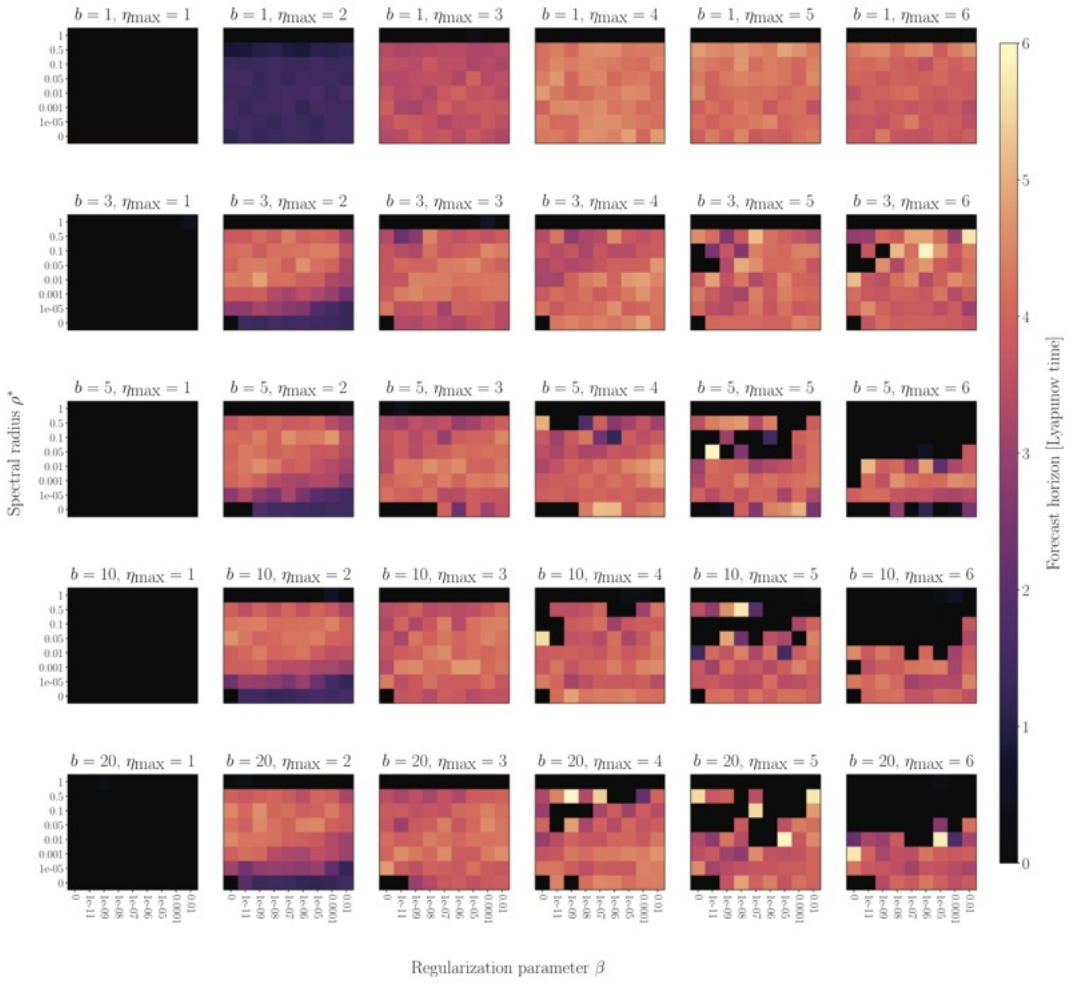


Figure 5.1: Minimal-RC performance for different configurations for the Lorenz system

In each of these blocks, the forecast horizon is plotted (encoded in the color scale) for different spectral radii on the y-axis versus regularization parameters on the x-axis. From top to bottom, the block size b increases, and from left to right, the maximum order of nonlinearity in the generalized reservoir state, η_{max}, increases by one for each square.

A notable observation is that across all values of the mentioned parameters, there is a significant improvement in predictive quality when η_{max} is increased from one to two. However, further increasing η_{max}—which corresponds to adding higher-order terms in the generalized reservoir state—does not result in substantial additional improvement in the forecast horizon. This suggests that a fundamental change occurs when moving from linear to quadratic order, enabling the Minimal-RC to learn effectively. Understanding this transition is one of the primary objectives of this thesis.

Two approaches were established at the outset of this thesis. First, the nonlinearity of the generalized reservoir state was varied in increments smaller than integer steps (Chapter 5.1). By exploring nonlinearities between one and two, the aim was to uncover novel insights into the behavior of the Minimal-RC. Second, a linear system was continuously blended with a quadratic system to gain a better understanding of the transition from linear to quadratic states (Chapter 5.2). In one approach, the nonlinearity in the system data is kept constant while the nonlinearity in the generalized reservoir state is varied. In the other approach, the nonlinearity of the system data is systematically varied while the nonlinearity of the generalized reservoir state remains fixed.

It is crucial to analyze the output matrices, as these contain the weights that are adjusted during training and thus represent where the essential processes occur. Some previously acquired knowledge about Minimal-RC has been incorporated into our simulations to enhance their efficiency. The most noteworthy results are presented in this section.

If not further specified, a regularization parameter of $\beta = 10^{-6}$ and a block size of $b = 3$ were used in all the simulations in this thesis.

5.1 Previous Knowledge about Minimal-RC

This type of analysis was conducted prior to this thesis, and the insight of this section, that minimal reservoir computers perform best when the maximum nonlinearities in the data and the generalized reservoir state are matched, was incorporated into the subsequent results. It appears that, in the introductory plot, there is no significant change in performance as η_{max} varies between one and two, with the notable improvement occuring precisely at $\eta_{max} = 2$. Furthermore, a deeper investigation revealed that it is not just the maximum nonlinearities that need to match, but rather that every term present in the system equations should, in some form, be represented in the Minimal-RC, which will be explored in greater detail in later sections (Chapter 5.7 & Chapter 5.8).

5.1.1 Fractions

As outlined in the theory section, Minimal-RC utilizes generalized reservoir states for training, which, for a given order of nonlinearity η, take the following form:

$$\vec{r}_g = \{\vec{r}, \vec{r}^2, ..., \vec{r}^{\eta-1}, \vec{r}^{\eta}\} \tag{5.1}$$

The generalized reservoir state is formed by concatenating the reservoir state with higher powers of itself.

Up to this point, only integer values have been used as nonlinearities in the generalized reservoir state. A new approach is to use fractional values as nonlinearities [3]. In our implementation, fractions are defined as objects with a numerator and a denominator, where the denominator is set to $d = 50$. Currently, only even numerators are permitted, as using odd numerators could result in complex-valued generalized reservoir states. This is because the current implementation raises the reservoir state to the power of the numerator first, and then to the power of $1/d$. An extension to complex-valued generalized reservoir states will be discussed in later chapters of this thesis (Chapter 5.5). The introduction of fractions will

enable the construction of generalized reservoir states of the form:

$$\vec{r}_g = \{\vec{r}, \vec{r}^{\frac{n}{50}}, ...\} \tag{5.2}$$

With $n \in \{50, 52, 54, ...k\}$, where k is an even number.

5.1.2 Lorenz Peak

A simulation using these fractional nonlinearities was conducted with the Lorenz system. The generalized reservoir state was systematically varied by using the regular reservoir state and concatenating it with exactly one additional power of itself. The numerator was varied from $n = 50$ to $n = 150$ in steps of two, thus the power ranged from 1 to 3. Figure 5.2 displays the forecast horizons, averaged over eight realizations for each setting and for several different spectral radii.

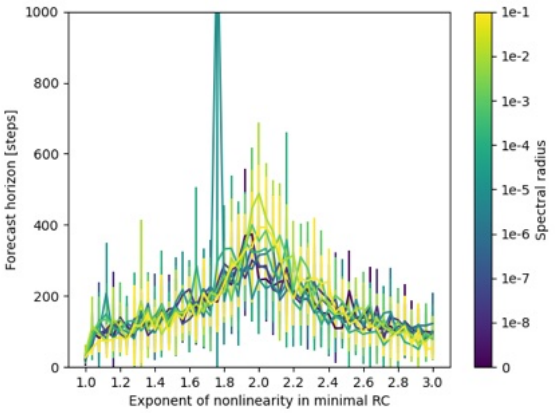


Figure 5.2: Forecast horizon plotted against nonlinearities in the generalized reservoir state for the Lorenz system

As can be observed, the forecast horizon reaches its maximum near a nonlinearity of two, which corresponds exactly to the highest nonlinearity present in the system equations of the Lorenz system [1]:

$$\dot{x} = a(y - x) \tag{5.3}$$

$$\dot{y} = x(b - z) - y \tag{5.4}$$

$$\dot{z} = xy - cz \tag{5.5}$$

It appears that the optimal nonlinearity of the generalized reservoir state is precisely two, corresponding to the highest power in the Lorenz equations. Prediction quality improves steadily as the nonlinearity approaches this value and decreases after. One outlier is observed at a nonlinearity of approximately 1.8 and a spectral radius of $\rho = 10^{-4}$. This is attributed to statistical fluctuations, which can naturally occur in such analyses.

5.1.3 Halvorsen Variation

In this simulation, the Halvorsen equations have been systematically varied [3]. A new set of equations, closely based on the original Halvorsen equations, has been defined. Specifically, the last term in each coordinate, which is squared in the standard Halvorsen system, has been replaced with the same term raised to a different power:

$$\dot{x} = -ax - 4y - 4z - y^{\frac{n}{50}} \tag{5.6}$$

$$\dot{y} = -ay - 4z - 4x - z^{\frac{n}{50}} \tag{5.7}$$

$$\dot{z} = -az - 4x - 4y - x^{\frac{n}{50}} \tag{5.8}$$

The same power was used in every dimension to preserve the cyclic symmetry of the equations. The approach was to also vary the nonlinearities in the generalized reservoir state in the same manner and then perform training for each combination. Initially, we verified whether the newly defined equations were solvable and identified which of them exhibited chaotic behavior, indicated by a Lyapunov exponent greater than zero. Figure 5.3 displays the Lyapunov exponents, encoded by color, for different values of a and various nonlinearities. Absence of color indicates that the corresponding equations were unsolvable.

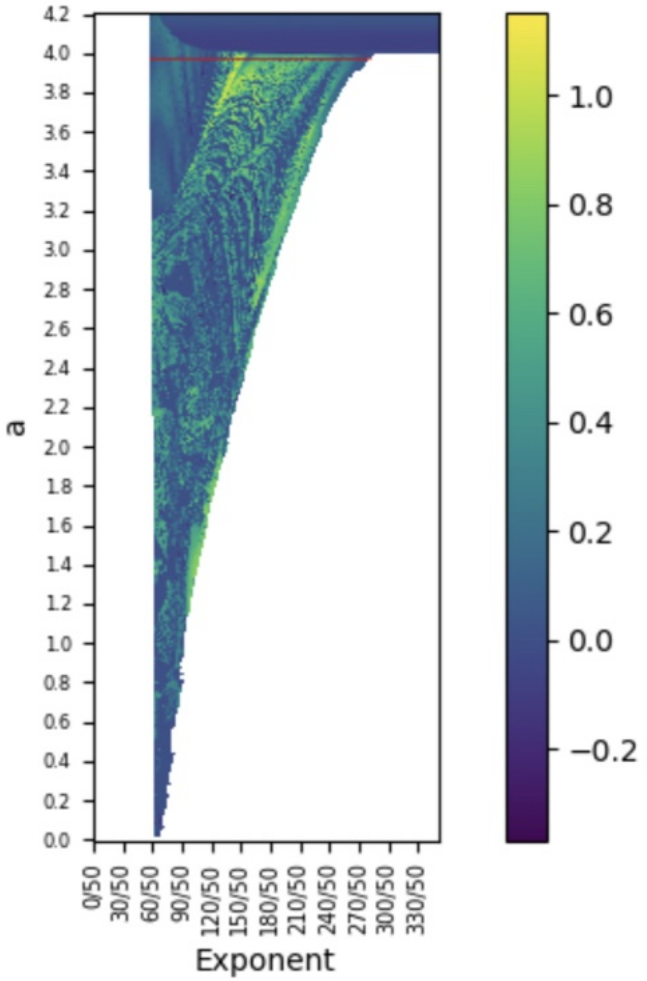


Figure 5.3: Defined Halvorsen equations: Lyapunov exponents for parameter a vs. nonlinearity

The red line indicates a regime of interest. For $a = 3.98$ (indicated by the red line) and $n = 56$ up to $n = 280$, the equations are both solvable and exhibit mostly chaotic behavior. In Figure 5.4, the forecast horizon of the predictions is presented. The y-axis represents the nonlinearity in the generalized reservoir state that is attatched to the linear reservoir state, while the x-axis corresponds to the nonlinearity in the last term of the modified Halvorsen equations.[3]

There is clear evidence that the performance of Minimal-RC is optimal when the nonlinearity of the generalized reservoir state matches the maximum nonlinearity present in the system equations. This observation is further supported by the peak seen in the results in the previous section.

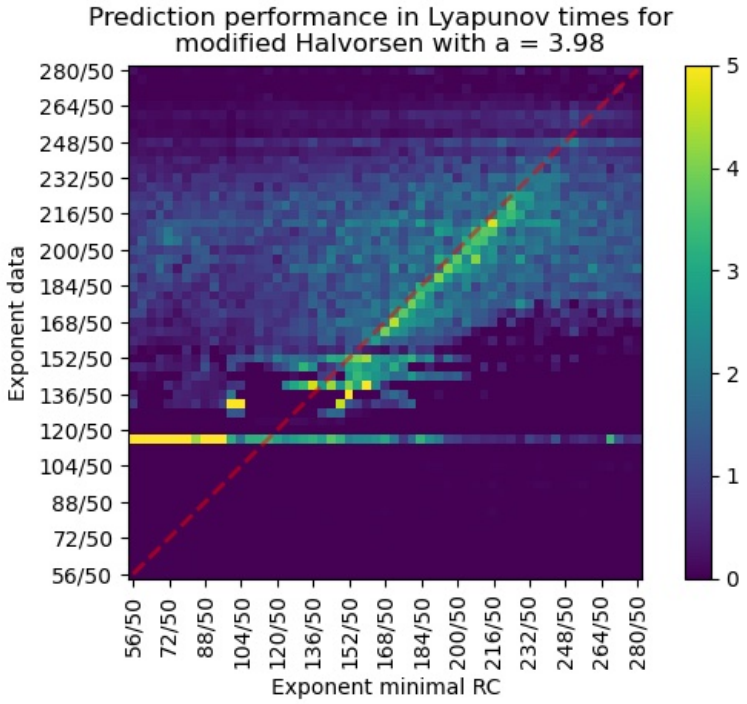


Figure 5.4: Forecast horizon in Lyapunov times for nonlinearity in generalized reservoir state plotted against nonlinearity in Halvorsen equation

5.2 Linear Process vs. Quadratic Process

The basic idea was to blend a linear process with a quadratic process by varying a parameter that controls the proportion of each system, and then to analyze the changes in the output matrices as the system shifts from being linearly biased to more quadratically biased. An unexpected issue was encountered: the mixed process could not be predicted by Minimal-RC.

Initially, we systematically combined a linear process with one of the established systems exhibiting a maximum nonlinearity of quadratic order. For this purpose, we used generalized reservoir states of both linear and quadratic order, which should, in principle, be sufficient to capture the dynamics of a purely linear system, a purely quadratic system, and any mixture of the two. For the linear process, we employed either an autoregressive process, a self-defined elliptical system, or a system referred to as the fully linear system, all of which exhibit strictly linear behavior.

These linear systems were systematically mixed with one of the known chaotic systems, such as the Lorenz system or the Halvorsen system. Each coordinate of the linear system ($\{x_n\}$, $\{y_n\}$, $\{z_n\}$) was combined with the corresponding coordinate of the quadratic system using the following equation [49]:

$$x_{new_n} = m * quadratic_n + (1 - m) * linear_n \tag{5.9}$$

By varying m from zero to one, we obtain linear data that is systematically mixed with quadratic data. The parameter m is incremented in steps of 0.01, and the standard minimal reservoir computing architecture is used to train the machine and make predictions. The results of one representative example are shown here. Additional results can be found in the appendix [B.1] for the interested reader.

All combinations of linear system and quadratic system consistently lead to the same finding: as m is varied from zero to one, there is a clear shift in the active matrix weights from the linear part to the quadratic part of the output matrix. Stronger activation of weights means that their absolute values become larger, regardless of sign. The example presented here is a mixture of the fully linear system and the Halvorsen system. The fully linear system is defined as follows [50]:

$$\dot{x} \;=\; sin(y) \tag{5.10}$$

$$\dot{y} = x + z \tag{5.11}$$

$$\dot{z} = x - y \tag{5.12}$$

Figure 5.5 displays the attractors resulting from this mixing, visualized for several values of m, and plotted over 50,000 time steps with $dt = 0.01$ each.

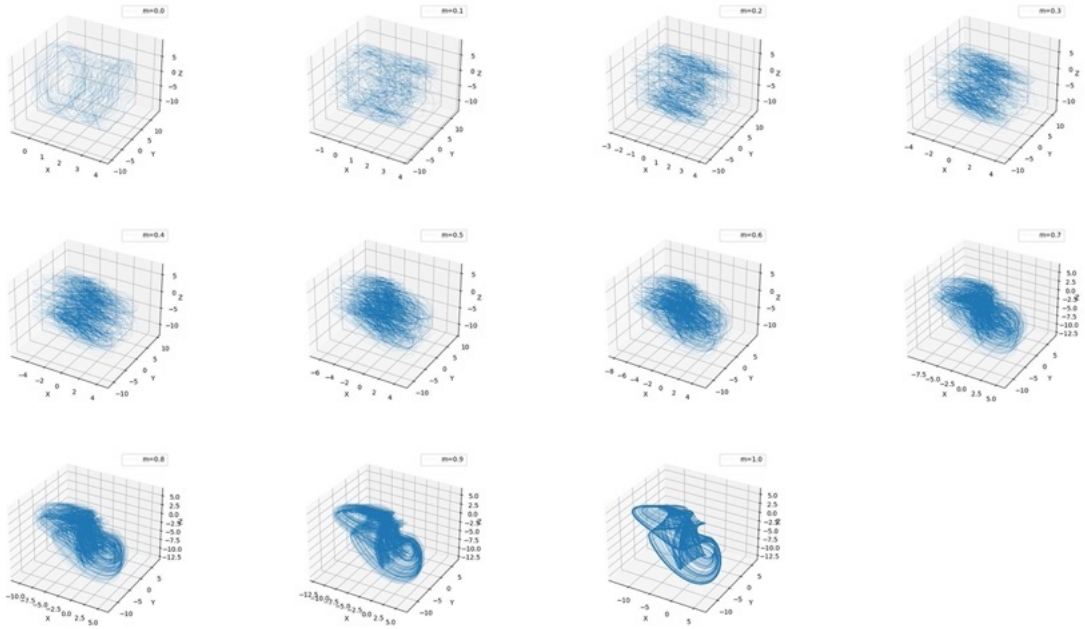


Figure 5.5: Attractors of fully linear system - Halvorsen system

For $m = 0$, the trajectory corresponds to the fully linear system, while for $m = 1$, the attractor is the pure Halvorsen system. Figure 5.6 shows the average forecast horizon over five realizations for varying m and different spectral radii.

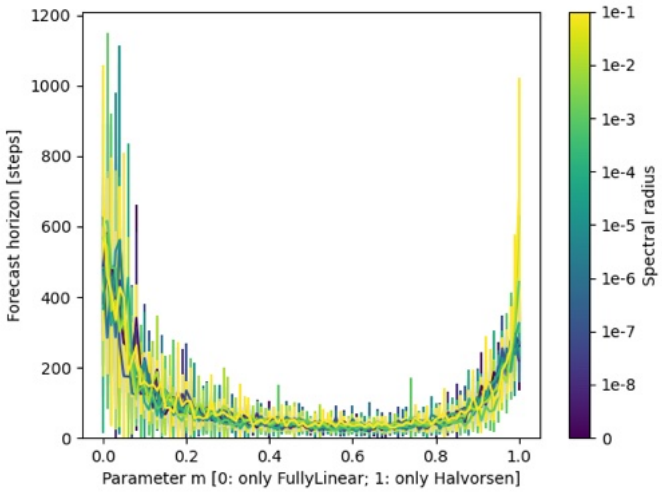
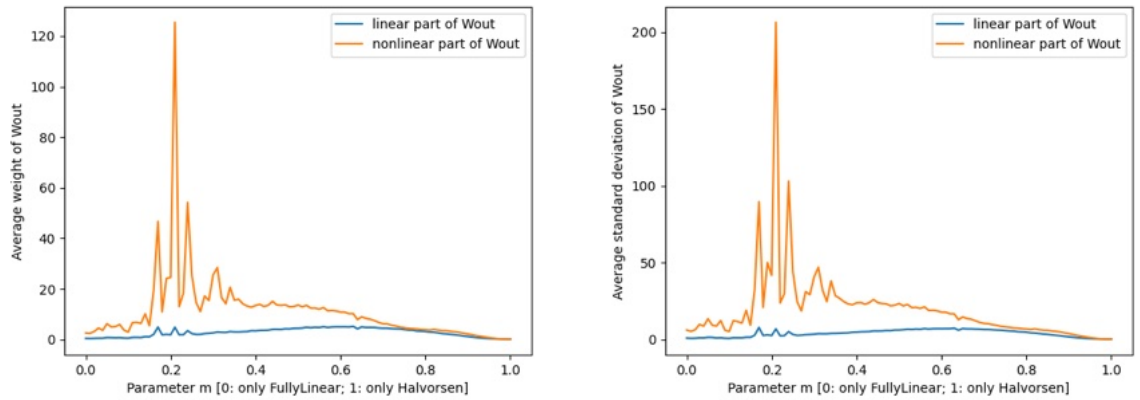


Figure 5.6: Forecast horizons of fully linear system - Halvorsen system for different spectral radii

The first observation is that, across all spectral radii, only the pure systems are predicted accurately.

For the mixed attractors, there is a significant reduction in the forecast horizon.

Figure 5.7 presents two statistics of the output matrices for the mixing of the two processes. The figure shows the average of the absolute values of all matrix weights corresponding to either the linear part or the quadratic part, as well as the standard deviation of these values for both the linear and quadratic sections of the output matrices. With a block size of $b = 3$, all possible combinations in the input matrix, and the linear reservoir state concatenated with its square, the resulting generalized reservoir state has length $3 \times 7 \times 2 = 42$. Consequently, the output matrices are of size 3×42. The first 21 columns correspond to the linear part of W_{out}, while the last 21 columns represent the quadratic part. The results for the spectral radius of $\rho = 10^{-7}$ are plotted here.



(a) Average of weights of W_{out}

(b) Average standard deviation of weights of W_{out}

Figure 5.7: Statistics of matrix weights for all values of m

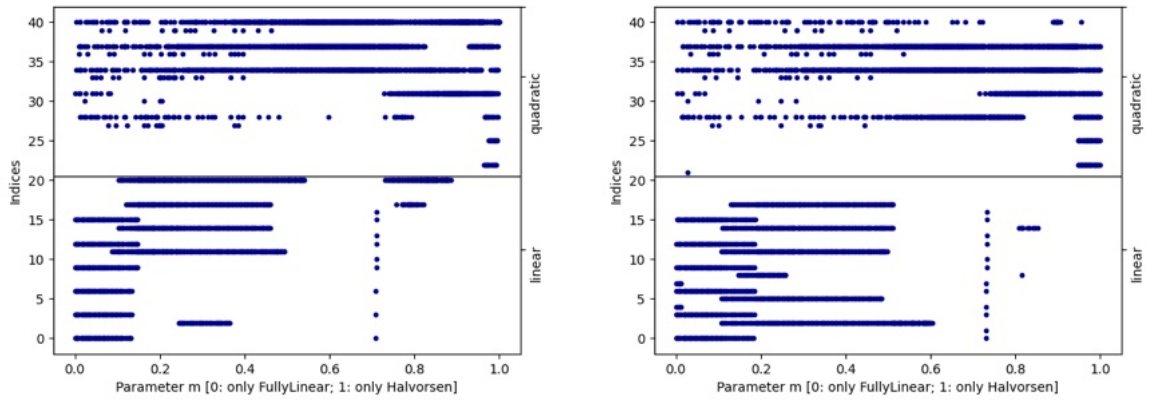
The weights, as well as their standard deviation, appear to increase in cases of poor prediction. When the minimal reservoir computer is unable to learn effectively, the matrix weights tend to diverge. For the pure systems, both the matrix weights and their standard deviation approach smaller, more stable values. This phenomenon has been observed across various systems and scenarios: when prediction quality decreases, the matrix weights often become extremely large and unstable.

Interestingly, the quadratic part of W_{out} appears to dominate even for the fully linear system, which is unexpected. One would anticipate stronger linear activation when predicting a linear time series. It is important to note that here we are considering the average values of the entire linear and quadratic parts, which may mask specific details. A single large value in the quadratic part can significantly increase the overall average. Since matrix weights tend to diverge when learning is unsuccessful, this effect likely explains the behavior observed in the quadratic part of the output matrices when predicting a linear system. Future analyses, such as implementing an outlier exclusion procedure, could provide clearer insights.

Figure 5.8 provides a more detailed analysis of the W_{out} activation during the mixing process, clearly illustrating the shift from linear activation to quadratic activation as m increases from zero to one.

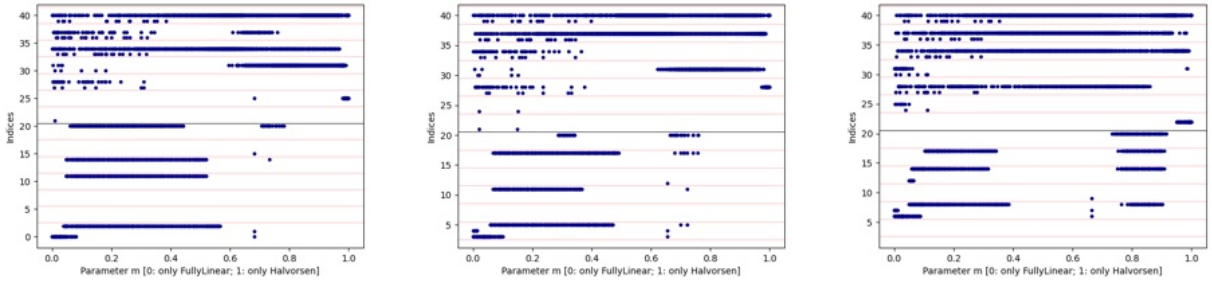
The y-axis represents the indices of the columns of the output matrices, while the x-axis corresponds to the values of m. For each of the 42 columns, the absolute weights have been summed and their standard deviation calculated. The maximum value among these has been identified, and all columns with values within 70% of the maximum have been plotted. This procedure was repeated for every value of m. As before, indices less than or equal to 21 correspond to the linear part of W_{out}, while indices greater than 21 belong to the quadratic part. The transition from linear activation to quadratic activation as m increases is clearly visible.

Figure 5.9 displays the column activation for each coordinate separately. The first row of the output matrix maps to the x-coordinate, the second row to the y-coordinate, and the third row to the z-coordinate. For each row, the maximum absolute value was determined, and all column indices of the



(a) Sum of absolute weights (b) Standard deviation

Figure 5.8: Activations of columns of W_{out} for all values of m



(a) Maps to x Coordinate (b) Maps to y Coordinate (c) Maps to z Coordinate

Figure 5.9: Indices of activated matrix weights for each coordinate for varying m

matrix weights within 70% of this maximum were plotted for each coordinate. The shift from linear to quadratic activation is evident in each matrix row. Additionally, for the pure Halvorsen system, there appears to be a cyclic symmetric activation in the matrix weights, which is consistent with the structure of the system equations. This observation will be examined in greater detail later (Chapter 5.8).

5.3 Mixing of two Quadratic Processes

Since the predictions for the mixing of the fully linear system and the Halvorsen system—as well as for all other scenarios where a linear system was mixed with a quadratic system (see appendix [B.1])—were very poor, we next attempted to mix two systems with the same maximum nonlinearity in their governing equations, both of which are known to yield good prediction results. These systems both achieve optimal predictions for the same order of nonlinearity in the generalized reservoir state. Specifically, we mixed two systems with similar Lyapunov exponents (the Lorenz and Halvorsen systems) and two systems with very different Lyapunov exponents (the Lorenz and Roessler systems).

Figure 5.10 displays the attractors resulting from the mixing of the Lorenz system with the Halvorsen system.

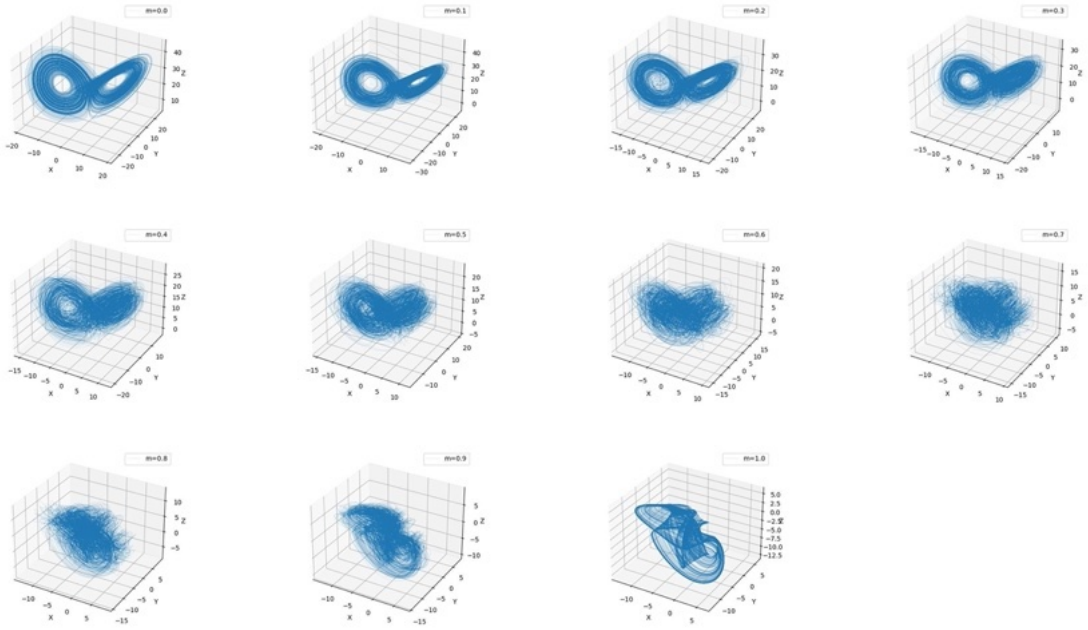


Figure 5.10: Attractors of Lorenz vs. Halvorsen

Figure 5.11 shows the flow resulting from the mixing of the Lorenz system with the Roessler system, whose equations are defined as follows [51]:

$$\dot{x} = -y - z \tag{5.13}$$

$$\dot{y} = x + ay \tag{5.14}$$

$$\dot{z} = b + z(x - c) \tag{5.15}$$

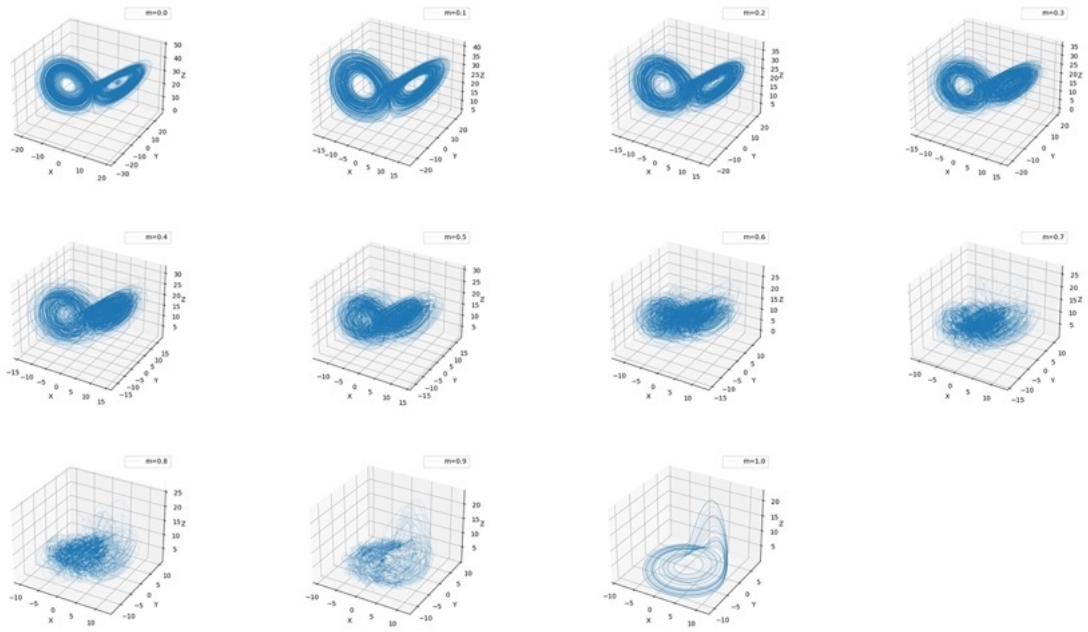


Figure 5.11: Attractors of Lorenz vs. Roessler

Figure 5.12 presents the forecast horizon for the Lorenz system mixed with the Halvorsen system, while Figure 5.13 displays the forecast horizon for the Lorenz system mixed with the Roessler system. Due to the occurrence of outliers, the forecast horizon was capped at a value of 800, and the maximum error was limited to 50 in Figure 5.13.

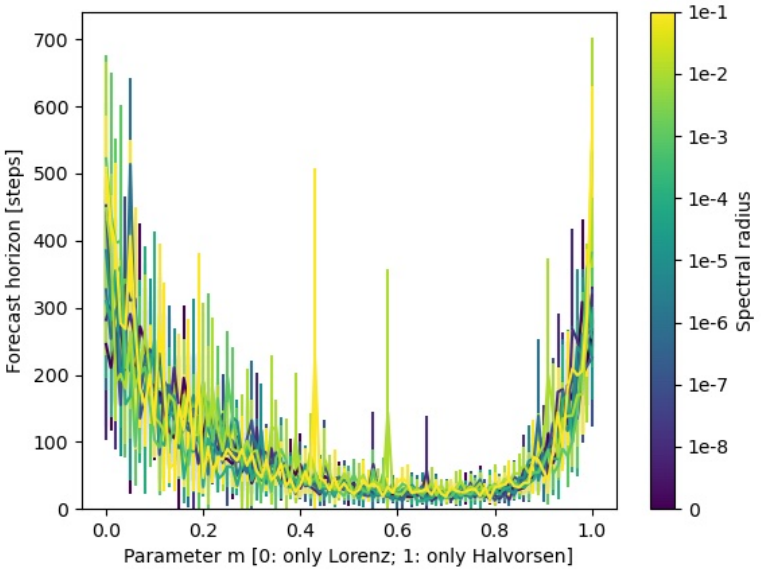


Figure 5.12: Forecast horizon of Lorenz - Halvorsen for different spectral radii

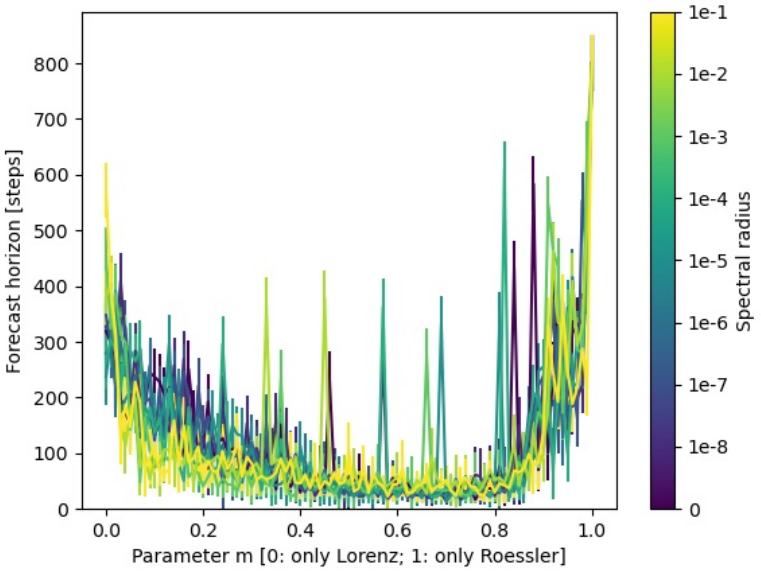


Figure 5.13: Forecast horizon of Lorenz - Roessler for different spectral radii

The results are both astonishing and unexpected, given that each system can be predicted by the minimal reservoir computer and both possess the same highest order of nonlinearity in their governing equations. However, when the systems are mixed in this manner, prediction quality deteriorates significantly. Furthermore, there appears to be little difference between mixing two systems with similar Lyapunov exponents and mixing systems with very different Lyapunov exponents. Numerous approaches

were attempted, such as rescaling or remapping both systems prior to mixing, but these strategies did not resolve the issue of the so-named "Valley of Unpredictability". The cause of this phenomenon and potential solutions remain unclear. The following sections aim to address and clarify this issue.

5.4 Valley of Unpredictability

We attempted to use higher nonlinearities in the generalized reservoir state. One possible explanation for the poor predictions is that, by mixing two systems, the maximum nonlinearity present in the data may increase in some way. To investigate this, we varied the nonlinearities from 50/50 to 200/50, incrementing the numerator by two, and calculated the Forecast horizon for each value of m over five realizations.

Figure 5.14 shows the forecast horizon, encoded in the color scale, plotted as a function of the nonlinearity in the generalized reservoir state (regular reservoir state concatenated with one higher power) and the mixing parameter m for the mixing of the fully linear system and the Halvorsen system. A spectral radius of $\rho = 10^{-3}$ was used in this plot, but the results are similar across many values of ρ.

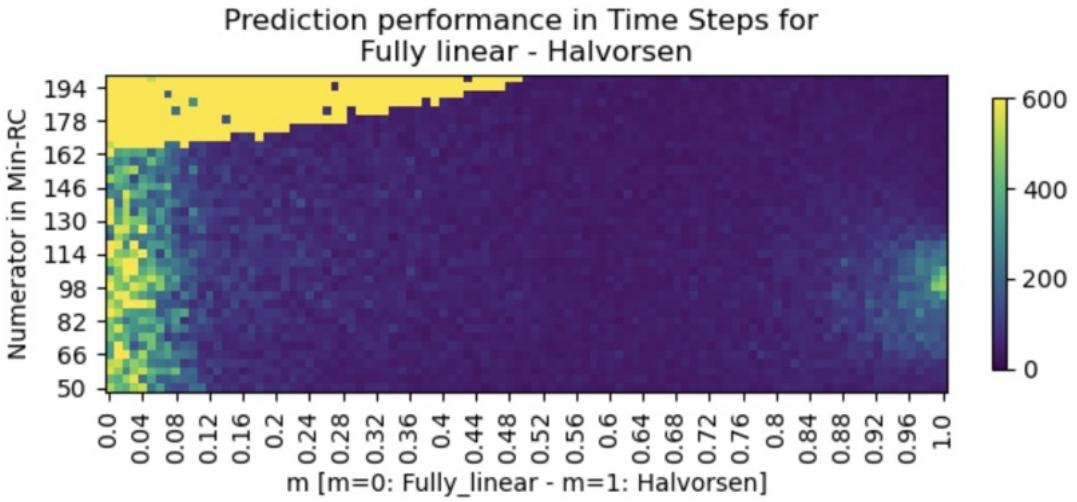


Figure 5.14: Forecast horizon of Fully linear - Halvorsen for different nonlinearities vs. m

The seemingly good predictions observed in the upper left corner can be disregarded, as they result from spurious effects and are not meaningful.

The purely linear system demonstrates good predictions across all nonlinearities. This is likely because, for the linear system, it is sufficient to use the regular or linear reservoir state. Previous sections also showed activation of output matrix weights in the quadratic part for purely linear systems. However, this was very likely because the Minimal-RC attempts to utilize the superfluous matrix weights, leading it to learn spurious or irrelevant patterns. The pure Halvorsen system exhibits optimal prediction performance precisely at the quadratic order in the generalized reservoir state, as expected. Importantly, no improvement in prediction quality is observed for the mixed data, even when higher powers are used in the Generalized Reservoir State.

Figure 5.15 presents the forecast horizon for the mixing of the Lorenz system and the Halvorsen system, with the nonlinearities plotted against m for a spectral radius of $\rho = 10^{-3}$. Again, the spurious effects in the upper left corner should be disregarded.

As can be seen, there is a peak in prediction quality for the pure systems at a nonlinearity of two, which matches the highest order present in the system equations. However, no improvement is observed for the mixed systems when using higher powers in the generalized reservoir state.

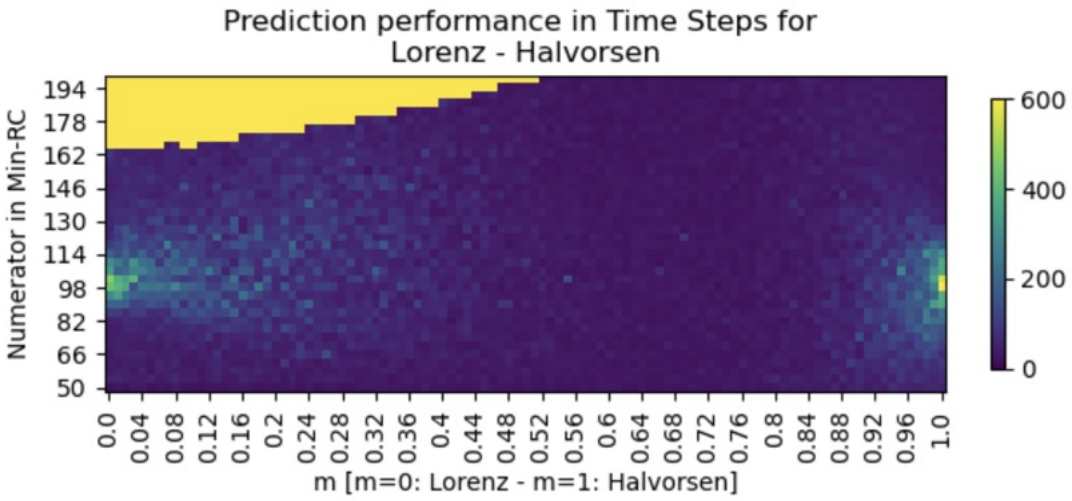


Figure 5.15: Forecast horizon of Lorenz - Halvorsen for different nonlinearities vs. m

Next, we applied the standard reservoir computing architecture to this particular problem, as shown in Figure 5.16.

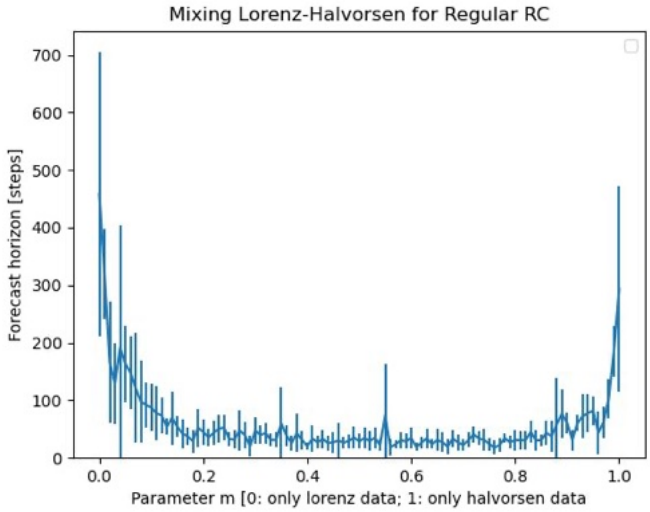


Figure 5.16: Forecast horizon of Lorenz - Halvorsen for regular reservoir computing

Once again, the same pattern emerges. The standard parameter configuration from the SCAN package [52], was used in this analysis.
A hyperparameter optimization has been conducted for the regular RC architecture, which did not lead to any significant improvement for the mixed systems.

A particularly interesting finding emerged when the system data were rescaled to values in the interval $[-1, 1]$ for each coordinate. Visually, the attractor appears identical to the regular Lorenz attractor, except that the values are constrained to the interval $[-1, 1]$ in each coordinate. However, the predictions have become significantly worse, indicating that rescaling each coordinate independently disrupts the correlations between coordinates and leads to a loss of well-defined system equations. This seemingly minor modification has a substantial impact on predictability, suggesting that having clearly defined equations in the data is essential for accurate predictions—at least when using generalized reservoir

states of the form [1, higher power].

It is highly likely that improved predictions could be achieved by incorporating more than one higher power in the generalized reservoir state. In particular, selecting the appropriate combination of higher powers may enable the identification of all relevant (and potentially very complex) nonlinearities and their interactions in the system equations. This approach would be a necessary step toward predicting more complex data, with potential real-world applications.

5.5 Complex Extension

A necessary step towards allowing complete freedom in the choice of fractions is to extend the Minimal-RC architecture to complex numbers. A challenge encountered prior to this extension was the occurrence of diverging numerical effects and numerically infinite values in the generalized reservoir state, which caused the training procedure to fail. Previously, exponentiation of the reservoir state was performed by first raising it to the power of $1/50$ and then to the power of n, where n is the numerator of the fraction. If the values in the reservoir state became sufficiently small, raising them to the $1/50$ power resulted in values so close to zero that subsequent operations produced diverging numerical effects, which persisted even after raising to the power of n afterwards.

One possible workaround would be to first raise the reservoir state to the power of n and then to $1/50$. However, if the values become sufficiently large, exponentiation with n can produce numerically infinite values, which also persist through subsequent operations. To maximize flexibility, we decided to allow for complex generalized reservoir states. In this approach, the state is raised directly to the power of $n/50$ in a single step, which can result in complex values when taking roots of negative numbers.

Four different approaches were tested. One was inspired by sources found on the Internet [53] (Chapter 5.5.1), while the other three were developed independently.

5.5.1 Complex Ridge Regression - Option 1

Consider the equation to solve for $\boldsymbol{W}_{out}$:

$$\boldsymbol{W}_{out} = \vec{r}^T \vec{s}_d (\vec{r}^T \vec{r} + \beta \mathbf{1})^{-1} \tag{5.16}$$

$\vec{s}_d$ is the target output. The extension to complex generalized reservoir states is straightforward in this case. One simply replaces the transpose vector $\vec{r}^T$ with the adjoint vector $\vec{r}^\dagger$, resulting in:

$$\boldsymbol{W}_{out} = \vec{r}^\dagger \vec{s}_d (\vec{r}^\dagger \vec{r} + \beta \mathbf{1})^{-1} \tag{5.17}$$

The generalized reservoir state remains complex-valued during both the prediction procedure and the training loop.

For each of the four approaches, datasets were trained for every element of the powerset of $X = [50, 60, \ldots, 140, 150]$, using these as numerators for the generalized reservoir state. The denominator was again fixed at 50, even though the distinction between numerator and denominator becomes unnecessary when working with complex values. Each possible combination of elements from X was trained over five realizations. Table 5.1 presents the combinations of nonlinearities drawn from X in the generalized reservoir state for four systems that achieved the best forecast horizons.

System	Best FH	Best combination
Lorenz	447.8	50,100,120,150
Halvorsen	304.8	50,100,150
Fully linear	1421.8	50,70
Lor - Hal	52.8	50,120,140,150

Table 5.1: Best combinations of nonlinearities and their forecast horizon

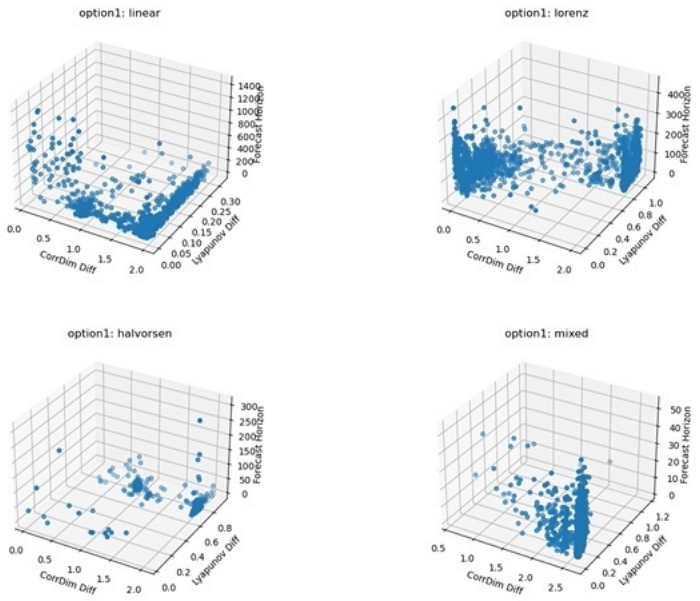


Figure 5.17: Prediction qualities for all possible combinations visualized - Option 1

Figure 5.17 displays the qualities, of the predictions, for all possible combinations out of X visualized.

The x-axis represents the difference in correlation dimensions between the real and predicted system, the y-axis shows the difference in their Lyapunov exponents, and the z-axis indicates the forecast horizon. Predictions are considered better the closer they are to the upper left corner of the plot. It can be observed that the fully linear system and the Lorenz system yield some good predictions, while the majority of predictions for the Halvorsen system are poor. The mixed system does not exhibit any good predictions. However, the difference in Lyapunov exponents is small, which may indicate that both the mixed system and its prediction exhibit highly chaotic behavior, resulting in only a minor discrepancy between their exponents.

5.5.2 Separate Output Matrix - Option 2

The idea is to fit the real part and the imaginary part of $\vec{r}$ separately, as follows:

$$\boldsymbol{W}_{out-real} = \vec{r}_{real}^{T}\vec{s}_d(\vec{r}_{real}^{T}\vec{r}_{real} + \beta\mathbf{1})^{-1} \tag{5.18}$$

$$\boldsymbol{W}_{out-imag} = \vec{r}_{imag}^{T}\vec{s}_d(\vec{r}_{imag}^{T}\vec{r}_{imag} + \beta\mathbf{1})^{-1} \tag{5.19}$$

The output matrix is then constructed as follows:

$$\boldsymbol{W}_{out} = \boldsymbol{W}_{out-real} + i\boldsymbol{W}_{out-imag} \tag{5.20}$$

The generalized reservoir state remains complex-valued during the prediction procedure as well. This method, however, did not succeed in producing good predictions.

5.5.3 Only Real Part for Training - Option 3

This method permits complex values in the generalized reservoir state during the training loop. However, when fitting the output matrix W_{out}, only the real parts of the generalized reservoir states are utilized. Consequently, it is essential to use only the real part of the generalized reservoir state during the prediction procedure as well. Table 5.2 presents the combinations of nonlinearities selected from X in the generalized reservoir state for four systems that achieved the best forecast horizons. Some unexpected combinations yield very good predictions. Figure 5.18 displays the qualities, of the predictions, for all possible combinations visualized.

System	Best FH	Best combination
Lorenz	753.6	60,70,80,90,110,130,140,150
Halvorsen	571.0	50,60,70,90,110,120,130,140,150
Fully linear	3044.8	70,80,90,100,110,120,130,140,150
Lor - Hal	73.4	50,100,120

Table 5.2: Best combinations of nonlinearities and their forecast horizon

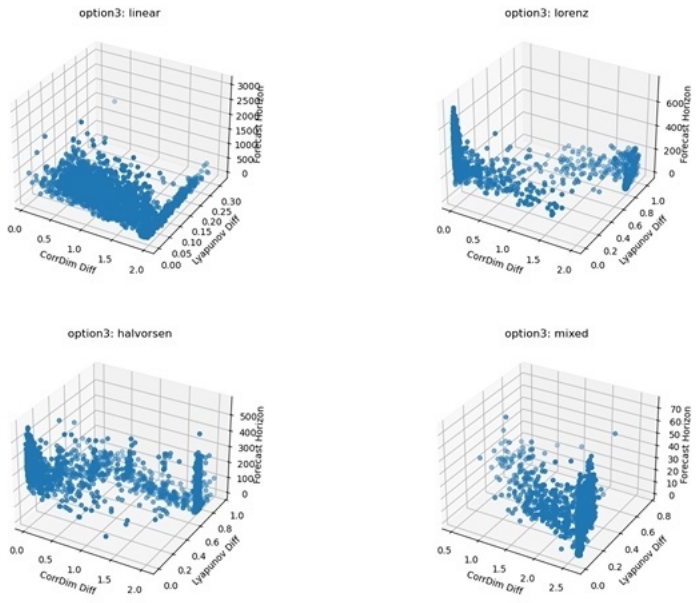


Figure 5.18: Prediction qualities for all possible combinations visualized - Option 3

In contrast to Option 1, the Halvorsen system can be predicted much more accurately, with significantly more points appearing in the upper left corner. However, the mixed system still results in poor predictions.

5.5.4 Real and Imaginary Part for Training - Option 4

As with method 3, option 4 also allows for complex values in the generalized reservoir state during the training loop. However, when fitting the output matrix, both the real and imaginary parts of the generalized reservoir state are used, but without including the complex unit. This is accomplished by concatenating the real part of the generalized reservoir state with its imaginary part, resulting in a real-valued generalized reservoir state for the prediction procedure as well. Table 5.3 presents the combinations of nonlinearities from X in the generalized reservoir state for four systems that achieved the best forecast horizons. Even better predictions are achieved with Option 4 compared to Option 3. Figure

System	Best FH	Best combination
Lorenz	779.0	50,60,80,90,110,120,130
Halvorsen	589.2	60,90,120,130
Fully linear	6477.8	50,60,70,90,100,110,120,130,140,150
Lor - Hal	77.4	70,100,120,130,140,150

Table 5.3: Best combinations of nonlinearities and their forecast horizon

5.19 displays the qualities, of the predictions, for all possible combinations visualized.

The predictions with option 4 are even better than those obtained with option 3, however, the mixed system still results in poor prediction quality.

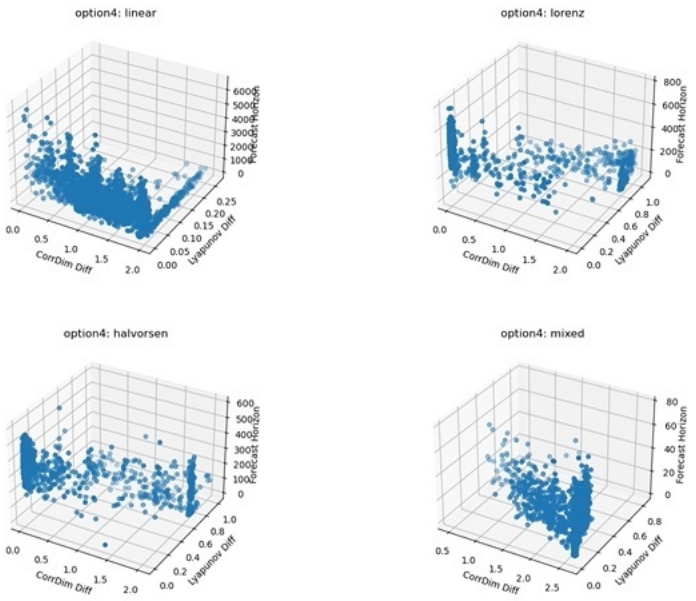


Figure 5.19: Prediction qualities for all possible combinations visualized - Option 4

An unexpected phenomenon arises from the use of complex numbers: an unusual combination of nonlinearities achieves the best prediction quality for both option 3 and option 4. It appears that the complex component may compensate for the mismatch between the nonlinearities present in the data and those in the Minimal-RC.

Since option 4 yields the best results, this method will be used for all further analysis of the complex extension of the Minimal-RC.

5.6 Predicting Mixed Systems with additional Information and Time Delay Embedding

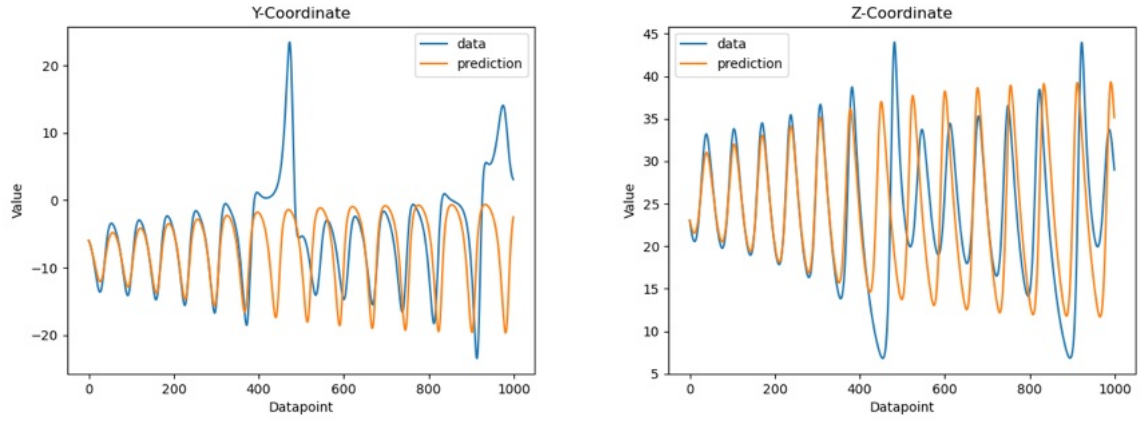
One hypothesis is that it is not sufficient for only the highest power in the system equations to match the generalized reservoir state. Rather, all terms present in the governing equations should be represented in the Minimal-RC. For complex systems, such as the mixed Lorenz-Halvorsen system, this may require including a large number of nonlinearities in the Minimal-RC. Consequently, it is necessary to analyze a set larger than $X = [50, 60, \ldots, 140, 150]$ from the previous section.

However, the challenge lies in the fact that the powerset of a set with n elements contains 2^n possible combinations, making an exhaustive search computationally infeasible for larger sets. We attempted to restrict the search to all possible combinations with a maximum of three elements, but this approach was unsuccessful for predicting the mixed system.

Mathematical considerations suggest that the way we mix the systems results in a loss of information. Specifically, we reduce a six-dimensional dataset to a three-dimensional one by not using the last data point as input for the next, but instead independently training the two systems and then mixing them, which leads to the loss of bijectivity in the underlying function.

We attempted to train the Lorenz system and the Halvorsen system using only two coordinates as input and predicting only those two coordinates. This serves as a simpler case, where some information is absent and bijectivity is lost. Both systems were tested with one coordinate removed. Figure 5.20 illustrates the case for the Lorenz system with the x-coordinate removed. The other cases are provided

in the appendix section [B.2]. Overall, the Lorenz system with one coordinate removed was predicted more accurately than the Halvorsen system. However, no in-depth analysis was conducted, as the main focus is on predicting the mixed system. It became apparent that the overall prediction quality remains poor, even for the simpler two-dimensional case.

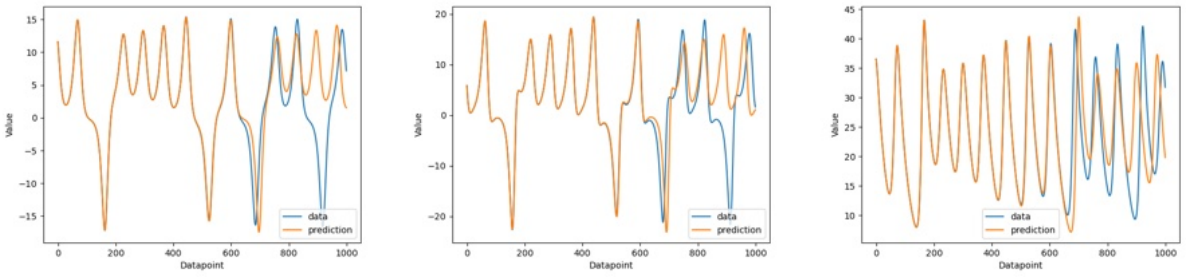


(a) First coordinate (b) Second coordinate

Figure 5.20: Lorenz system with removed x-coordinate with [1,2,3,4,5,6,7] as nonlinearities and $\rho = 10^{-9}$

Some trends can be captured by the Minimal-RC, especially at the beginning of the prediction. However, there is no perfect replication, since information from the x-coordinate is missing. This information is important for the overall prediction. In theory, nonlinearities up to quadratic order should be sufficient to predict the Lorenz system. However, it is not clear how the removal of one coordinate affects the underlying structure. To potentially improve prediction quality, nonlinearities of [1, 2, 3, 4, 5, 6, 7] were chosen for the generalized reservoir state instead of using only purely linear and quadratic terms.

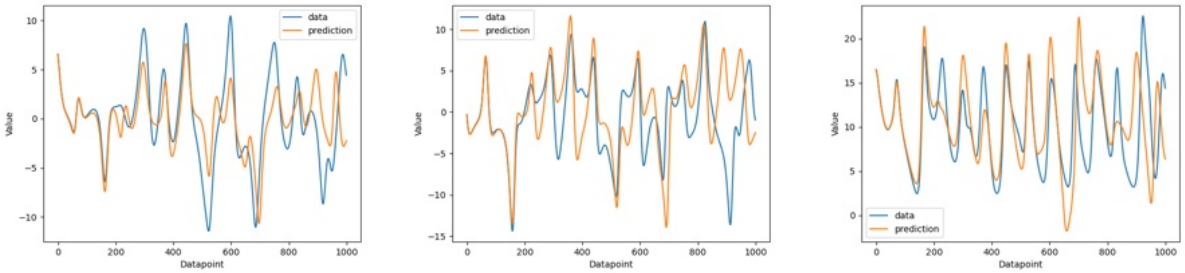
It was possible to predict the mixed system by including one of the two individual systems as additional input, thereby using a six-dimensional input. The first three dimensions were selected to represent the individual system, while the second three dimensions were chosen to represent the mixed system. The individual system must be the exact same dataset used to generate the mixed system. It is important to note that increasing the number of input dimensions allows for more possible input combinations. For these experiments, we restricted the maximal input nonlinearity order to three. Figure 5.21, Figure 5.22, and Figure 5.23 present the results for the Lorenz system and the mixed system, using a maximum input nonlinearity order of 3, a generalized reservoir state of [1, 2, 3], and $\rho = 10^{-6}$.



(a) X coordinate (b) Y coordinate (c) Z coordinate

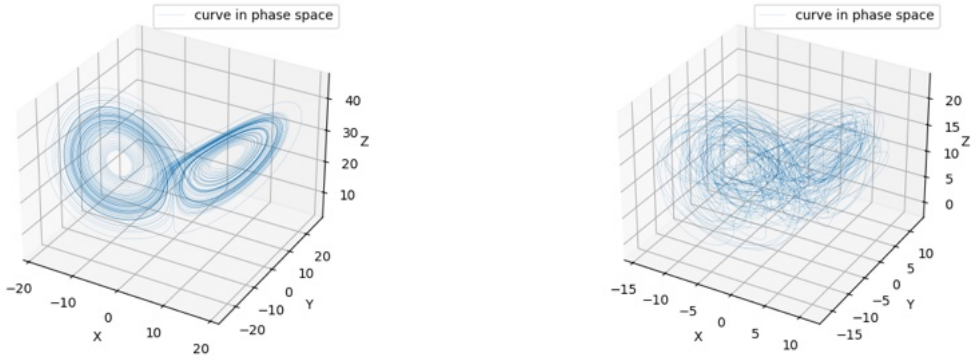
Figure 5.21: Prediction of the first three coordinates (1000 steps) - Lorenz System

Good results were also obtained when using the Halvorsen system instead (see appendix [B.3]). It is interesting to observe that, by including the information from one of the systems in the mix, the computer is able to separate the mixed system into its original components and produce good predictions.



(a) X coordinate (b) Y coordinate (c) Z coordinate

Figure 5.22: Prediction of the last three coordinates (1000 steps) - Mixed system



(a) Lorenz system (b) Mixed system

Figure 5.23: Prediction of Lorenz- and Mixed system as six dimensional input

A problem occurs when both individual systems and the mixed system are used together as a nine-dimensional input. The number of possible input combinations is 2^n, where n is the number of coordinates. With nine input coordinates, this results in 512 possible combinations. We attempted to train the machine using all 512 combinations. This process was time-consuming and ultimately led to diverging trajectories. Even when restricting the maximum order of input nonlinearity to smaller values, the results still diverged. The Minimal-RC attempts to learn from all of these input combinations, which causes it to pick up irrelevant or spurious patterns along with potentially useful information.

Since the approach of adding information from one of the original systems to the mixed system amounts to a form of "cheating"—as it assumes prior knowledge of the system, which is not typically available in real-world applications—we developed an alternative solution using time delay embedding. As described in the theory section, time delay embedding allows for the reconstruction of any attractor by using delayed coordinates.

Given that Minimal-RC encounters difficulties when working with nine-dimensional inputs, and that using a large number of delay coordinates is feasible, we decided to employ regular reservoir computing for this problem instead. By optimizing the lag and the delay parameter τ in the embedding, good predictions were achieved using this approach [54]. Figure 5.24 illustrates the improvement in prediction quality achieved by using time delay embedding.

The actual trajectory is shown in blue, the prediction without time delay embedding in orange, and the prediction with time delay embedding in red. This approach results in an improvement in the forecast horizon from 31 steps to 297 steps. The hyperparameter combination that yielded the best performance for the RC consisted of 45.000 training steps, 1.000 reservoir nodes, a regularization parameter of $\beta = 10^{-3}$, a spectral radius of $\rho = 0.1$, an embedding dimension of $n = 48$, and a delay of $\tau = 6$.

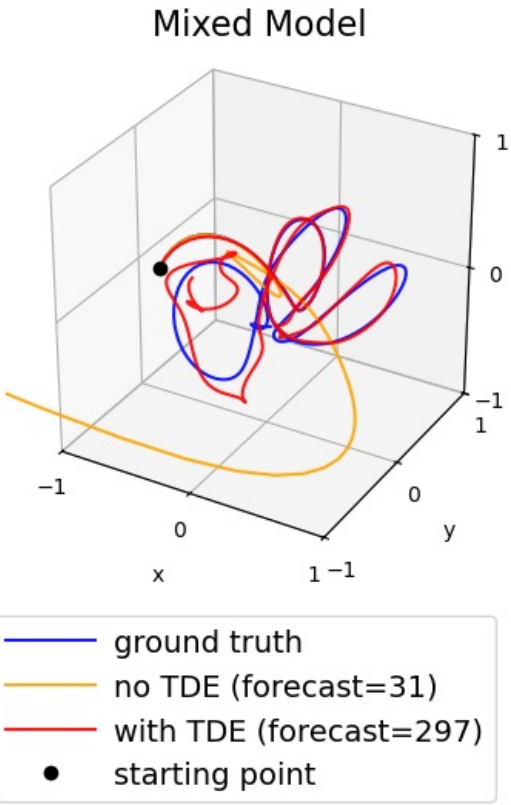


Figure 5.24: Prediction of mixed system with and without time delay embedding

5.7 Replicating Terms in System Equations by applying Mathematical Operations to the Reservoir State

One system that involves more complex terms than those discussed so far is the Ikeda map [55]. The Ikeda map is a two-dimensional map defined by:

$$\phi_n = 0.4 - 6/(1 + x_n^2 + y_n^2) \tag{5.21}$$

$$x_{n+1} = 1 + \nu(x_n cos(\phi_n) - y_n sin(\phi_n)) \tag{5.22}$$

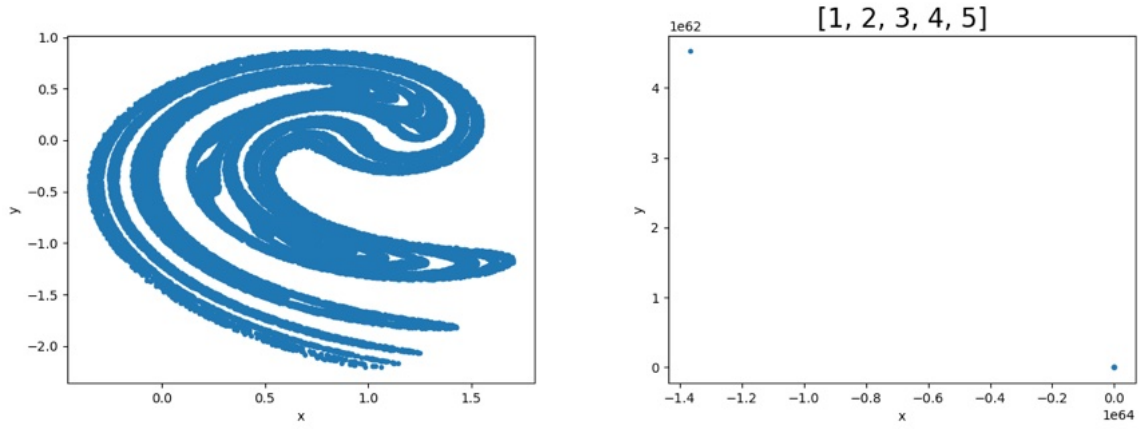
$$y_{n+1} = \nu(x_n sin(\phi_n) + y_n cos(\phi_n)) \tag{5.23}$$

With $\nu = 0.9$. We also tested all possible combinations of the elements of $X = [50, 60, \ldots, 140, 150]$ as numerators in the generalized reservoir state, as well as all combinations from a larger set with up to three elements. However, none of these approaches resulted in good predictions.

It is also known that a machine learning framework called SINDy [56], which is designed to recover governing system equations directly from data, is unable to recover the equations of the Ikeda map. One prerequisite for SINDy to be effective is that the underlying equations must contain only a few terms that define the dynamics, resulting in governing equations that are sparse within a high-dimensional nonlinear function space. Since the sparsity condition is not fully satisfied by the Ikeda map, SINDy fails to recover its equations. [57]

As with the mixed system, no good predictions could be achieved for the Ikeda map when using elements of a given powerset as possible nonlinearities in the generalized reservoir state.

Inspired by Karmogolov Arnold Networks (KANs) [58]—a machine learning architecture for predicting complex systems that utilizes basic functions and is capable of predicting the Ikeda map—we decided to incorporate similar functions into the generalized reservoir state. Previously, only the reservoir states and their higher powers were used to construct the generalized reservoir state. With knowledge of the underlying equations of the Ikeda map, we defined new functions to generate a generalized reservoir state specifically tailored to the Ikeda map. These functions take the regular reservoir state as input, and their outputs are concatenated with the reservoir state, enabling greater complexity than simply employing higher powers.



(a) 50.000 datapoints of Ikeda map

(b) Prediction with [1,2,3,4,5] as nonlinearities

Figure 5.25: Ikeda map

Functions such as $\sin(\vec{r}_i)$ and $\cos(\vec{r}_i)$ were used, as they naturally occur in the Ikeda map, where i denotes the ith element of $\vec{r}$. Additionally, the following functions were defined, which closely resemble the equations of the Ikeda map:

$$x(\vec{r}_i) = 0.4 - 6/(1 + \vec{r}_i^2) \tag{5.24}$$

$$sinx(\vec{r}_i) = sin(0.4 - 6/(1 + \vec{r}_i^2)) \tag{5.25}$$

$$cosx(\vec{r}_i) = cos(0.4 - 6/(1 + \vec{r}_i^2)) \tag{5.26}$$

Figure 5.25 (a) displays the raw data of the Ikeda map, while Figure 5.25 (b) presents the prediction produced by the Minimal-RC using nonlinearities the [1, 2, 3, 4, 5] and a spectral radius of $\rho = 10^{-6}$. As can be seen, with this choice of generalized reservoir state, accurate predictions cannot be achieved. This finding supports the earlier claim that all terms involved in the system equations must be present in the Minimal-RC for successful prediction.

Figure 5.26 shows the attractor for various combinations of the functions defined above, used as nonlinearities in the generalized reservoir state, for a spectral radius of $\rho = 10^{-6}$.

From a purely visual perspective, it is evident that the more information about the system is included in the Minimal-RC, the better the attractor can be reconstructed and predicted. It is important to note that, even with the added functions, the reconstructed attractors still differ from the actual one. Nevertheless, the trend of improved predictions with the inclusion of more relevant information in the Minimal-RC is clear.

Figure 5.27 presents the average correlation dimension over five realizations for several spectral radii, considering different combinations of the functions defined above. These combinations progressively incorporate more information relevant for predicting the system's equations.

The correlation dimension was selected as the primary measure in this analysis, as the forecast horizon did not yield meaningful results for the two-dimensional map in this case due to its consistently poor values. The focus here is on the quality of attractor reconstruction.

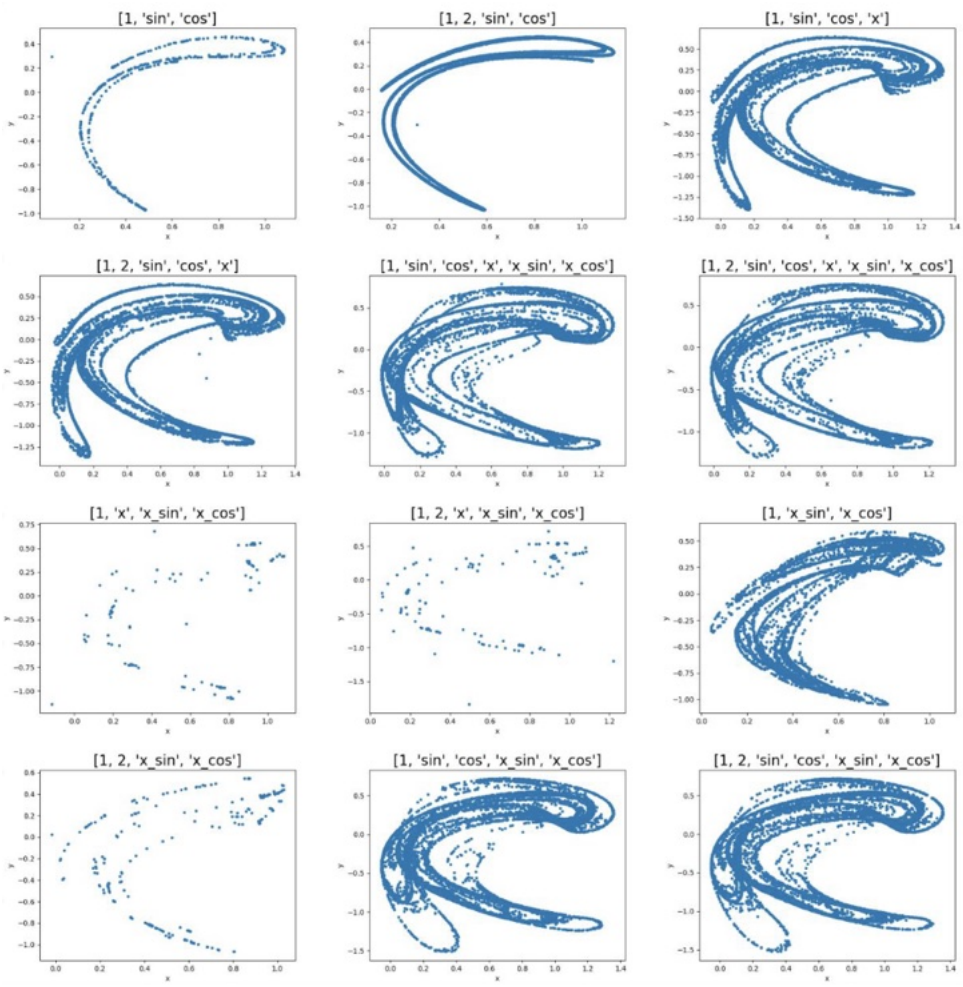


Figure 5.26: Qualities for all combinations visualized - Option 4

As shown, the predictions are not yet highly accurate. However, there is a clear trend: predictive quality improves as more relevant information—incorporated via functions in the generalized reservoir state—is included. For the generalized reservoir state [1, 2, 3, 4, 5], which is not closely related to the system equations, there is almost no attractor reconstruction. In contrast, the best performing combination ([1, sin, cos, x], across several spectral radii) achieves correlation dimensions of about 0.12. Although this is still not very accurate compared to the true correlation dimension of approximately 0.26, the trend is evident. These functions closely resemble the equations of the Ikeda map.

It is also notable that SINDy failed to reconstruct the system equations and to make any predictions for the Ikeda map. A library consisting of [1, $\sin(x)$, $\cos(x)$, x] was used, as well as a library involving specifically defined functions. Thus, the results here represent a step in the right direction.

Another example of a system that involves nonlinearities beyond polynomials is the Thomas system [59]:

$$x = -bx + sin(y) \tag{5.27}$$

$$x = -by + sin(z) \tag{5.28}$$

$$x = -bz + sin(x) \tag{5.29}$$

The parameter b is set to 0.18. Figure 5.28 displays the attractor of the Thomas system plotted in phase space.

In this case, a value of $dt = 0.3$ was used, as this time step is more suitable for the Thomas system [60]. The equations resemble those of the Halvorsen system in that they also exhibit an underlying cyclic

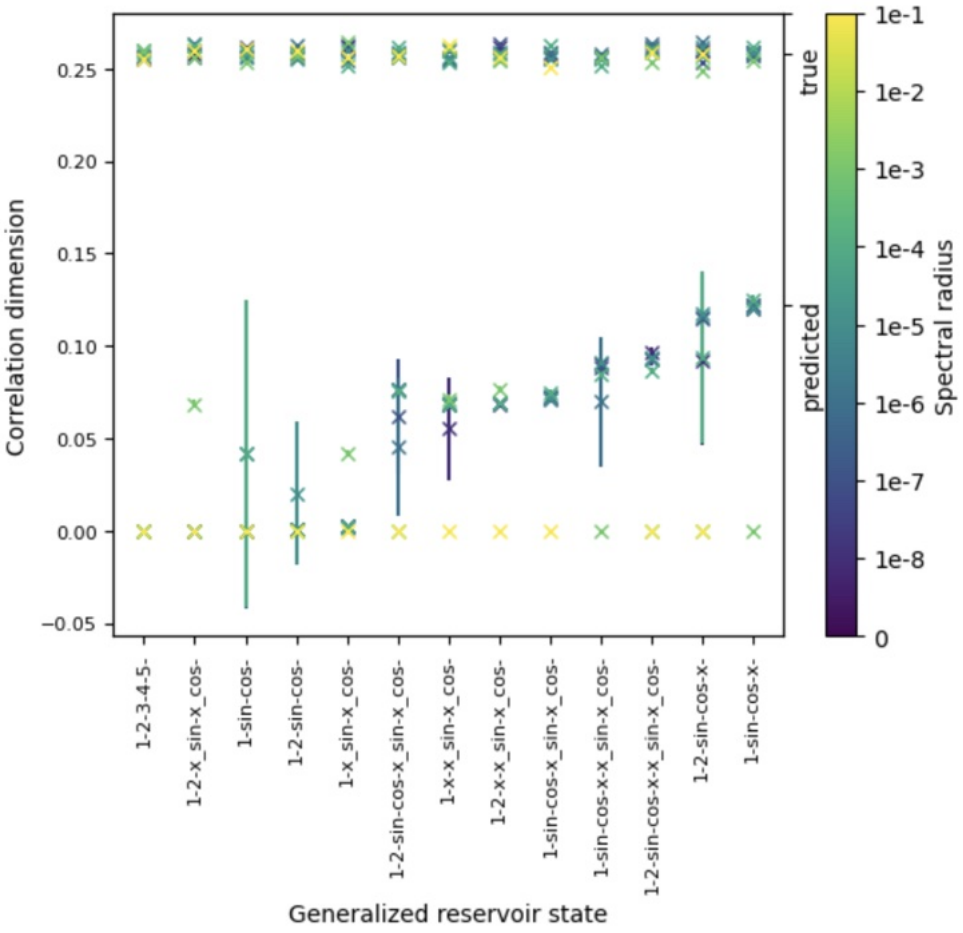


Figure 5.27: Correlation dimension for all combinations for several spectral radii: Prediction vs. true

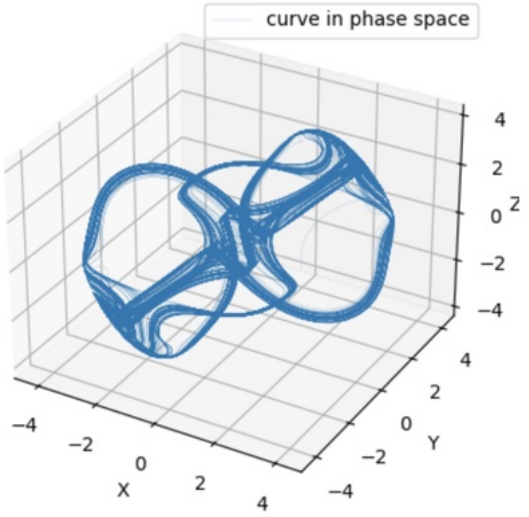


Figure 5.28: Thomas attractor for 50.000 steps of dt=0.3

symmetry. A similar experiment to the one performed for the Ikeda map was conducted. A function was defined that, in theory, is particularly well suited for the Thomas equations:

$$cyc(\vec{r}_i) \;=\; \vec{r}_i + sin(\vec{r}_{((i+2)mod9)+1}) \tag{5.30}$$

where $i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. This function exploits the cyclic symmetry and the fact that, aside from the $\sin(*)$ terms, only linear terms are present in the equations. The function is applied exclusively to the portion of the reservoir state that contains information from the input nonlinearity of one, regardless of the order of input nonlinearity. Therefore, i is restricted to values less than ten, since a block size of three is being used.

For example, consider $i = 1$. The first entry of the reservoir state corresponds to the x-coordinate of the input data. The *cyc* function adds the sin of the fourth entry of the reservoir state, which corresponds to the y-coordinate of the input. The scaling of the input coordinates by the w vector in the input matrix is taken into account by ensuring that the first component is added to the sin of the fourth component, the second to the sin of the fifth, and so on. This reasoning applies specifically for a block size of three. The function is designed to mimic the structure of the governing system equations and is applied only to the relevant components of the reservoir state. It is then concatenated with the linear part and all other nonlinearities in the generalized reservoir states as usual, but it always has a fixed size of nine, regardless of the order of nonlinearity in the input combinations. To clarify:

$$cyc(\vec{r}_1) = \vec{r}_1 + sin(\vec{r}_4) = \vec{r}_{x_1} + sin(\vec{r}_{y_1}) = cyc(\vec{r}_{x_1}) \tag{5.31}$$

$$cyc(\vec{r}_2) = \vec{r}_2 + sin(\vec{r}_5) = \vec{r}_{x_2} + sin(\vec{r}_{y_2}) = cyc(\vec{r}_{x_2}) \tag{5.32}$$

$$cyc(\vec{r}_3) = \vec{r}_3 + sin(\vec{r}_6) = \vec{r}_{x_3} + sin(\vec{r}_{y_3}) = cyc(\vec{r}_{x_3}) \tag{5.33}$$

$$cyc(\vec{r}_4) = \vec{r}_4 + sin(\vec{r}_7) = \vec{r}_{y_1} + sin(\vec{r}_{z_1}) = cyc(\vec{r}_{y_1}) \tag{5.34}$$

$$... \tag{5.35}$$

$$cyc(\vec{r}_9) = \vec{r}_9 + sin(\vec{r}_3) = \vec{r}_{z_3} + sin(\vec{r}_{x_3}) = cyc(\vec{r}_{z_3}) \tag{5.36}$$

The subscript j in x_j, y_j, z_j refers to the jth component within the corresponding block. With a block size of $b = 3$, each input combination occupies three entries in the reservoir state. Figure 5.29 presents the forecast horizon for different generalized reservoir states across various spectral radii and orders of input nonlinearity. The orders of input nonlinearity are not distinguished in the plot, as there is no significant difference in forecast horizon for different values. This is expected, since the equations do not contain mixed terms such as xy.

Several interesting findings emerge. There is no significant difference between input nonlinearity orders one, two, and three, which is to be expected. The reason for this will be discussed in more detail in the next section. As a result, no distinction is made between the different orders of input nonlinearity. Good predictions are also observed for generalized reservoir states that include odd powers, while poor predictions occur for states composed solely of even powers. This may be explained by the fact that the Taylor expansion of $sin(*)$ contains only odd terms, thereby implicitly incorporating relevant information. Additionally, good predictions arise for generalized reservoir states that include $sin(*)$ and $cyc(*)$ functions, especially when combined with other terms.

For example, comparing $[1, 3, 5, 7, 9]$, $[1, 3]$, and $[1, 3, cyc]$ shows that using only linear and cubic reservoir states is not sufficient. Achieving good results requires either higher powers or the inclusion of intelligently designed functions. The predictions are arguably slightly better when using $[1, 3, 5, 7, 9]$ as powers compared to $[1, 3, cyc]$ or $[1, 3, sin]$. However, the latter two options involve fewer computational steps, and it is notable that the Minimal-RC architecture can be modified in specific ways to better suit particular equations and trends in the time series data.

It is evident that predictions improve as more knowledge of the system equations is incorporated into the generalized reservoir state. This finding further supports the claim that achieving a match between the governing equations and the generalized reservoir state, which needs to implement the relevant information in one or the other form, leads to better predictive performance. This presents a Catch-22 situation: paradoxically, Minimal-RC can make better predictions once the governing equations are known. However, if the equations are already known, the use of machine learning becomes unnecessary.

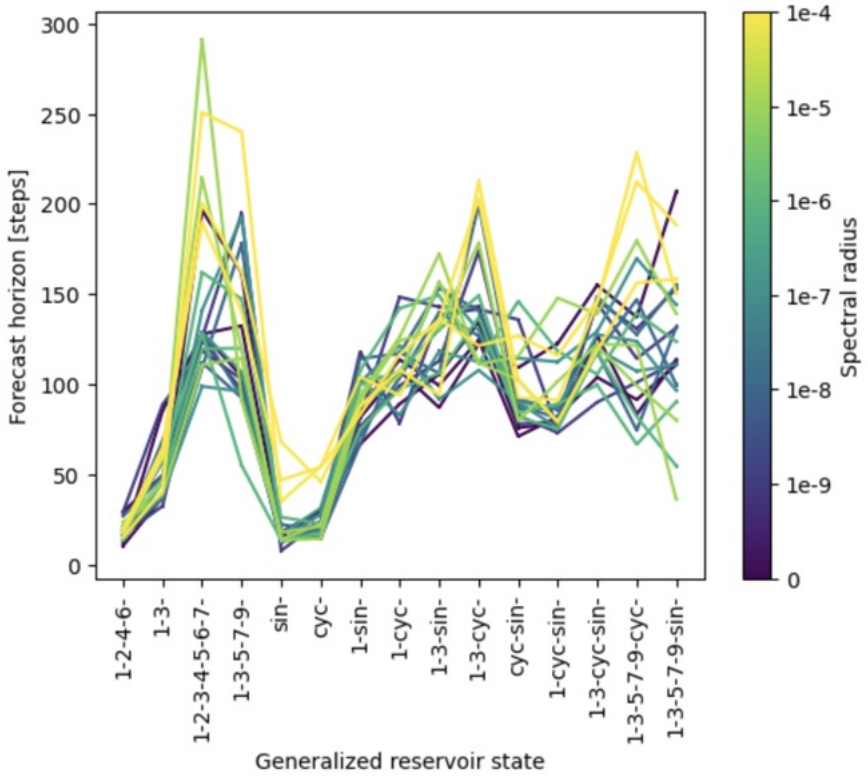


Figure 5.29: FH for different generalized reservoir states for several spectral radii and all orders of input nonlinearity

5.8 Replicating Terms in System Equations by Fine-Tuning the Input Matrices

As the correlation between the governing system equations and the generalized reservoir state becomes increasingly apparent, a more detailed analysis is warranted. The underlying assumption is that the relevant information from the system data must be present in the Minimal-RC in some form.
For example, consider the input combination $x + y$. With a nonlinearity order of $\eta = 2$, this would include information from $(x + y)^2 = x^2 + 2xy + y^2$, meaning that the xy term is embedded within the squared part of the generalized reservoir state when using $x + y$ as an input combination. It is proposed that including only the relevant terms from the equations is sufficient. Thus, if the system equations contain only an xy term, it would suffice to use the combined input xy rather than $x + y$, which introduces additional, potentially irrelevant information. Of course, this would require prior knowledge of the system equations, which is typically unavailable in real-world scenarios. Nevertheless, experiments were conducted to demonstrate that this assumption holds.

Implementing input combinations such as xy is not straightforward within the existing Minimal-RC architecture, which is primarily designed to input sums of coordinates like $x + y$. To address this, a slight modification was introduced. The process of creating Lorenz-specific input combinations mirrors the approach used for Halvorsen-specific input, which are the two systems that are used in the following. This will be demonstrated using the Lorenz system as an example. The input operation was structured as follows:

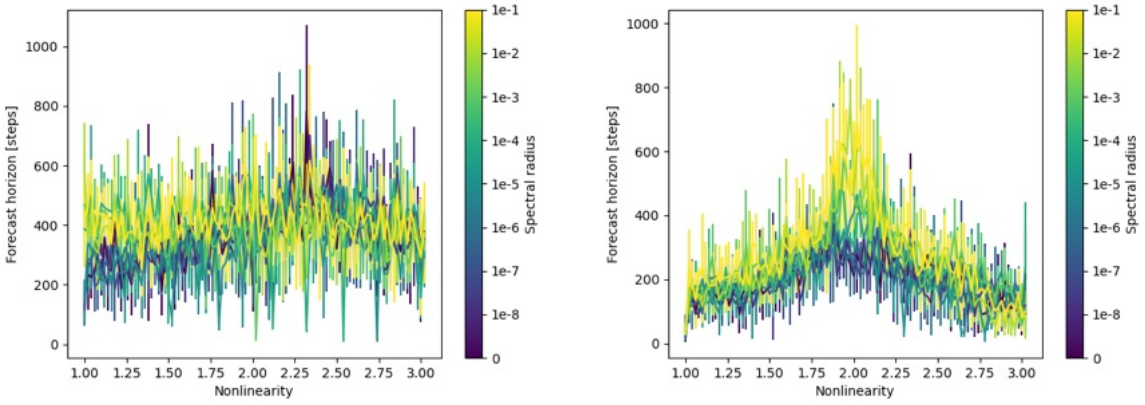
$$\boldsymbol{W}_{in}\vec{u}(t) = \begin{pmatrix} w & 0 & 0 \\ 0 & w & 0 \\ 0 & 0 & w \\ w & 0 & 0 \\ w & 0 & 0 \\ 0 & w & 0 \end{pmatrix} \vec{u}(t) = \begin{pmatrix} x \\ y \\ z \\ x \\ x \\ y \end{pmatrix} (t) \otimes w$$

The last three input combinations were then multiplied by the values of the coordinates y, z, and z during the training, prediction, and synchronization loops, prior to processing the input through the reservoir. These modified values were further multiplied by the corresponding entries of the w-vector before multiplying with the operation $\boldsymbol{W}_{in}\vec{u}(t)$. After, the term $\boldsymbol{A}\vec{r}(t)$ was added and then the regular procedure was conducted. The procedure for obtaining the next reservoir state was therefore modified as follows:

$$\vec{r}(t+1) = \boldsymbol{A}\vec{r}(t) + \boldsymbol{W}_{in}\vec{u}(t) * \vec{m}(t) \tag{5.37}$$

$$\vec{m}(t) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ y \\ z \\ z \end{pmatrix} (t) \otimes w \tag{5.38}$$

First, the input combinations were varied to tailor them specifically for the Lorenz system. The Lorenz system involves the linear terms x, y, and z, as well as the nonlinear terms xy and xz, in combination with constant terms. Therefore, the input combinations x, y, z, xy, xz, and yz were used. The "linear reservoir state"—which, due to these combinations, is not purely linear—was then concatenated with higher powers of itself as usual. If the assumption is correct, the Lorenz system should be predictable using only this "linear" state, and indeed, this was observed in practice (Figure 5.30(a)).



(a) Lorenz system

(b) Halvorsen system

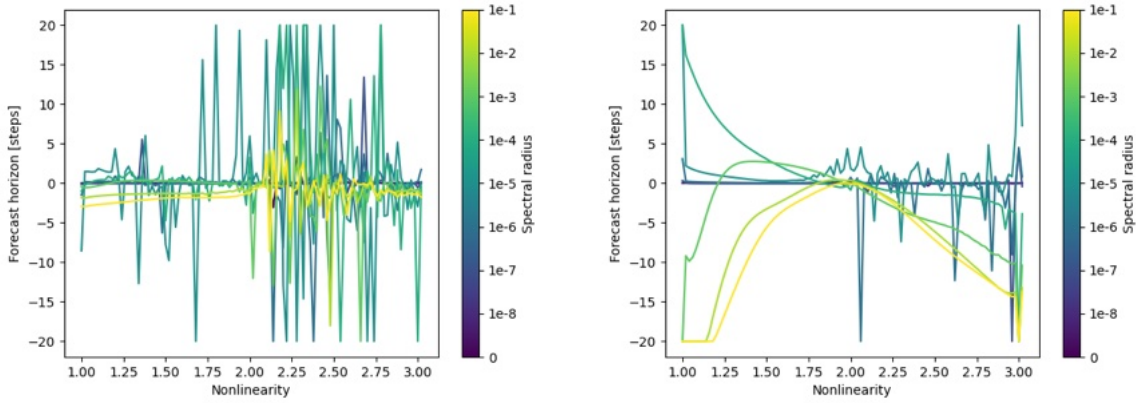
Figure 5.30: Forecast horizons plotted against nonlinearities

The Halvorsen system includes terms such as x^2, y^2, and z^2. If the assumption holds, it would be necessary for the quadratic reservoir state to be part of the generalized reservoir state in order to achieve good predictions, which is indeed confirmed by the results. Figure 5.30 displays the forecast horizons for both the Lorenz system and the Halvorsen system across several spectral radii, with the nonlinearity varied from one to three. Here, varying the nonlinearity means that the linear reservoir state is always included in the generalized reservoir state, which is then concatenated with exactly one higher power.

For the Lorenz system, good predictions are observed across all nonlinearities. This occurs because the "linear" reservoir state is included in every case, already providing all the necessary information. For the Halvorsen system, prediction quality peaks at quadratic nonlinearity, which is when all essential information is present for accurate prediction making.

Figure 5.31 displays the average output matrix weights for different nonlinearities for both the Lorenz system and the Halvorsen system, which shows some interesting trends in the matrix weights.

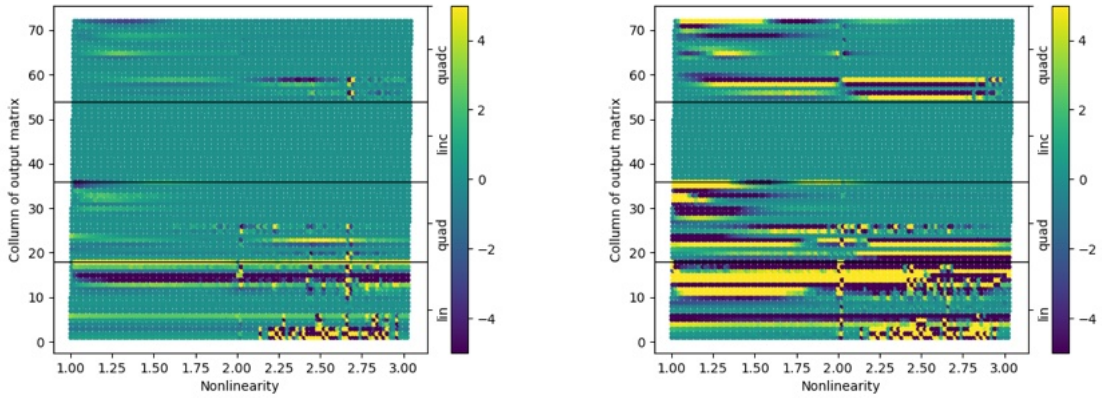
For the Halvorsen system, a reduction in the average W_{out} weights is observed as the quadratic nonlinearity is approached across most spectral radii. This aligns with earlier observations in this thesis: good



(a) Lorenz system

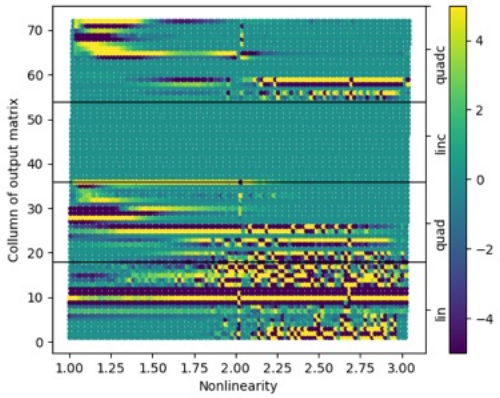
(b) Halvorsen system

Figure 5.31: Average output matrix weights plotted against nonlinearities



(a) Maps to x coordinate

(b) Maps to y coordinate



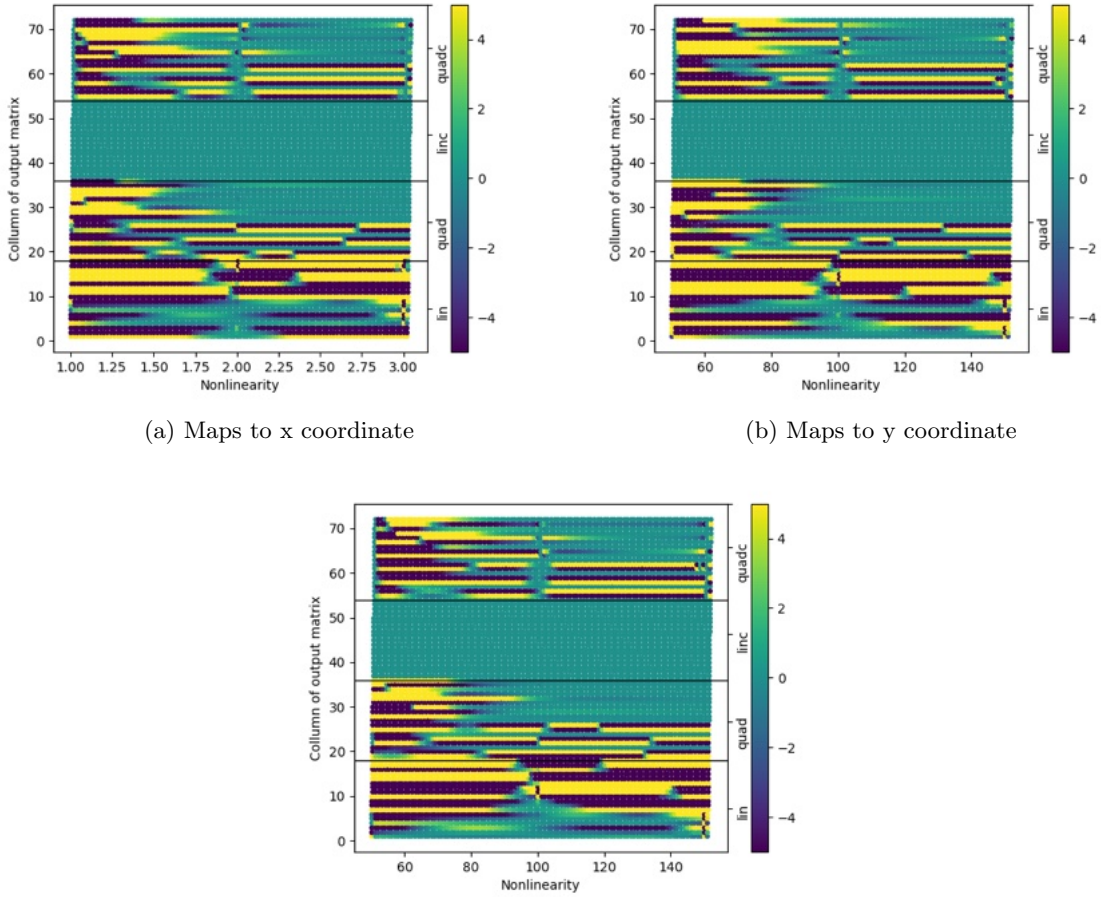
(c) Maps to z coordinate

Figure 5.32: Activated matrix collumns for Lorenz system

predictions result in stabilized matrix weights. In contrast, the average W_{out} weights for the Lorenz system remain relatively constant for nonlinearities less than two, but exhibit greater fluctuations for nonlinearities larger than two. This likely indicates that the Minimal-RC is attempting to learn from irrelevant information when higher powers are used, which is unnecessary in this context and causes the matrix weights to diverge.

The instability in the matrix weights probably arises from sections of the output matrix that do not contribute meaningfully to the predictions. When the "linear" reservoir state is concatenated with a higher power close to one, less irrelevant information is incorporated, leading to more stable learning compared to using a much higher power. Figure 5.32 illustrates the activation of the matrix columns for the Lorenz system, showing the X-, Y-, and Z-parts of the matrices. The secondary y-axis on the right side of the plot indicates the linear part (lin), the quadratic part (quad), the linear complex part (linc), and the quadratic complex part (quadc) of the generalized reservoir state.

For this study, the complex extension has been employed. As expected, there is no activation in the linear and complex parts of the generalized reservoir state. Similarly, there is no complex activation for exact quadratic nonlinearity, which is also expected. The real linear part demonstrates activation in certain columns consistently across all nonlinearities, suggesting that these columns are relevant for making predictions. Other columns exhibit activation as well, but this activity is not stable across all nonlinearities and is likely irrelevant for prediction. Figure 5.33 illustrates the activation of the matrix columns for the Halvorsen system, showing the X-, Y-, and Z-parts of the matrices.



(a) Maps to x coordinate

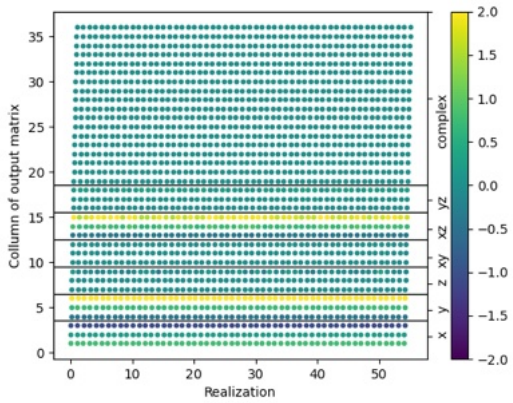
(b) Maps to y coordinate

(c) Maps to z coordinate

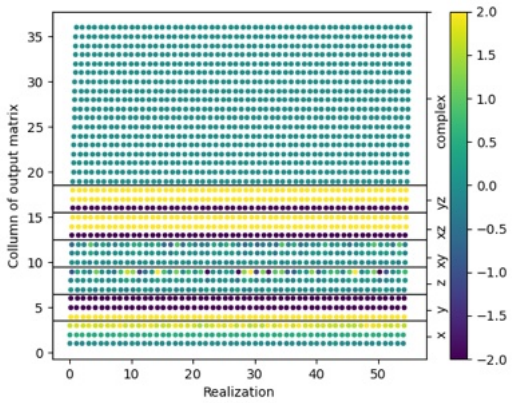
Figure 5.33: Activated matrix collumns for Halvorsen system

In contrast to the Lorenz system, there are no columns that are consistently activated across all nonlinearities for the Halvorsen system, however, there is a clear change for the quadratic nonlinearity. It is also notable that when the match between the system equations and the generalized reservoir state is imperfect, the complex part of the output matrices appears to compensate for this mismatch through increased activation.

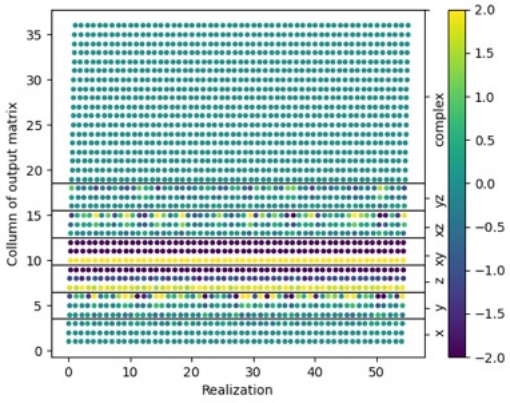
To further investigate, the matrices for both systems were analyzed using only the terms present in



(a) Maps to x coordinate



(b) Maps to y coordinate



(c) Maps to z coordinate

Figure 5.34: Activated matrix collumns for Lorenz system

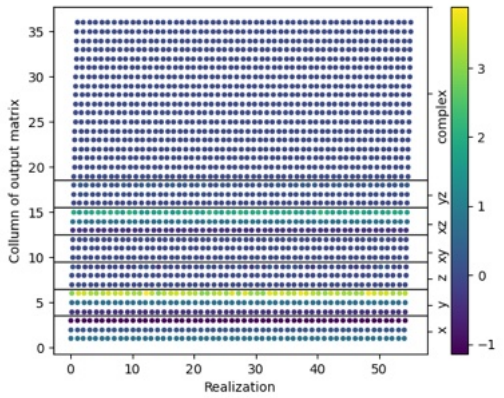
the equations, which naturally leads to good predictions. The aim is to identify parallels with the system equations. Figure 5.34 displays the matrix columns for the Lorenz system when a purely "linear" reservoir state is used.

Once again, there is no complex activation. The equation for the x-coordinate contains linear terms in x and y, and the corresponding matrix columns display clear activation. The xz column is also activated. For the y-coordinate, the equation includes linear terms in x and y, as well as one quadratic term xz; all of these columns are activated. The yz column also demonstrates activation.

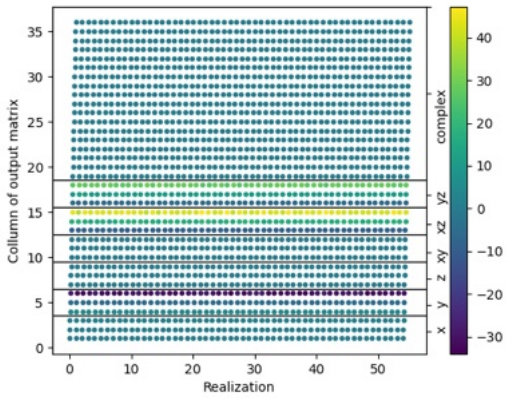
Some columns that are not directly part of the system equations are activated as well. It is possible that they contribute to the prediction in a way that is not yet fully understood. Regarding the z-coordinate, the terms z and xy are clearly and consistently activated across all realizations. While some fluctuations are present, constant activation of the relevant columns can be observed in every realization.

The activation of the xz column in the x-part of the output matrix and the yz column in the y-part are not immediately obvious, but they may indicate a more complex interaction between the rows of the output matrix. It should also be noted that, for better visualization, the values have been capped at upper and lower limits of two and minus two before plotting the activations. Figure 5.35 presents the same plot without the values being capped.

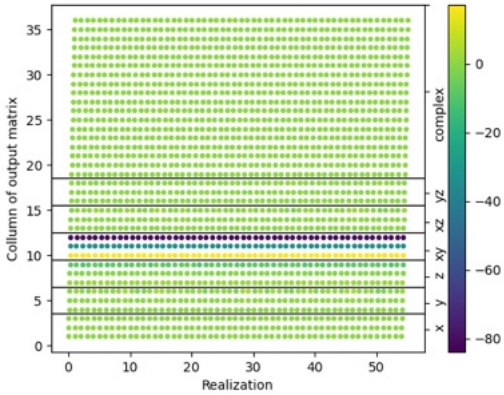
The input combinations were also varied to specifically suit the Halvorsen system, which involves the terms x, y, z, x^2, y^2, and z^2. Accordingly, these input combinations were defined as the "regular"



(a) Maps to x coordinate



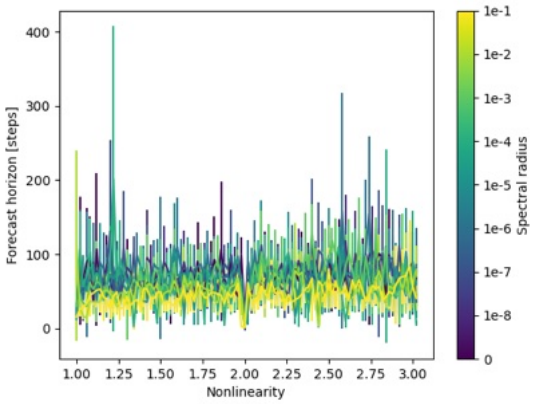
(b) Maps to y coordinate



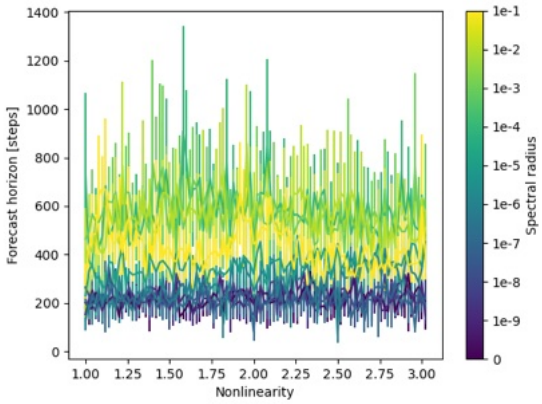
(c) Maps to z coordinate

Figure 5.35: Activated matrix collumns for Lorenz system without capped values

reservoir state for this case. Regardless of the nonlinearities used in the generalized reservoir state, cross terms such as xy are never included, making the Lorenz system unpredictable with this configuration. Figure 5.36 presents the forecast horizons for the "regular" reservoir state (as newly defined here) combined with a higher power, for both the Lorenz system and the Halvorsen system.

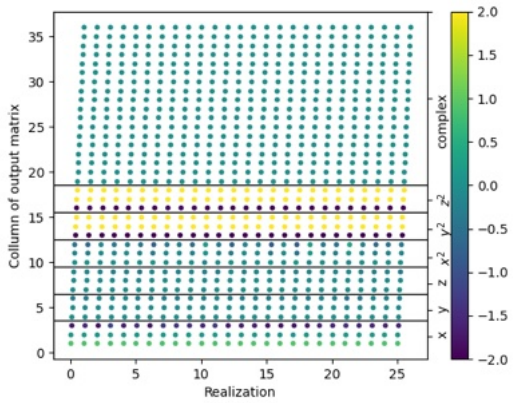


(a) Lorenz system

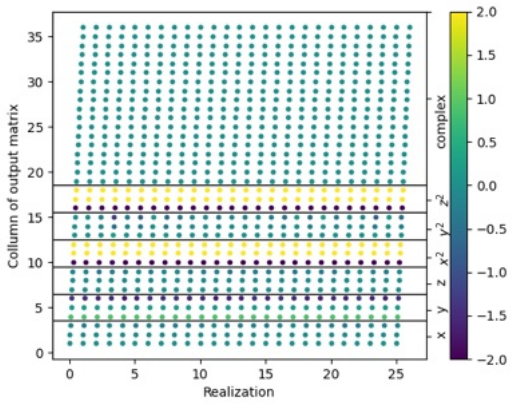


(b) Halvorsen system

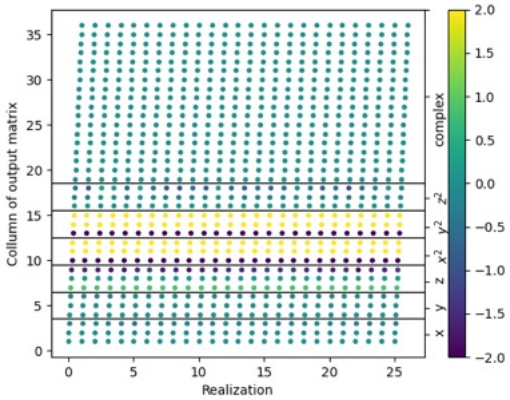
Figure 5.36: Forecast horizons for varied nonlinearities



(a) Maps to x coordinate



(b) Maps to y coordinate



(c) Maps to z coordinate

Figure 5.37: Activated matrix collumns for Halvorsen system

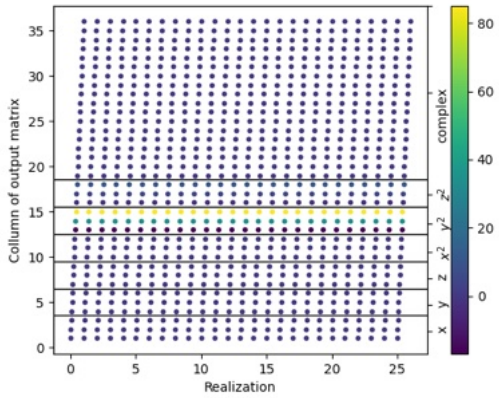
As expected, prediction quality is high for all higher powers in the Halvorsen system and poor for all higher powers in the Lorenz system. A similar pattern emerges as before: the matrix weights remain mostly stable across all nonlinearities for the Halvorsen system, while they are unstable for the Lorenz system. Consistently activated columns (in the "linear" part) are also observed in the Halvorsen system. Figure 5.37 displays the activations of the output matrices for the Halvorsen system, using only the "regular" reservoir state, which already contains all the necessary information for accurate prediction.

The equation for the x-coordinate involves the terms x, y, z, and y^2. Activations are observed in the x and y^2 columns, as well as in the z^2 column. A similar pattern is seen for the y- and z-coordinates. The equation for the y-part consists of x, y, z, and z^2 terms, and shows activation in the y and z^2 columns, as well as in x^2. The z-equation includes x, y, z, and x^2, with its corresponding matrix row activated in the z and x^2 columns, as well as in y^2.

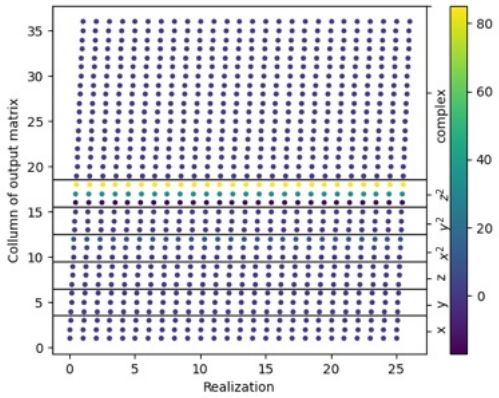
The Halvorsen system is cyclically symmetric, and this symmetry is reflected in the cyclic activation of the matrix columns. The quadratic term in each equation is always activated in the corresponding matrix row. The linear term with the prefactor a is also always activated in the matrix row corresponding to the coordinate in the equations. Interestingly, the other two linear terms are inactive in every case, even though they also appear in the data. Additionally, the quadratic term for the coordinate in question is always inactive (for example, x^2 is inactive in the x-part of the output matrix), while the remaining quadratic coordinate is activated.

Not every column is activated as would be expected from the structure of the equations, suggesting

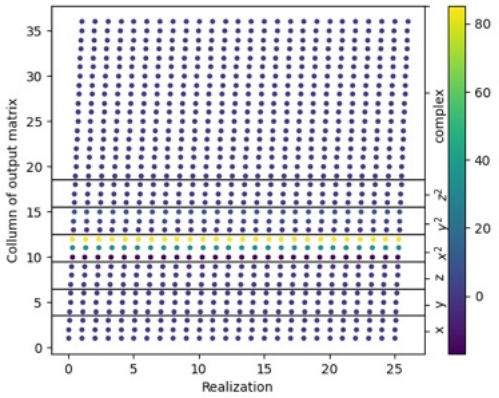
the presence of more complex interactions between matrix rows, possibly involving cancellation effects. Nevertheless, a clear parallel can be observed between the cyclic symmetry in the equations and the activation patterns of the matrix columns. Figure 5.38 presents the same plot without the values being capped.



(a) Maps to x coordinate



(b) Maps to y coordinate



(c) Maps to z coordinate

Figure 5.38: Activated matrix collumns for Halvorsen system without capped values

Interestingly, the quadratic terms appear to dominate the overall activation of the matrix weights. Additional plots illustrating the behavior of the matrix weights when varying the generalized reservoir state are provided in the appendix section [B.4].

An interesting observation in both examples is the consistent structure among the three indices associated with the same input combination ($b = 3$). The value corresponding to the first index is either positive or negative, while the values of the second and third indices have the opposite sign. The last index of each input combination corresponds to the component that contains information solely from past data points, since the new data point is multiplied by zero in the input matrix. In contrast, the first and second indices also incorporate information from the current data point. Through the multiplication of the generalized reservoir state with the output matrix, the Minimal-RC appears to blend memory of past data with current data by utilizing different signs. This process enables the filtering of relevant information from both the previous and current data points to make predictions.

Consider one more example: the Arneodo system [61], which is defined by the following equations:

$$\dot{x} = y \tag{5.39}$$

$$\dot{y} = z \tag{5.40}$$

$$\dot{z} \;=\; -ax - by - z + cx^3 \tag{5.41}$$

With $a = -5.5$, $b = 3.5$, and $c = -1$, the Arneodo attractor appears as shown in Figure 5.39.

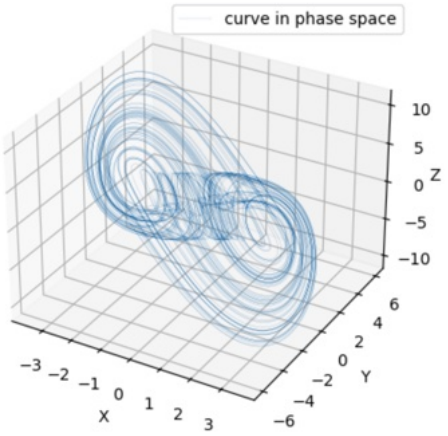


Figure 5.39: Arneodo attractor plotted for 50.000 steps of dt=0.01

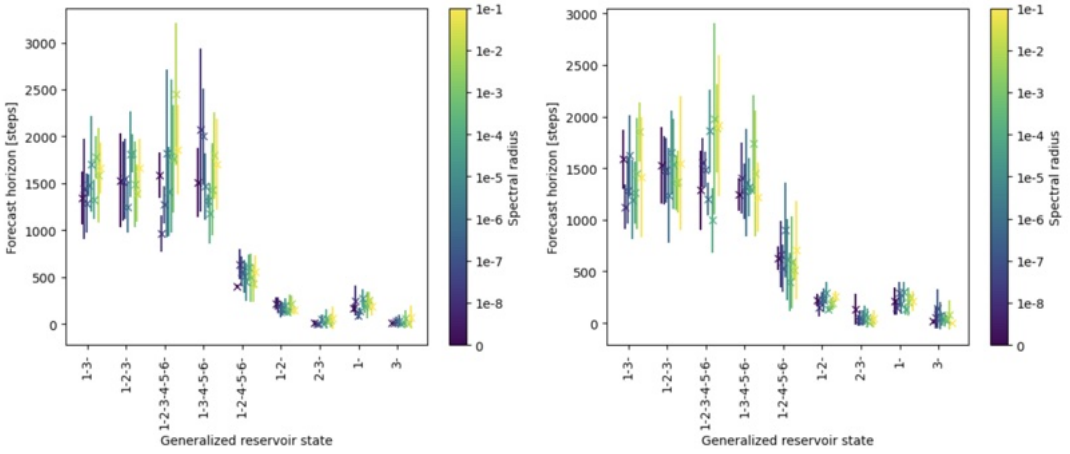
It is clear that the governing equations of the Arneodo system involve only linear terms (y, z, $-ab$, $-by$, $-z$) and a single cubic term (cx^3). There are no quadratic terms present in the data. Therefore, it should be sufficient to use only the linear reservoir state and the cubic reservoir state during training. It should also suffice to include input combinations up to linear order. Input combinations such as $x + y$, which would result in quadratic terms like $(x + y)^2 = x^2 + 2xy + y^2$ in the quadratic reservoir state, are expected to be unnecessary, as there are no cross terms like xy in the equations.
Figure 5.40 displays the forecast horizon for several generalized reservoir states and spectral radii, considering all three possible nonlinearities in the input for the standard Minimal-RC architecture (no complex extension was used in this case).

As expected, the forecast horizon is highest for generalized reservoir states that include both the linear and cubic reservoir states. Including additional components beyond the linear and cubic parts does not significantly improve predictive quality, since these additional parts do not contain relevant information for the system. Similarly, using input nonlinearities of order higher than one does not yield substantial improvements, as anticipated. These observations hold consistently across all spectral radii used in the simulations.

Figure 5.41 presents these findings in an alternative plot, where spectral radii are not distinguished. This further emphasizes the importance of selecting the appropriate generalized reservoir state. In this plot, the order of nonlinearity in the input is color-coded, making it clear that this parameter does not have significant impact in this particular example. All necessary information is already present in the input nonlinearity of one, which is automatically included in orders two and three as well. The same simulations were performed using the complex version of the Minimal RC architecture with input combinations x, y, z, xy, xz, yz, yielding similar results. This variation of Minimal-RC also includes the relevant information for predicting the Arneodo system for a linear and cubic order in generalized reservoir state. These results are provided in the appendix [B.5].

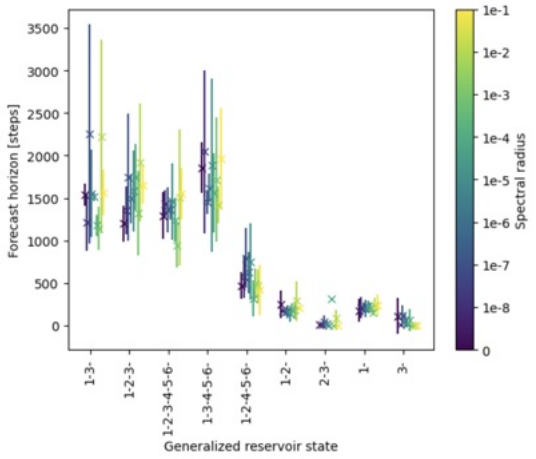
The selection of input combinations, as well as the choice of nonlinearities in the generalized reservoir state, is therefore of considerable importance. For relatively simple systems such as the Halvorsen or Lorenz systems, which involve only simple polynomials, it is generally sufficient to use x, y, z, xy, xz, and yz as input combinations. The powers in the generalized reservoir state can then reproduce higher-order nonlinearities present in the system data.

However, challenges arise when the system equations include more complex terms, such as xyz or x^2yz^3. This type of information would not be captured without appropriate input combinations. Input combinations like $x + y$ or $x + y + z$ become essential in these cases, as they can represent such terms when



(a) Input nonlinearity of 1

(b) Input nonlinearity of 2



(c) Input nonlinearity of 3

Figure 5.40: FH of different generalized reservoir states and spectral radii for all possible input nonlinearities for the Arneodo system

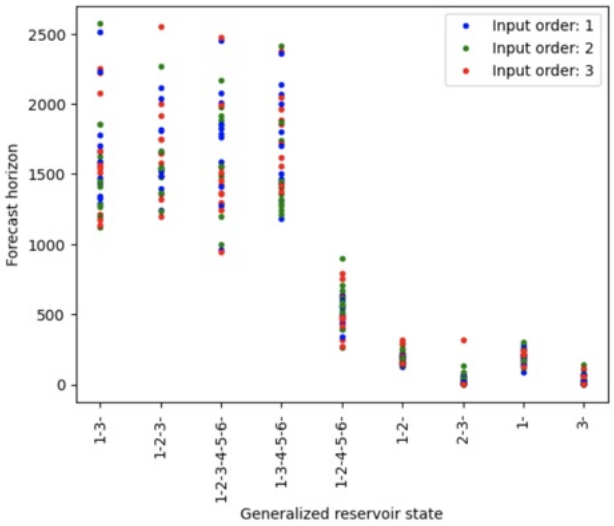


Figure 5.41: FH of different generalized reservoir states for all possible input nonlinearities for the Arneodo system

higher powers are used in the generalized reservoir state. At the same time, it is assumed that including only the relevant information is optimal. Feeding more irrelevant information into the Minimal-RC increases the likelihood of the machine learning unnecessary or distracting patterns, which can overwhelm the model, particularly for more complex systems.

5.9 Spectral Radius

As discussed in the theory section, the target spectral radius determines the extent to which information from past data points is retained (Chapter 5.4). A target spectral radius of zero implies that no information from the past is incorporated. Unlike regular reservoir computing, the maximum allowable value for the spectral radius in Minimal-RC is one, since no activation function is used to keep the values bounded.
Figure 5.42 presents a plot of the forecast horizon as a function of the spectral radius, varied from zero to one, for the complex Minimal-RC architecture applied to the Halvorsen system.

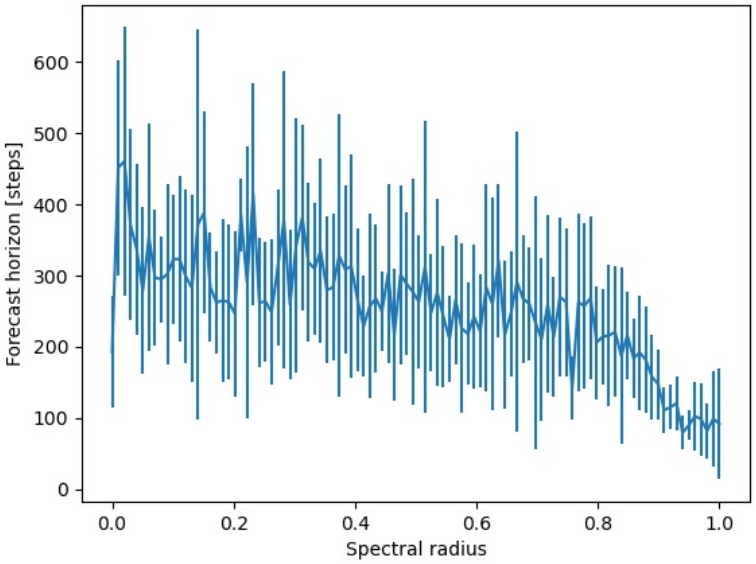
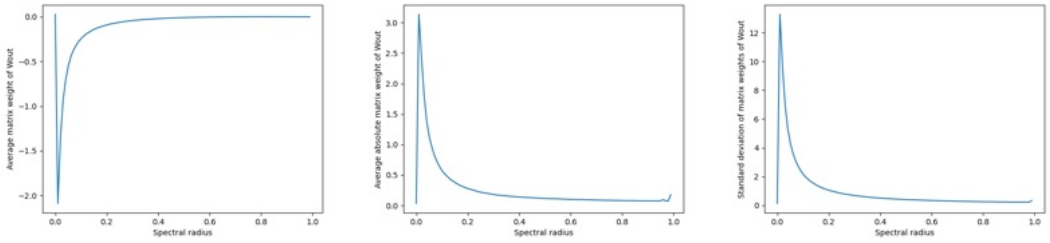


Figure 5.42: FH vs. Spectral radius for Halvorsen system for 20 realizations

The minimal set of necessary input information (x, y, z, x^2, y^2, z^2) with a power of one was used in this analysis. A similar experiment was conducted using the standard complex Minimal-RC framework, with an input nonlinearity order of one and a power of two in the generalized reservoir state—effectively providing the same amount of information. In the first scenario, the squared coordinates are incorporated directly during the training loop, while in the second scenario, they are concatenated after the loop for fitting the output matrix. Both approaches produced similar results. Detailed findings from the latter scenario are available in the appendix section [B.6].
The plot shows good forecast horizons, with quality decreasing as the spectral radius approaches one.

Figure 5.43 presents three different metrics calculated for the output matrix weights, plotted against the spectral radius.

Interesting behavior is observed in the matrix weights for spectral radius values between 0 and 0.2. When the spectral radius is 0, the matrix weights are very small. A sharp peak appears for small spectral radii, which then saturates as the spectral radius increases further. Interestingly, the prediction performance remains fairly consistent across different spectral radii. This phenomenon was also observed for the Lorenz system when using the architecture specifically tuned for it. Additional details can be found in the appendix section [B.6].



(a) Average weight (b) Average absolute weight (c) Average standard deviation

Figure 5.43: Average matrix weights vs. spectral radius for Halvorsen system

A smaller spectral radius results in less memory of past data points (recall that $A = \frac{\rho^*}{b}J$). In the extreme case of a spectral radius of zero, there is no knowledge of past data points at all. In this case, the evolution of the reservoir state is given by:

$$\vec{r}(t+1) \;=\; \boldsymbol{W}_{in}\vec{u}(t) \tag{5.42}$$

The small values observed in the output matrix weights can be explained by recalling that the generalized reservoir state contains values only from the most recent data point. This means that only a matrix multiplication with small parameters is needed, since the next data point differs only slightly from the previous one, since they are neighbouring points in a continuous flow.
The trend of having larger matrix weights (in terms of magnitude) for smaller spectral radii is likely due to the fact that, as the spectral radius increases, the values in the generalized reservoir state become larger. This is because larger values are added to the current data point, multiplied by the input matrix, which then requires smaller weights for the matrix multiplication to yield an accurate prediction:

$$\text{if: } \rho \uparrow \rightarrow \;\; \vec{r}(t) \uparrow \tag{5.43}$$

When the value of the generalized reservoir state is increased, the weights of the output matrix decrease, since large values in the reservoir state need small values in the output matrix to compensate back onto the next data point.
For this particular case, the amount of memory retained does not appear to have a significant impact on the prediction quality.

A measure, which should give more insights is the relative standard deviation. The relative standard deviation is defined as the regular standard deviation divided by the average of the absolute weights of the output matrix. Figure 5.44 presents the relative standard deviation plotted against the spectral radius.

It can be seen that the relative standard deviation decreases as the spectral radius increases, but does not approach zero except for values very close to one. This relative variability allows for predictions to be made, even when the regular standard deviation becomes very small. The fairly consistent values of the relative standard deviation across all spectral radii are consistent with the similar values observed for the forecast horizons.

Figure 5.36b presents the forecast horizons for different spectral radii and nonlinearities of the generalized reservoir state for the Minimal-RC architecture specifically tuned for the Halvorsen system. The first notable finding is that good predictions are achieved across all "nonlinearities," as the "linear" reservoir state already contains all the necessary information. Secondly, the spectral radius appears to have a significant influence on prediction quality in this case. Figure 5.45 provides a more detailed analysis.

The x-axis represents the $\log_{10}$ of the spectral radii used in Figure 5.36b. The y-axis displays the forecast horizon, the average absolute weight of the output matrix, and the relative standard deviation of the matrix weights. Only the "linear" input and a purely linear reservoir state were used to generate this plot. It is important to note that, although the forecast horizon improves for certain spectral radii, this does not imply that predictions for other spectral radii are poor. The forecast horizon begins to improve for spectral radii greater than approximately $\rho = 10^{-5}$. At this point, the average absolute

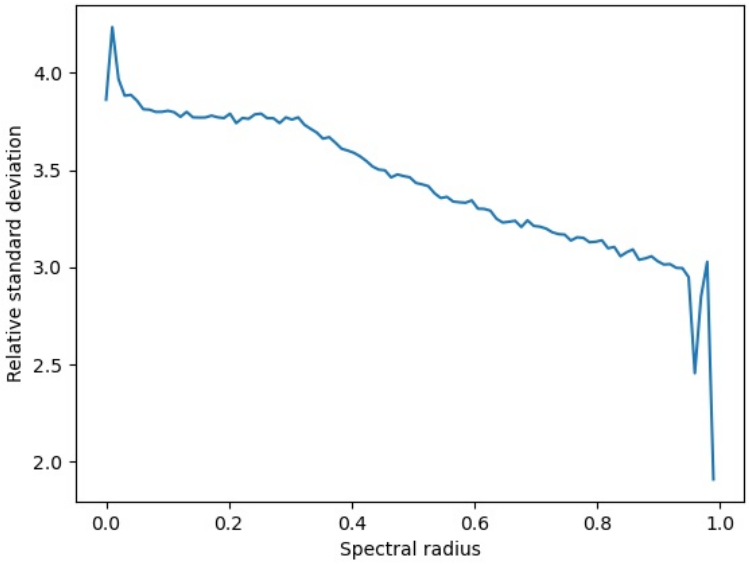


Figure 5.44: Relative standard deviation vs. spectral radius for Halvorsen system

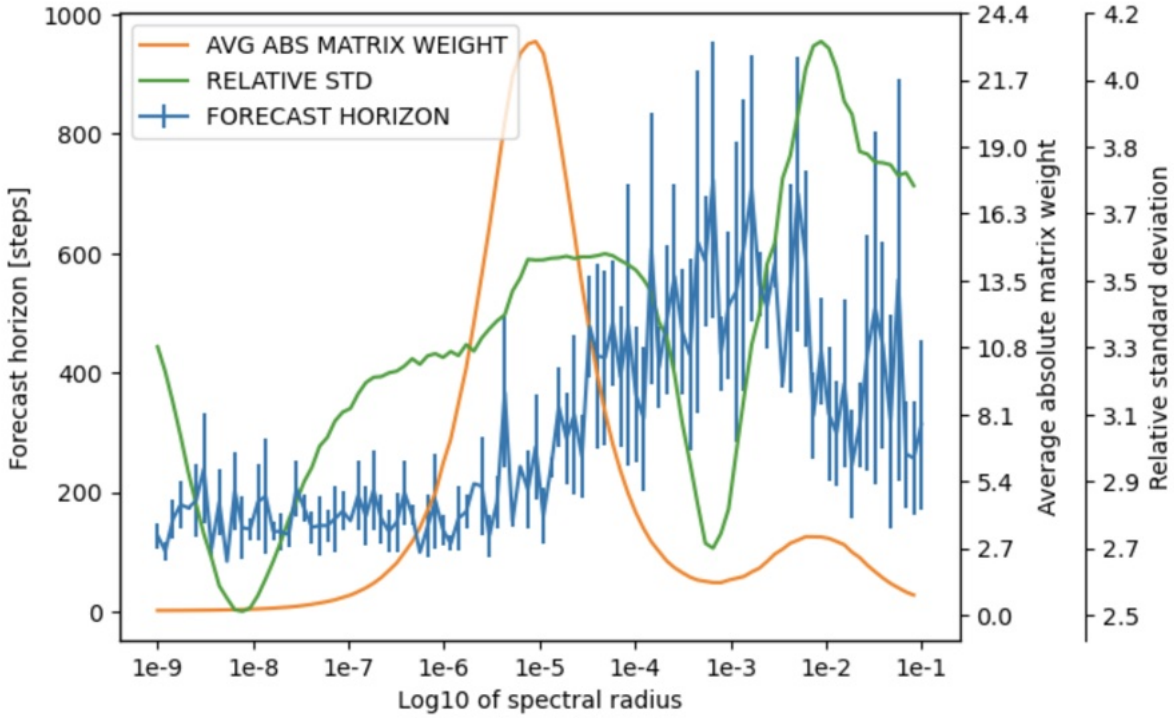


Figure 5.45: Variation of spectral radius for Halvorsen-specific complex Minimal-RC

weight of W_{out} exhibits a sharp peak and then decreases. For the best-performing spectral radius around $\rho = 10^{-3}$, a clear dip in the relative standard deviation is observed. There is also a local minimum in the average absolute weight of W_{out}. While the exact influence of the spectral radius on the forecast horizon and output matrix weights remains unclear, a correlation is evident in this specific example.

5.10 Fixed Points

A problem encountered during prediction with the regular Minimal-RC architecture (no complex extension) was the formation of fixed points, meaning that once the trajectory reached such a point, it remained there indefinitely—contrary to the behavior observed in the training data. These fixed points appeared randomly in two different systems. The starting point clearly plays a key role in the emergence of fixed points. This section aims to provide a better understanding of the mechanisms underlying the formation of fixed points during prediction.

One system in which this problem was particularly pronounced is the Aizawa system [62]. The Aizawa attractor is a system that involves higher-order nonlinearities and is defined by:

$$\dot{x} = (z-b)x - dy \tag{5.44}$$

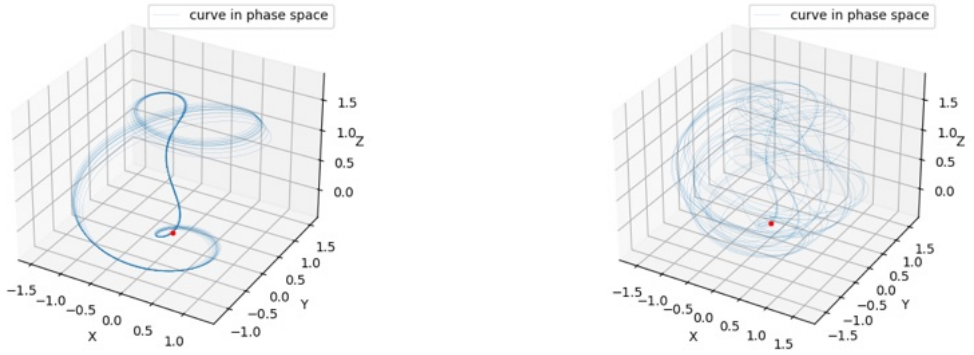
$$\dot{y} = dx + (z-b)y \tag{5.45}$$

$$\dot{z} = c + az - z^3/3 - (x^2+y^2)(1+ez) + fzx^3 \tag{5.46}$$

With $a = 0.95$, $b = 0.7$, $c = 0.6$, $d = 3.5$, $e = 0.25$, $f = 0.1$.

By applying the insights gained from previous sections, a generalized reservoir state of the form [1, 2, 3, 4] was used, as all of these nonlinearities are present in the equations. This choice led to good forecast horizons.

Figure 5.46 presents both the training data and the prediction for 20,000 steps for the Aizawa system, using a spectral radius $\rho = 10^{-1}$, with the fixed point indicated in red. Figure 5.47 displays the prediction.



(a) Data (b) Prediction

Figure 5.46: Aizawa attractor with fixed point in red

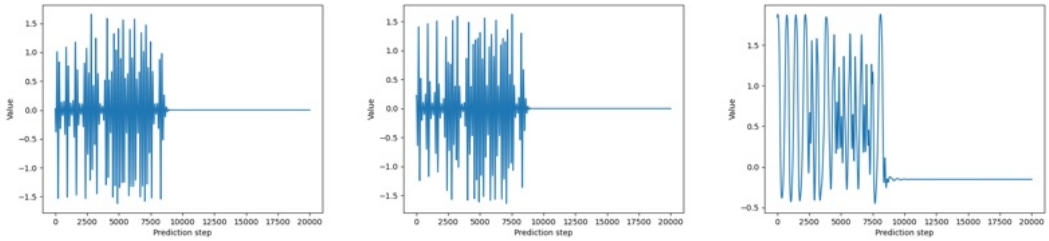
Formally, a fixed point is given by:

$$\vec{r}(t+1) = \vec{r}(t) \tag{5.47}$$

In our framework, the next data point is calculated by:

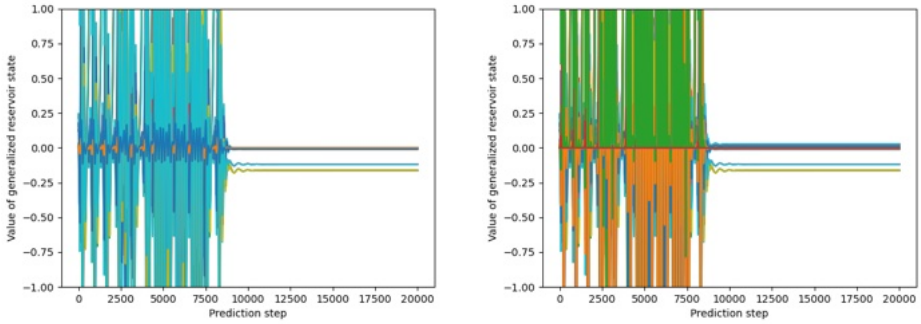
$$\vec{u}(t+1) = \boldsymbol{W}_{out}\vec{r}(t) \tag{5.48}$$

This means that, once the fixed point is reached, either the generalized reservoir state $\vec{r}(t)$ remains unchanged indefinitely—producing the same output at every iteration—or multiple distinct generalized reservoir states yield the same output when multiplied by $\boldsymbol{W}_{out}$. Figure 5.48 displays all values of the



(a) X coordinate (b) Y coordinate (c) Z coordinate

Figure 5.47: Prediction for 20.000 steps



(a) Linear reservoir state (b) Generalized reservoir state

Figure 5.48: All values of reservoir state vector plotted for the prediction

reservoir state vector plotted over the entire prediction period.

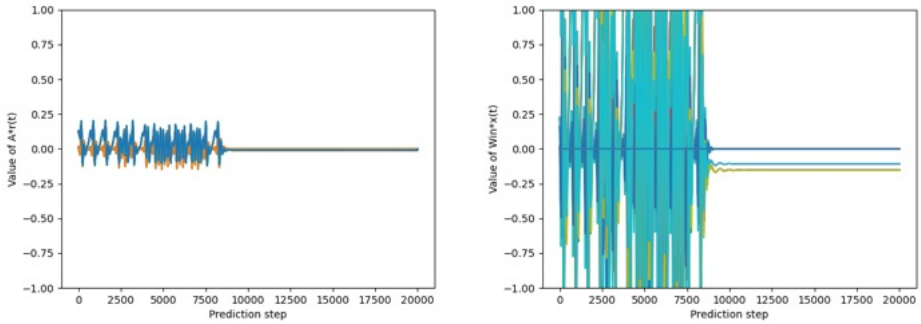
It becomes evident that, once the fixed point is reached, both the linear and generalized reservoir state remain constant. Formally:

$$\vec{r}(t+1) = \vec{r}(t) \tag{5.49}$$

The reservoir state is calculated as follows:

$$\vec{r}(t+1) = \boldsymbol{A}\vec{r}(t) + \boldsymbol{W}_{in}\vec{u}(t) \tag{5.50}$$

Figure 5.49 presents all the values of $\boldsymbol{A}\vec{r}(t)$ and $\boldsymbol{W}_{in}\vec{u}(t)$ plotted over the entire prediction period.



(a) $\boldsymbol{A}\vec{r}(t)$ (b) $\boldsymbol{W}_{in}\vec{u}(t)$

Figure 5.49: All values of $\boldsymbol{A}\vec{r}(t)$ and $\boldsymbol{W}_{in}\vec{u}(t)$ plotted for the prediction

It appears that, once the fixed point occurs, the primary contribution to the new reservoir state comes

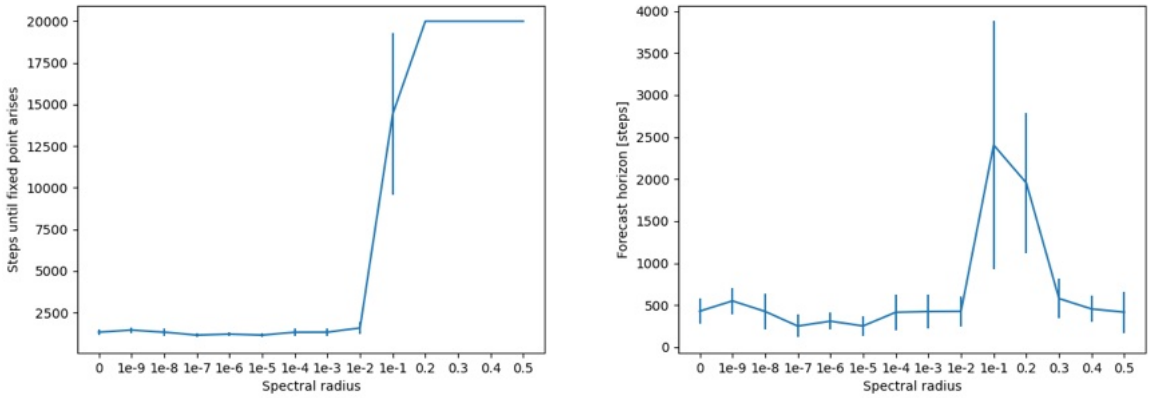
from $\boldsymbol{W}_{in}\vec{u}(t)$, as $\boldsymbol{A}\vec{r}(t)$ is nearly zero. Both $\boldsymbol{W}_{in}\vec{u}(t)$ and the linear reservoir state exhibit identical values. These values are determined by the z coordinate, since the x and y coordinates are nearly zero at the fixed point. When multiplied by $\boldsymbol{W}_{in}$, which contains the w vector, three different values are produced for a block size $b = 3$, one of which is zero. This behavior was also observed in another example involving the prediction of the x coordinate of the Halvorsen system, which was reconstructed using time delay embedding. Further details can be found in the appendix [B.7.1].

Somehow fixed points arise, when $\boldsymbol{A}\vec{r}(t)$ becomes zero or nearly zero. The values of the next reservoir state arise solely from the most recent data point, which remains unchanged in every iteration. Since the reservoir state no longer changes, the predicted data points also remain constant. This suggests that the contribution of $\boldsymbol{A}\vec{r}(t)$ is essential for dynamic evolution.

The key question is why $\boldsymbol{A}\vec{r}(t)$ yields zero values. One point to consider is that, for a spectral radius of $\rho = 0$, the term $\boldsymbol{A}\vec{r}(t)$ also becomes zero. However, in that scenario, W_{out} is trained to compensate for this, as it is also zero during the training loop. $\boldsymbol{A}$ involves multiplications by ρ/b, which means its values tend to be small in all cases. For $\boldsymbol{A}\vec{r}(t)$ to yield values close to zero, the reservoir state must become sufficiently small. Therefore, it is likely not a coincidence that the fixed point arises at very small values.

It was observed that fixed points emerge after fewer steps when using smaller spectral radii for the Aizawa system. This can be explained by the fact that multiplication by ρ/b with smaller spectral radii causes the values to approach zero more rapidly, even if the reservoir state values are initially larger. Thus, values close to zero are reached more quickly.

Figure 5.50 displays the number of steps required to reach the fixed point, as well as the forecast horizons for several spectral radii in the Aizawa system. A fixed point is considered to be reached once the value remains within an interval of 0.01 for at least 200 steps. Five realizations were trained and predicted.



(a) Fixed point

(b) Forecast horizon

Figure 5.50: Fixed point and forecast horizon for several spectral radii

For spectral radii $\rho \geq 0.2$, no fixed point arises, however, the forecast horizon also decreases in this range. Additionally, the differences in Lyapunov exponent and correlation dimension increase in this region (appendix [B.7.2]). The best-performing spectral radius is $\rho = 0.1$, which yields the largest forecast horizon and also requires the most steps to reach the fixed point (fixed point arises however). It should also be noted that the x-axis is not spaced according to its actual values.

Another experiment was conducted using the best-performing spectral radius, $\rho = 0.1$. The aim was to detect any anomalies in the output matrices when fixed points arise during prediction. To this end, four categories were defined based on the time taken to reach a fixed point: category one included scenarios where a fixed point was reached within 0 to 5,000 steps, category two, within 5,000 to 10,000 steps, category three, within 10,000 to 15,000 steps, and the last category, after more than 15,000 steps. 100 realizations were performed, all starting from the same initial condition. Of these, 41 realizations fell

into category one, 27 into category two, 20 into category three, and 12 into the last category.A similar observation was made when using a slightly different starting point: again all four categories were represented. This is surprising, as one would typically expect to obtain the same result each time when training on the same dataset.

Interestingly, even though fixed points were reached after very different numbers of steps, there were almost no differences in the output matrices across the categories. Metrics such as matrix rank, singular values, and average matrix weights were nearly identical in all cases. This finding further highlights the significance of the term $\boldsymbol{A}\vec{r}(t)$: the absence of anomalies in the output matrices suggests that the occurrence of fixed points is primarily driven by the near-zero values in $\boldsymbol{A}\vec{r}(t)$, rather than by any structural changes in the output matrices themselves. This may be due to an intrinsic mechanism in the system or could arise by coincidence, but it underscores the critical role of $\boldsymbol{A}\vec{r}(t)$ in the formation of fixed points.

Additional experiments were conducted to determine whether the fixed point is an intrinsic feature of the Aizawa system or merely a good local approximation produced by the Minimal-RC. The same dataset was used for 100 realizations with a spectral radius of $\rho = 0.1$, introducing small variations to the starting point in the prediction loop by randomly drawing a value from the interval $[-10^{-4}, 10^{-4}]$ and adding it to the initial point of the prediction. From this perturbed initial state, the actual trajectory was computed and compared to the predicted one. The aim was to assess whether these tiny variations would also lead to fixed points in the calculated data, or if this phenomenon is unique to the Minimal-RC prediction. The correlation dimension was calculated for both trajectories, with the results shown in Figure 5.51.

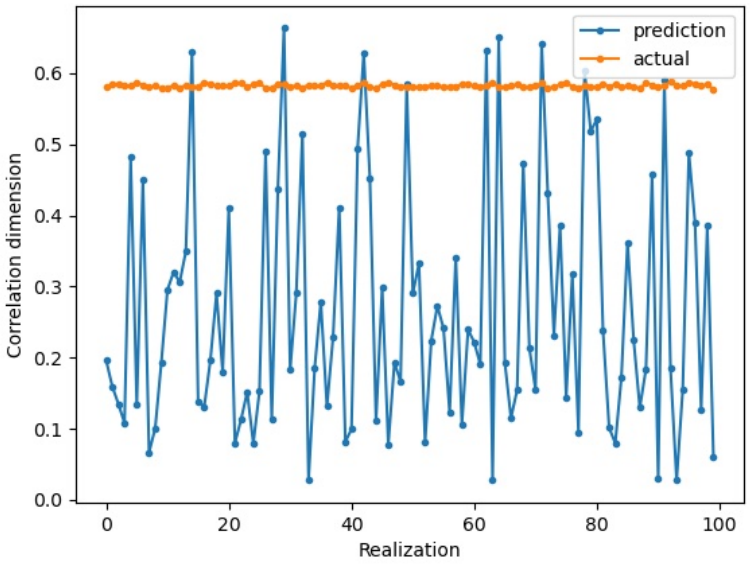


Figure 5.51: Actual vs. predicted correlation dimension for slightly varied staring conditions in the prediction loop

It becomes clear that the actual correlation dimension remains fairly consistent across all realizations, with only minor variations resulting from the small differences in starting conditions. In contrast, the correlation dimension of the predicted trajectories exhibits large fluctuations, indicating significant differences in attractor shapes due to the varying times at which fixed points occur. This suggests that fixed points do not arise intrinsically in the Aizawa system—as evidenced by the constant correlation dimensions in the actual data—but are instead a local approximation produced by the Minimal-RC.

Chapter 6

Summary

This section summarizes the key findings and insights gained from a comprehensive investigation of minimal reservoir computing (Minimal-RC) applied to a variety of nonlinear and chaotic systems. Through systematic experimentation, theoretical analysis, and detailed examination of model behavior, several important principles and empirical observations have emerged. The following points encapsulate the most significant results and their implications for the understanding and further development of Minimal-RC.

The predictive quality of minimal reservoir computing (Minimal-RC) is maximized when the nonlinearities present in the generalized reservoir state closely match all the relevant nonlinearities in the governing system equations—not just the highest order. This was demonstrated for a variety of systems, including the Lorenz, Halvorsen, Arneodo, and Aizawa system. Carefully constructed input combinations, tailored to the terms present in the system equations, generally yield better predictions. However, this approach presupposes knowledge about the system equations, which is not always available in practical, real-world scenarios. This also presents a Catch-22: knowledge of the system equations is a prerequisite for making accurate predictions, yet possessing this knowledge would eliminate the need for machine learning in the first place. Nevertheless, this realization provides valuable insight into the functioning and limitations of Minimal-RC. While including only relevant terms can help Minimal-RC avoid learning from irrelevant patterns and may improve performance—especially in complex systems—it is not universally the case that superfluous information always leads to bad predictions. In some situations, the presence of additional, redundant input terms does not necessarily degrade prediction quality.

Minimal-RC struggles to predict systems created by mixing two or more sets of governing equations if all relevant nonlinearities are not represented in the generalized reservoir state. It is very likely that the correct combination of terms could yield good predictions, but for such complicated systems, these combinations are unknown and probably numerous. As a result, finding the appropriate set of input combinations is challenging. Time delay embedding offers a solution for reconstructing the underlying system dynamics in these cases, but this approach is quite difficult to implement effectively within the Minimal-RC framework.

Introducing complex-valued generalized reservoir states and custom functions (inspired by the structure of the governing equations) can improve prediction quality for systems with nonlinearities beyond simple polynomials, such as the Ikeda and Thomas system. However, the observed phenomenon—that prediction quality improves when the relevant terms are present in the generalized reservoir state—is not restricted to the complex extension, it was simply demonstrated with this approach. Even with such extensions, perfect predictions remain challenging if all relevant terms are not included.

The architecture of Minimal-RC can be customized for specific systems by designing the generalized reservoir state and the input combinations to reflect the structure of the system's equations. This adaptability is crucial for extending Minimal-RC to a broader class of dynamical systems.

A particularly important and insightful observation relates to the activation patterns in the output matrix weights. When Minimal-RC is well-tuned and achieves good predictions, the output matrix exhibits strong, stable activation in those columns that correspond to the terms present in the system equations. For example, if a system equation includes a term like xz, the corresponding column in the

output matrix tends to show consistent, elevated weights across multiple realizations. This parallelism between matrix activation and governing equation structure not only serves as a diagnostic tool for model interpretation but also underscores the necessity of including all relevant information in the generalized reservoir state. It is important to note that, while a correlation is present, there is not an exact one-to-one correspondence. Furthermore, columns associated with irrelevant or unnecessary nonlinearities often display fluctuating or divergent weights.

In summary, all of these findings indicate that the essential information present in the data must be incorporated and preserved throughout the processing pipeline of Minimal-RC. If important terms or features are omitted, the model will be unable to make accurate predictions, regardless of other architectural or parametric choices.

The guiding question of this thesis can hence be answered by the observation, that the sharp transition in prediction quality between order 1 and 2 in the introductory plot corresponds to the inclusion of essential information at exactly order 2, reflecting all the nonlinearities present in the system equations.

The spectral radius controls the memory of the reservoir and has a significant impact on prediction quality. There is a regime of spectral radius values in which prediction quality (as measured by the forecast horizon) is maximized for most systems. The choice of spectral radius also influences the distribution and stability of output matrix weights. It is clear that the size of the matrix weights varies with the spectral radius, but the relative standard deviation of the weights remains in the same range across different spectral radii. This stability in relative standard deviation enables the model to make reliable predictions even as the absolute size of the weights changes.

Fixed points can emerge during prediction, resulting in the reservoir state converging to a single value and the model outputting the same prediction indefinitely, thus failing to capture the system's dynamics. For the Aizawa system, the spectral radius is particularly relevant to the formation of fixed points: smaller spectral radii lead to faster convergence to a fixed point. After a fixed point is reached, the reservoir state values, as well as all the terms used to calculate them, remain constant. Of particular significance is that the term $\boldsymbol{A}\vec{r}(t)$ becomes zero after the fixed point is reached, indicating that the dynamic evolution of the reservoir is lost and the model is no longer processing all information needed to make accurate predictions.

In conclusion, this thesis has provided valuable insights into the mechanisms, limitations, and possibilities of minimal reservoir computing for predicting nonlinear and chaotic systems. The results highlight both the strengths of the approach and the subtle complexities involved in its design and application. However, many aspects remain unclear, such as the systematic identification of optimal input combinations for arbitrary systems, the role of redundancy in input features, and the deeper theoretical reasons behind certain empirical phenomena. These open questions point to promising directions for further scientific investigation and methodological refinement in the future.

Appendix A

References

1. E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963
2. H. Ma, D. Prosperino & C. Räth, "A novel approach to minimal reservoir computing," *scientific reports*, 2023
3. D. Prosperino, H. Ma & C. Räth, "Tailored minimal reservoir computing: On the bidirectional connection between nonlinearities in the model and in data," *AIP Publishing*, 2025
4. R. M. May, "Simple mathematical models with very complicated dynamics," *Nature*, vol. 261, no. 5560, pp. 459-467, 1976
5. S. Chen, S. Feng, W. Fu & Y. Zhang, "Logistic Map: Stability and Entrance to Chaos," *Journal of Physics: Conference Series*, vol. 2014, no. 012009, 2021
6. "Logistic map," *Wikipedia*, 03.02.2026, `https://en.wikipedia.org/wiki/Logistic_map`
7. P. J. Woolf, et al., "Chemical Process Dynamics and Controls, 10.4: Using Eigenvalues and Eigenvectors to Find Stability and Solve ODEs," *University of Michigan*, 2009
8. A. Altland & J. v. Delft, "Mathematics for Physicists: Introductory Concepts and Methods," *Cambridge University Press*, 1st ed., pp. 308-313, 2019
9. S. H. Strogatz, "Nonlinear Dynamics and Chaos With Applications to Physics, Biology, Chemistry and Engineering," *CRC Press*, 1st ed., pp. 146-151, 2018
10. M. B. de Oliveira, "Mathematics for Complex Systems," *Universtiy of Southampton*, Lecture 5, 2014
11. H. Goldstein, "Classical Mechanics: Pearson New International Edition," *Pearson*, pp. 491-494, 2013
12. P. A. Dirmeyer, C. A. Schlosser, K. L. Brubaker, "Precipitation, Recycling, and Land Memory: An Integrated Analysis," *Journal of Hydrometeorology*, vol. 10, no. 1, pp. 278–288, 2009
13. J. H. Frank, "Advances in imaging of chemically reacting flows," *Journal of Chemical Physics*, vol. 154, no. 4, 040901, 2021
14. T. D. Sauer, "Attractor reconstruction," *Scholarpedia*, vol. 1, no. 10, pp. 1727, 2006
15. F. Takens, "Detecting strange attractors in turbulence," *Lecture Notes in Mathematics, Springer-Verlag*, vol. 898, pp. 366–381, 1981
16. Y. LeCun, Y. Bengio G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015
17. R. M. Schmidt, "Recurrent Neural Networks (RNNs): A gentle Introduction and Overview," *arXiv*, 1912.05911, 2019

18. C. Olah, "Understanding LSTM networks," *Stanford University*, 2015

19. P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550-1560, 1990

20. S. Hochreiter, J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997

21. J. Chung, C. Gulcehre, K. Cho, Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv*, 1412.3555, 2014

22. R. J. Trudeau, "Introduction to Graph Theory," *Dover Publications*, ed. 2, 1993

23. K. Kepesidis, "Lecture: Deep Learning for Physicists," *LMU Munich, Physics Department*, Lecture 8, 2023

24. P. Erdős, A. Rényi, "On random graphs," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290-297, 1959

25. A. L. Barabási, R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509-512, 1999

26. A. Capocci, V. D. P. Servedio, F. Colaiori, L. S. Buriol, D. Donato, S. Leonardi, G. Caldarelli, "Preferential attachment in the growth of social networks: the case of Wikipedia," *Physical Review E*, vol. 74, no. 3, 2006

27. G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano & A. Hirose, "Recent Advances in Physical Reservoir Computing: A Review," *arXiv*, 1808.04962, 2019

28. H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," *German National Research Center for Information Technology (GMD)*, GMD Report 148, 2001

29. A. Haluszczynski, D. Köglmayr, C. Räth, "Controlling dynamical systems to complex target states using machine learning: next-generation vs. classical reservoir computing," *arXiv*, 2307.07195, 2023

30. J. Herteux, C. Räth, "Breaking Symmetries of the Reservoir Equations in Echo State Networks," *arXiv*, 2010.07103, 2020

31. Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, E. Ott, "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 4, 2017

32. D. Gerth, "A new interpretation of (Tikhonov) regularization," *arXiv*, 2103.08218, 2021

33. A. E. Hoerl, R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, vol. 12, no. 1, pp. 55-67, 1970

34. J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, E. Ott, "Using Machine Learning to Replicate Chaotic Attractors and Calculate Lyapunov Exponents from Data," *arXiv*, 1710.07313, 2017

35. D. Duncan, C. Räth, "Optimizing the combination of data-driven and model-based elements in hybrid reservoir computing," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 10, 2023

36. D. J. Gauthier, E. Bollt, A. Griffith, W. A. S. Barbosa, "Next Generation Reservoir Computing," *arXiv*, 2106.07688, 2021

37. C. Runge, "Über die numerische Auflösung von Differentialgleichungen," *Mathematische Annalen*, vol. 46, pp. 167–178, 1895

38. W. Kutta, "Beitrag zur näherungsweisen Integration totaler Differentialgleichungen," *Zeitschrift für Mathematik und Physik*, vol. 46, pp. 435–453, 1901

39. "Runge-Kutta methods," *Wikipedia*, 31.01.2026, `https://en.wikipedia.org/wiki/RungeKutta_methods`

40. E. Fehlberg, "New high-order Runge-Kutta formulas with step size control for systems of first-and second-order differential equations," *Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 44, no. 10-11, pp. 17–19, 1964

41. A. Haluszczynski, C. Räth, "Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 10, 2019

42. A. Haluszczynski, J. Aumeier, J. Herteux, C. Räth, "Reducing network size and improving prediction stability of reservoir computing," *arXiv*, 2003.03178, 2020

43. D. Goldberg, "What Every Computer Scientist Should Know About Floating-Point Arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5-48, 1991

44. P. Grassberger, I. Procaccia, "Measuring the strangeness of strange attractors," *Physica D: Nonlinear Phenomena*, vol. 9, no. 1, pp. 189-208, 1983

45. P. Grassberger, "Generalized dimensions of strange attractors," *Physica D: Nonlinear Phenomena*, vol. 13, no. 1, pp. 34-46, 1983

46. M. T. Rosenstein, J. J. Collins, C. J. D. Luca, "A practical method for calculating largest Lyapunov exponents from small data sets," *Physica D: Nonlinear Phenomena*, vol. 65, no. 1, pp. 117-134, 1993

47. M. Sandri, "Numerical Calculations of Lyapunov Exponents," *The Mathematica Journal*, vol. 6, no. 3, pp. 78-84, 1996

48. G. Cassoni, A. Cocco, A. Tamer, A. Zanoni, P. Masarati, "Rotorcraft stability analysis using Lyapunov characteristic exponents estimated from multibody dynamics," *Springer Nature Link*, vol. 15, pp. 703–719, 2024

49. I. Laut, C. Räth, "Surrogate-assisted network analysis of nonlinear time series," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 26, no. 10, 2016

50. H. Ma, A. Haluszczynski, D. Prosperino, C. Räth, "Identifying causality drivers and deriving governing equations of nonlinear complex systems," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 10, 103128, 2022

51. O. E. Rössler, "An equation for continuous chaos," *Physics Letters A*, vol. 57, pp. 397–398, 1976.

52. DLR-KI, "scan," `https://github.com/DLR-KI/scan`, 2022 [GitHub repository]

53. "Proofs involving ordinary least squares," *Wikipedia*, 31.01.2026, `https://en.wikipedia.org/wiki/Proofs_involving_ordinary_least_squares`

54. A. Fitz, M. Klatt, "Reservoir computing for mixed chaotic systems: how time-delay embeddings lead out of a valley of unpredictability," *German Aerospace Center*, 2026 (UNPUBLISHED)

55. K. Ikeda, H. Daido & O. Akimoto, "Optical Turbulence: Chaotic Behavior of Transmitted Light from a Ring Cavity," *Physical Review Letters*, vol. 45, no. 9, pp. 709-712, 1980

56. S. L. Brunton, J. L. Proctor, J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *PNAS*, vol. 113, no. 15, pp. 3932–3937, 2016

57. M. Moradi, S. Panahi, E. M. Bollt, Y. Ch. La, "Data-driven model discovery with Kolmogorov-Arnold networks," *arXiv*, 2409.15167, 2024

58. Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, M. Tegmark, "KAN: Kolmogorov-Arnold Networks," *arXiv*, 2404.19756, 2024

59. R. Thomas, "Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis, 'labyrinth chaos'," *International Journal of Bifurcation and Chaos*, vol. 9, no. 10, pp. 1889-1905, 1999

60. J. C. Sprott, "Chaos and Time-Series Analysis," *Oxford University Press*, vol. 69, 2003

61. A. Arneodo, P. H. Coullet, E. A. Spiegel, C. Tresser, "Asymptotic Chaos," *Physica D (Nonlinear Phenomena)*, vol. 14, no. 3, pp. 327-347, 1985

62. W. F. Langford, "Numerical Studies of Torus Bifurcations," *International Series of Numerical Mathematics, Birkhäuser Verlag Basel*, vol. 70, pp. 285–295, 1984

Appendix B

Supplementary Plots

B.1 System Mixing

B.1.1 Autoregressive Process - Lorenz System

The Autoregressive process is generated by $a_n = s_n\sqrt{|s_n|}$, where $s_n = cs_{n-1} + \eta_n$. $c = 0.9$ and η_n is drawn from a gaussian random distribution. Every coordinate of the system ($\{x_n\}$, $\{y_n\}$, $\{z_n\}$) was mixed with the same Autoregressive process $\{a_n\}$. [49]

A shift from linear towards quadratic activation of the output matrices gets apparent. Also matrix weights stabilize for good predictions.

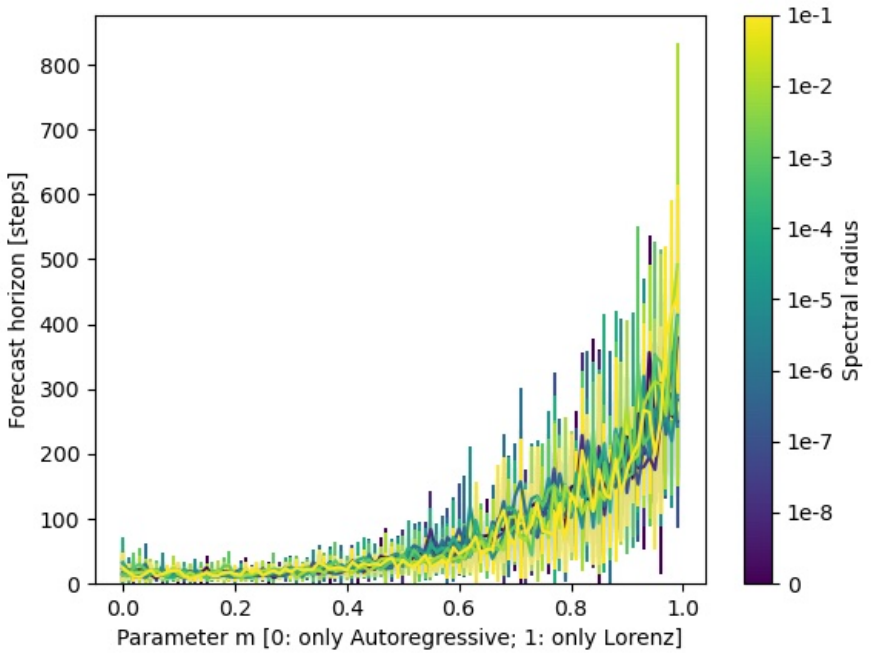


Figure B.1: Forecast horizon vs. mixing parameter m for several spectral radii

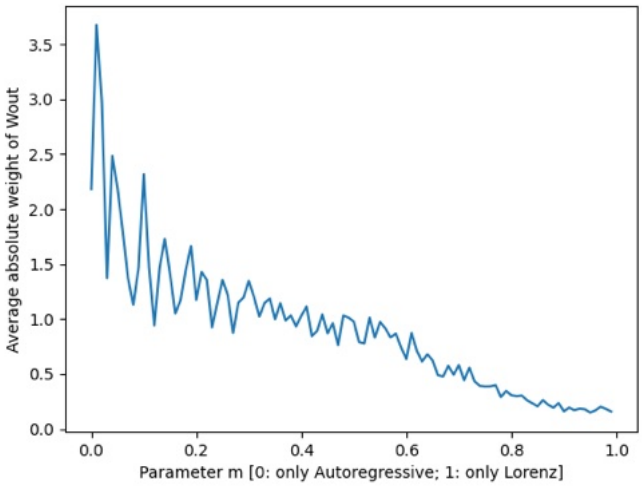
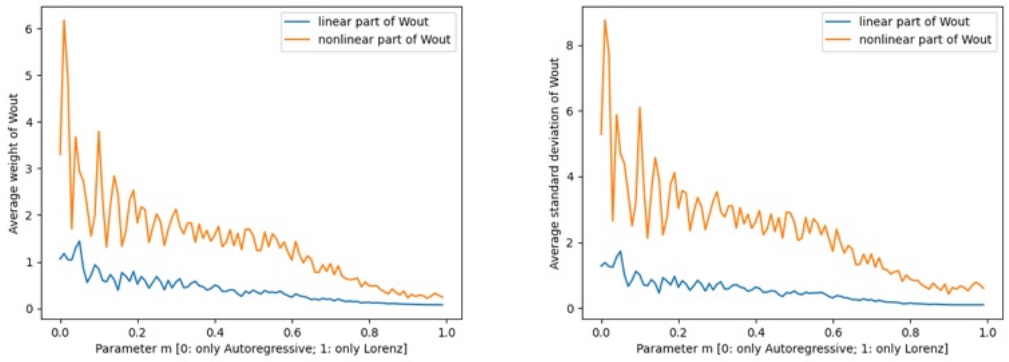
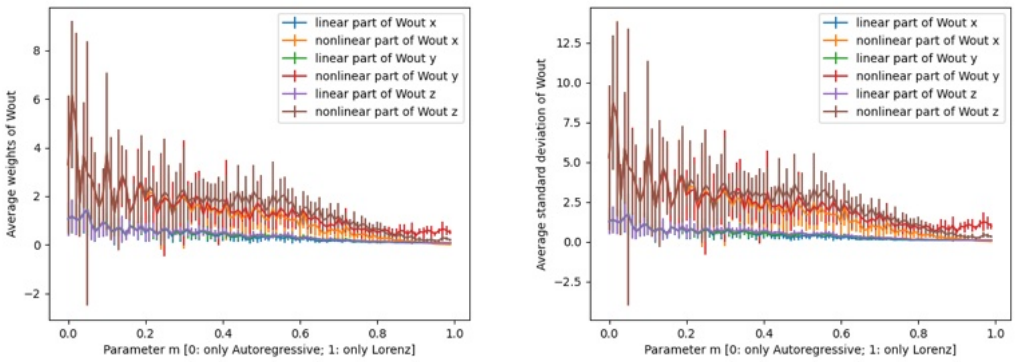


Figure B.2: Average absolute weight of W_{out} vs. mixing parameter m for spectral radius of $\rho = 10^{-7}$



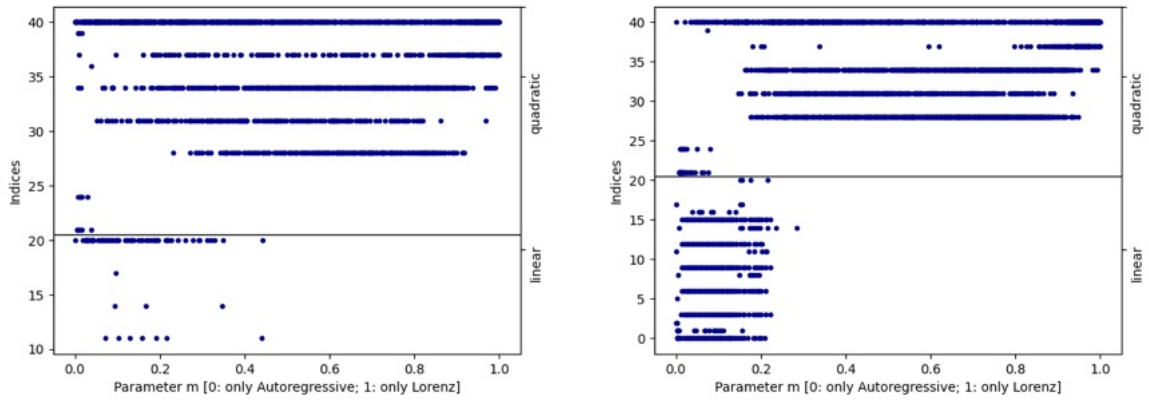
(a) Average absolute weights (b) Average standard deviation of weights

Figure B.3: Matrix weights for linear- and quadratic part of W_{out} for all values of m



(a) Average absolute weights (b) Average standard deviation of weights

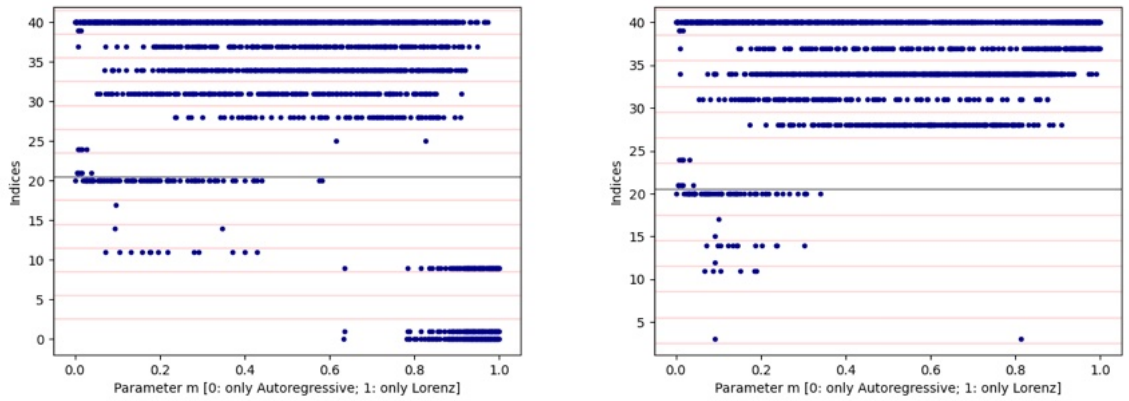
Figure B.4: Matrix weights for linear- and quadratic part and X-,Y- and Z part of W_{out} for all values of m



(a) Activation - absolute values

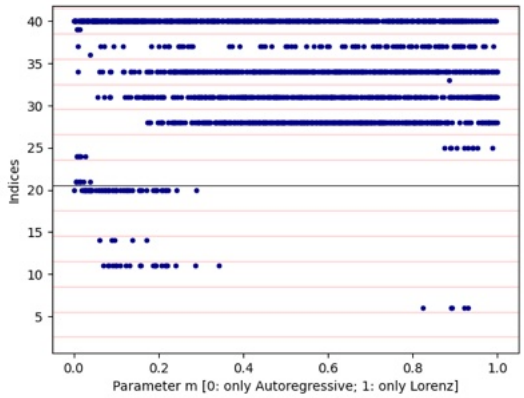
(b) Activation - standard deviation

Figure B.5: Activated matrix weights (within 70% of max value) of W_{out} for all values of m



(a) Maps to x coordinate

(b) Maps to y coordinate



(c) Maps to z coordinate

Figure B.6: Activated matrix weights (within 70% of max value) for all columns of W_{out} for all values of m

B.1.2 Ellipse - Halvorsen System

A smaller "Valley of Unpredictability" can be seen, with no real divergence of matrix weights. Matrix weights get bigger, however, for the unpredictable (or less predictable) mixed system. A shift from linear towards quadratic activation can be seen again. The elliptical system is defined as follows:

$$\dot{x} = \cos(t) \tag{B.1}$$

$$\dot{y} = \sin(t) \tag{B.2}$$

$$\dot{z} = \sin(t) \tag{B.3}$$

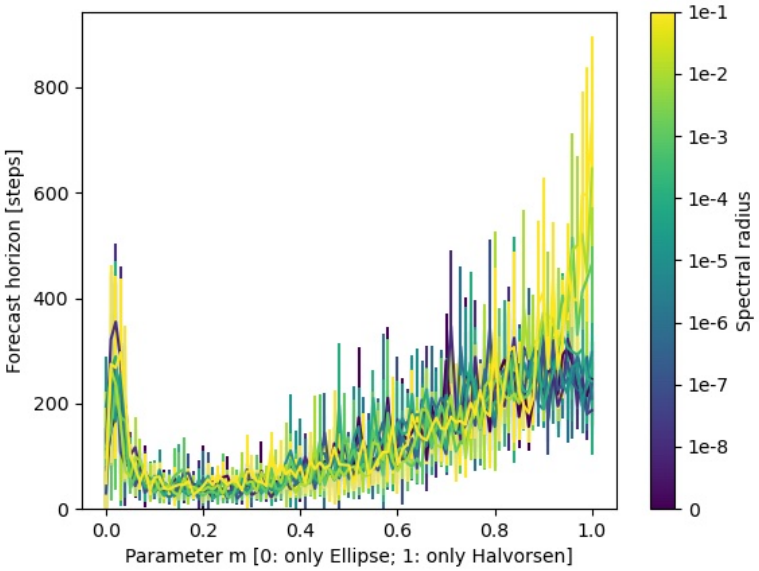


Figure B.7: Forecast horizon vs. mixing parameter m for several spectral radii

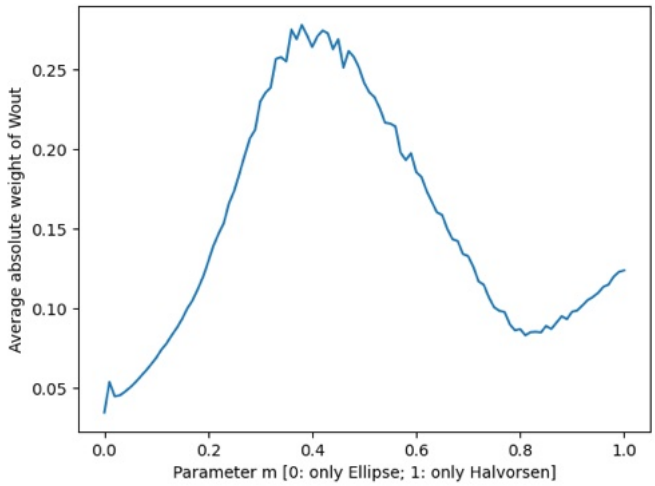
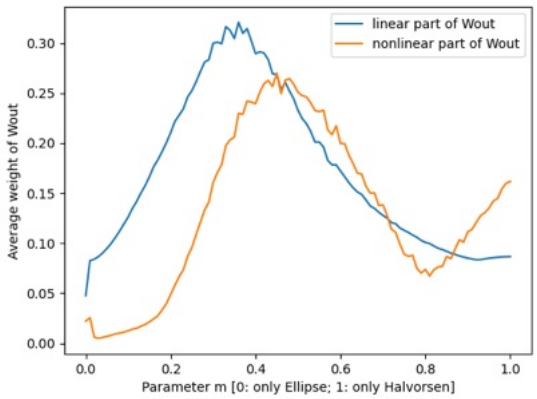
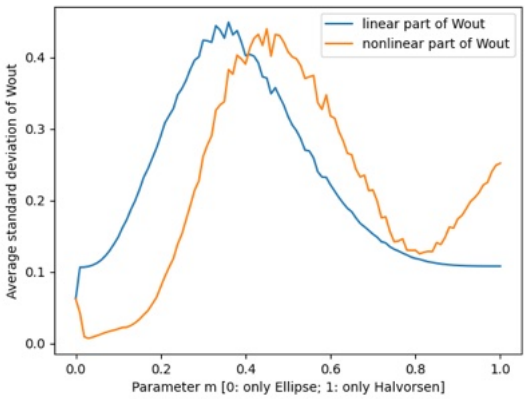


Figure B.8: Average absolute weight of W_{out} vs. mixing parameter m for spectral radius of $\rho = 1e^{-7}$

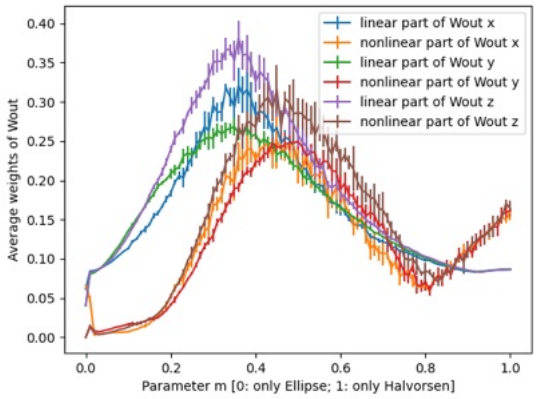


(a) Average absolute weights

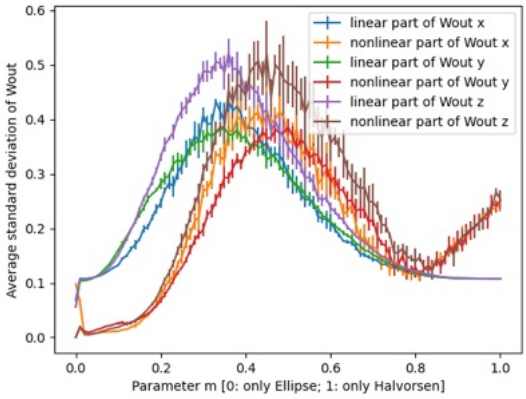


(b) Average standard deviation of weights

Figure B.9: Matrix weights for linear- and quadratic part of W_{out} for all values of m

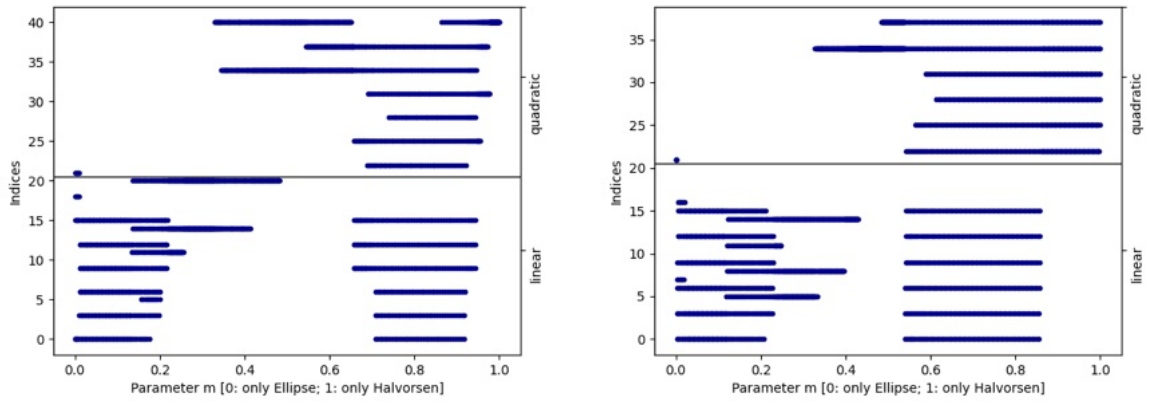


(a) Average absolute weights



(b) Average standard deviation of weights

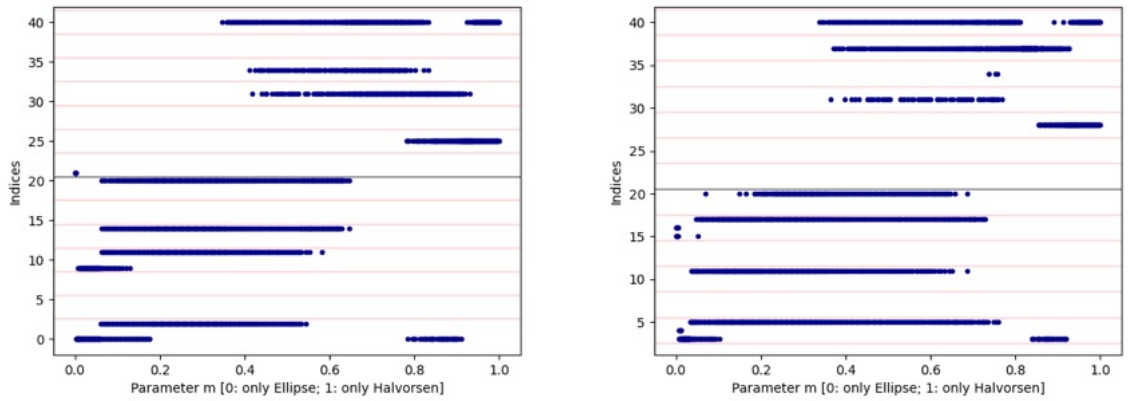
Figure B.10: Matrix weights for linear- and quadratic part and X-,Y- and Z part of W_{out} for all values of m



(a) Activation - absolute value

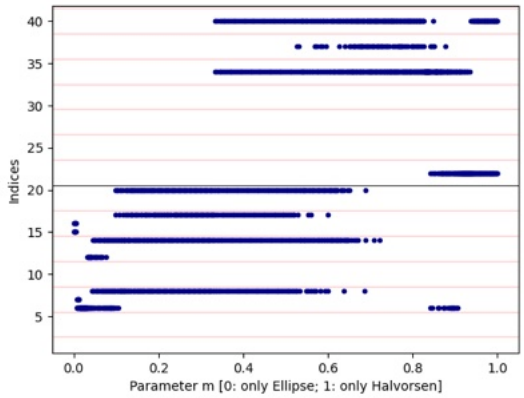
(b) Activation - standard deviation

Figure B.11: Activated matrix weights (within 70% of max value) of W_{out} for all values of m



(a) Maps to x coordinate

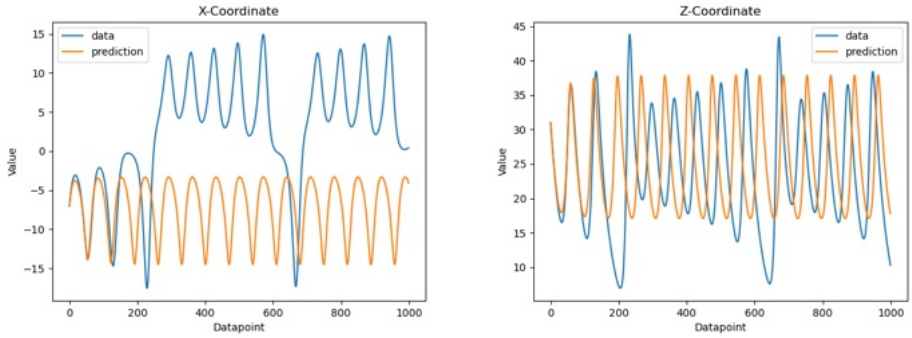
(b) Maps to y coordinate



(c) Maps to z coordinate

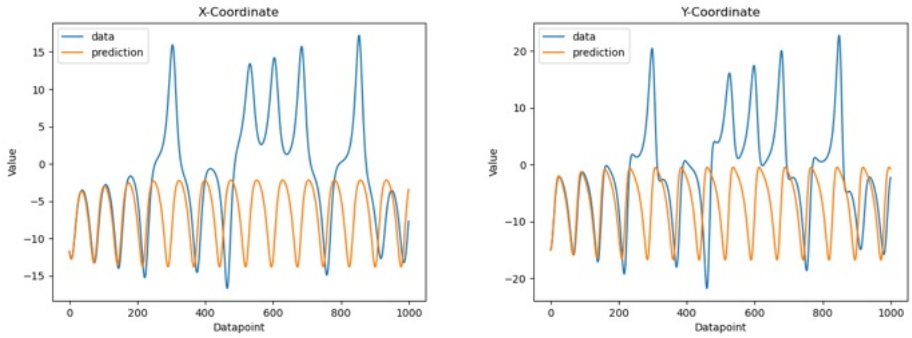
Figure B.12: Activated matrix weights (within 70% of max value) for all columns of W_{out} for all values of m

B.2 2D Predictions of 3D Systems



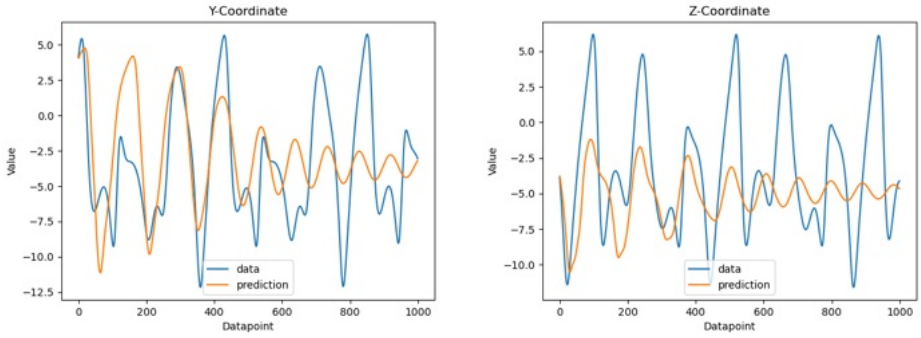
(a) First coordinate (b) Second coordinate

Figure B.13: Lorenz system with removed y-coordinate with [1,2,3,4,5,6,7] as nonlinearities and $\rho = 10^{-9}$



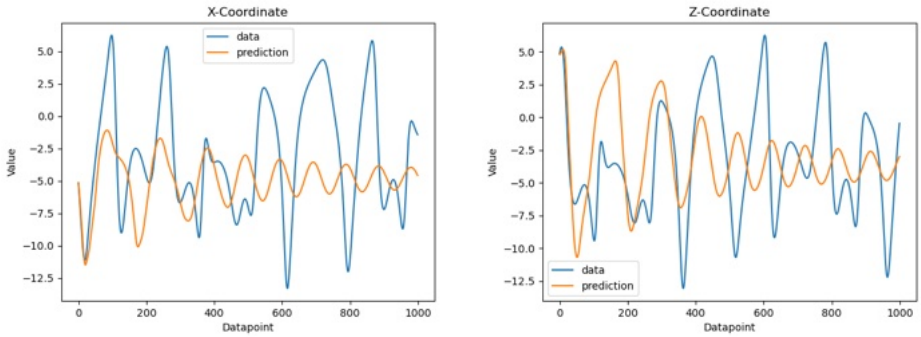
(a) First coordinate (b) Second coordinate

Figure B.14: Lorenz system with removed z-coordinate with [1,2,3,4,5,6,7] as nonlinearities and $\rho = 10^{-9}$



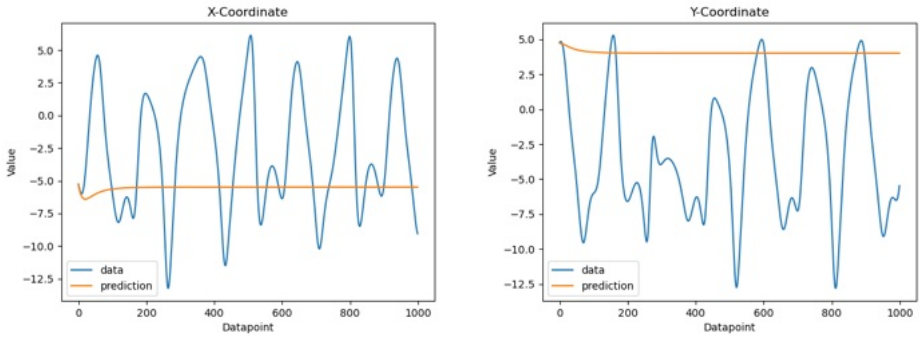
(a) First coordinate (b) Second coordinate

Figure B.15: Halvorsen system with removed x-coordinate with [1,2,3,4,5,6,7] as nonlinearities and $\rho = 10^{-9}$



(a) First coordinate (b) Second coordinate

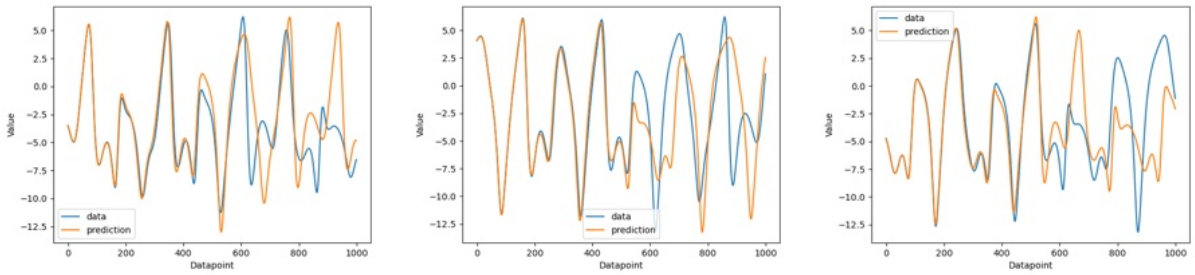
Figure B.16: Halvorsen system with removed y-coordinate with [1,2,3,4,5,6,7] as nonlinearities and $\rho = 10^{-9}$



(a) First coordinate (b) Second coordinate

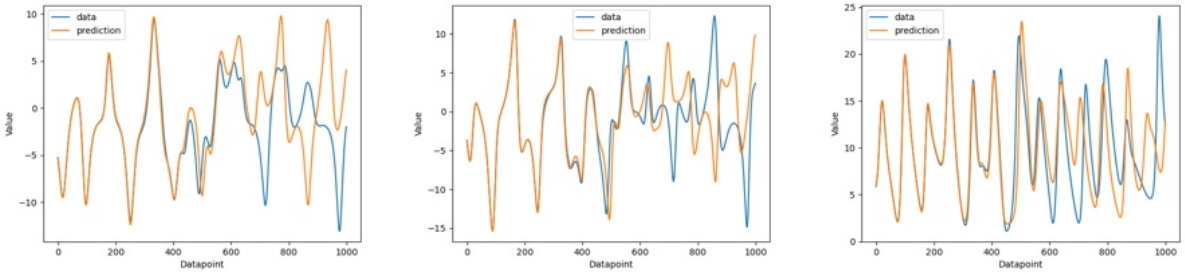
Figure B.17: Halvorsen system with removed z-coordinate with [1,2,3,4,5,6,7] as nonlinearities and $\rho = 10^{-9}$

B.3 6D - Mixed System



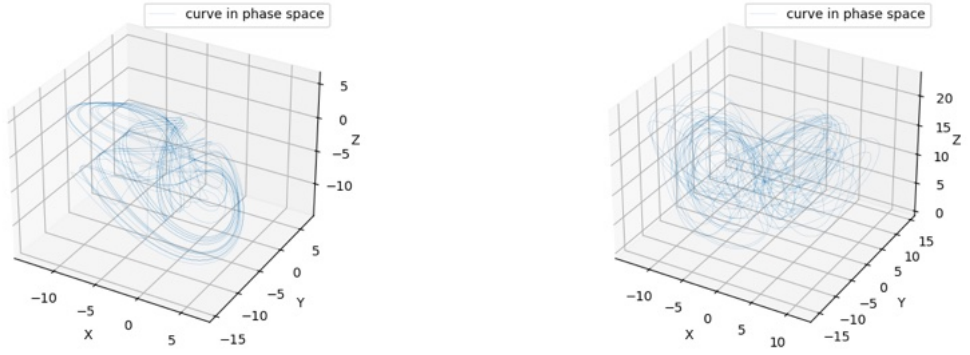
(a) X coordinate (b) Y coordinate (c) Z coordinate

Figure B.18: Prediction of the first three coordinates (1000 steps) - Halvorsen system



(a) X coordinate (b) Y coordinate (c) Z coordinate

Figure B.19: Prediction of the last three coordinates (1000 steps) - mixed system

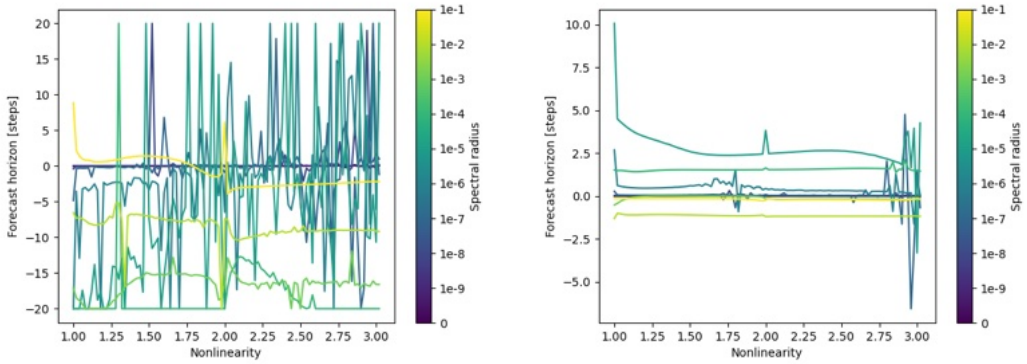


(a) Halvorsen system (b) Mixed system

Figure B.20: Prediction of Halvorsen- and mixed system as six dimensional input

B.4 Halvorsen System - Specific Training

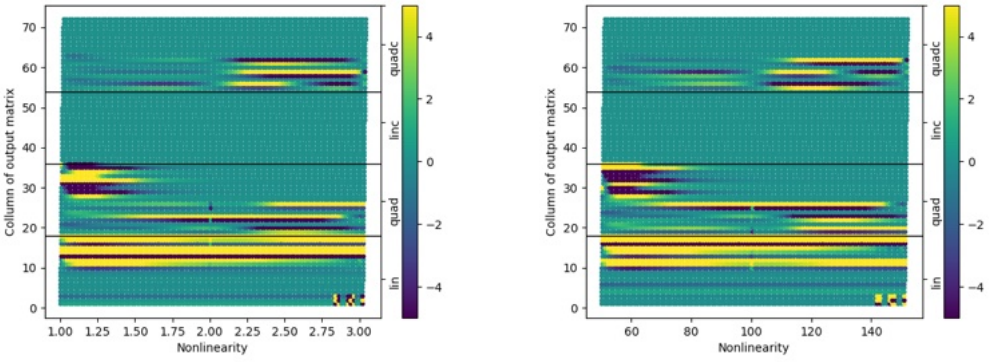
For the Halvorsen system, stabilized matrix weights can be seen across most "nonlinearities" in the generalized reservoir state, whereas the Lorenz system shows instability, since the information of xy, xz and yz is not present.



(a) Lorenz system

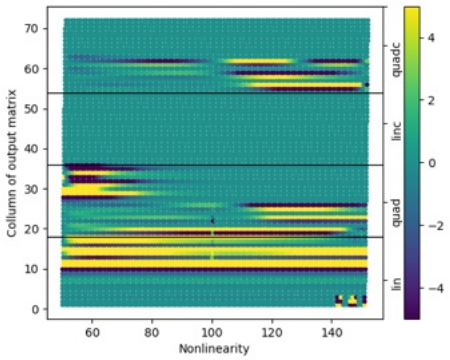
(b) Halvorsen system

Figure B.21: Average output matrix weights plotted against nonlinearities for Halvorsen specific input



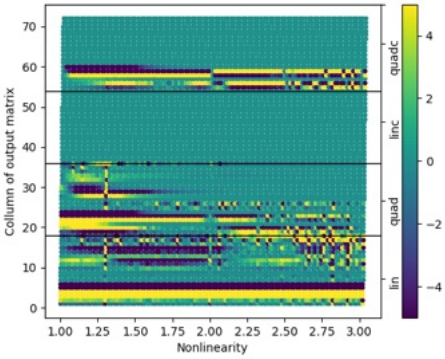
(a) Maps to x coordinate

(b) Maps to y coordinate

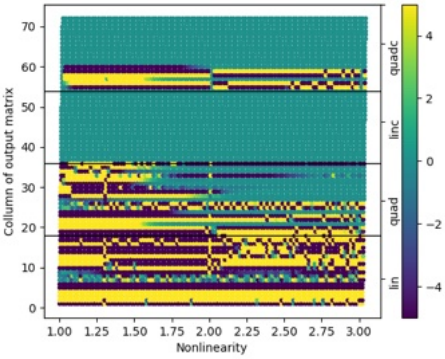


(c) Maps to z coordinate

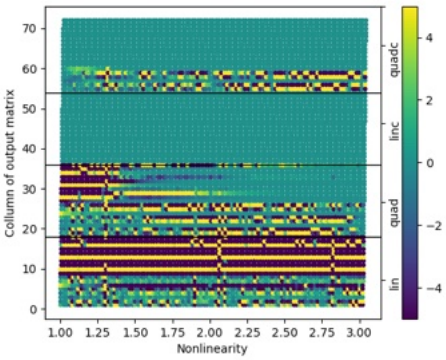
Figure B.22: Activated matrix collumns for Halvorsen system and Halvorsen specific input



(a) Maps to x coordinate



(b) Maps to y coordinate



(c) Maps to z coordinate

Figure B.23: Activated matrix collumns for Lorenz system and Halvorsen specific input

B.5 Arneodo System

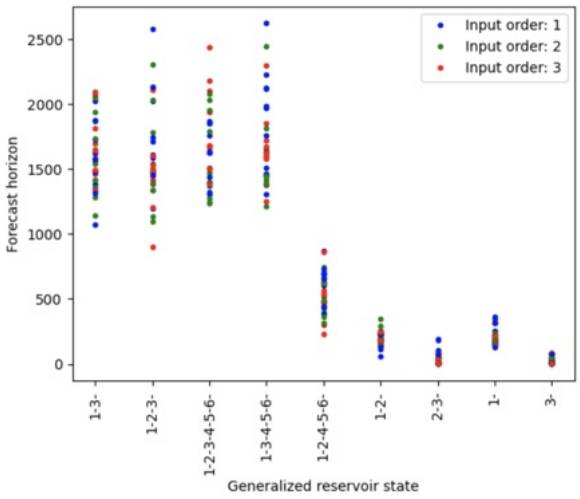
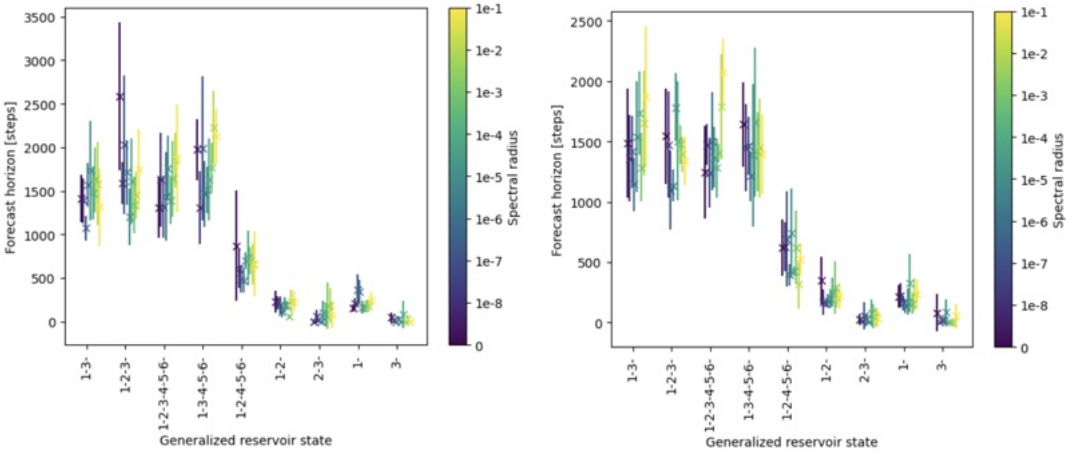
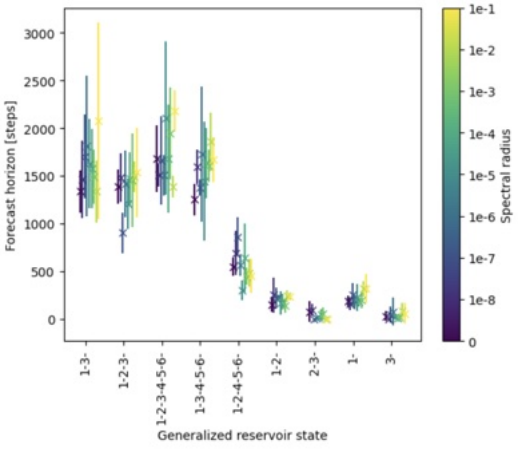


Figure B.24: FH of different generalized reservoir states for all input nonlinearities for complex extension



(a) Input nonlinearity of 1

(b) Input nonlinearity of 2



(c) Input nonlinearity of 3

Figure B.25: FH of different generalized reservoir states and spectral radii for all possible input nonlinearities for complex extension

B.6 Spectral Radius

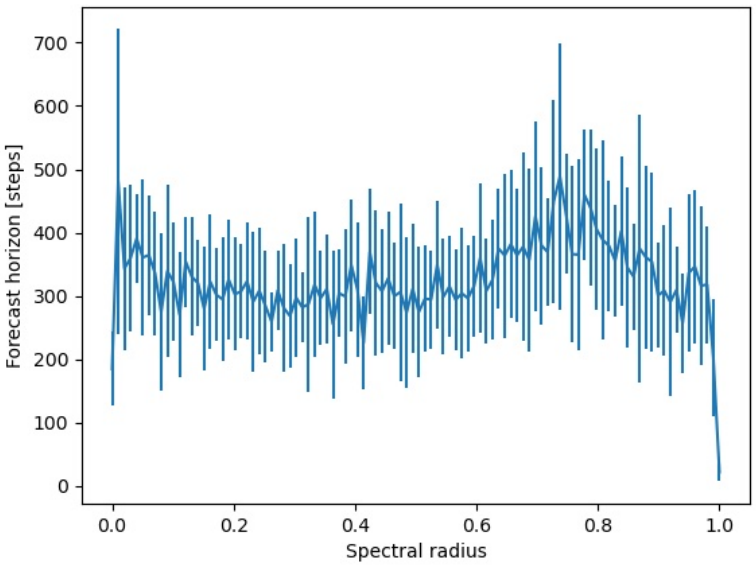
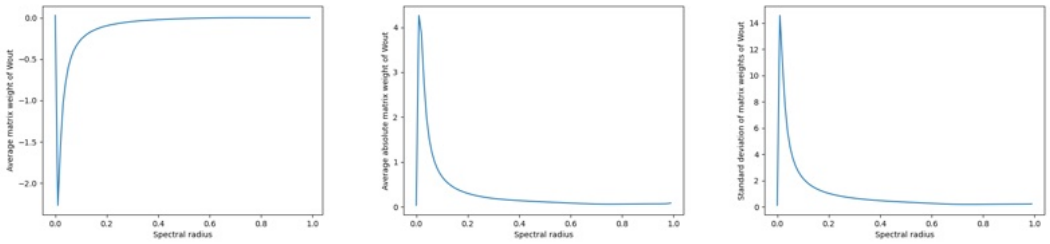


Figure B.26: FH vs. spectral radius for Halvorsen system for regular (not Halvorsen specific) complex Minimal-RC



(a) Average weight (b) Average absolute weight (c) Average standard deviation

Figure B.27: Average matrix weights vs. spectral radius for Halvorsen system for regular complex minimal-RC

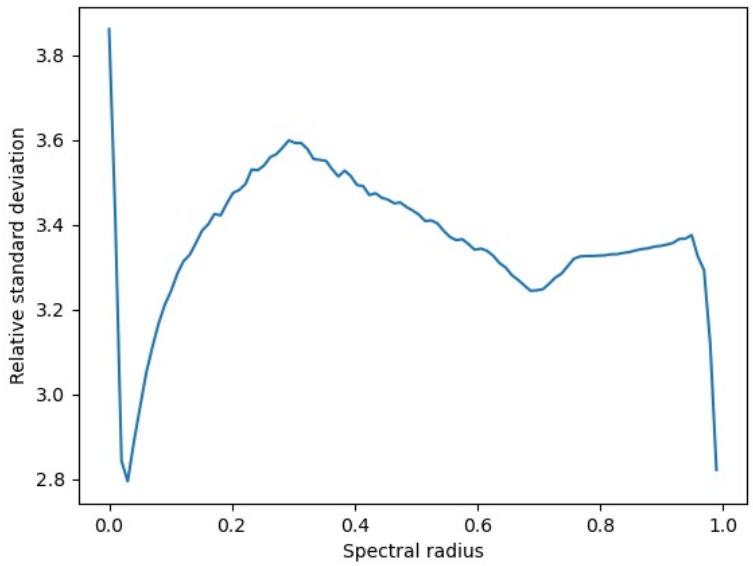


Figure B.28: Relative standard deviation vs. spectral radius for Halvorsen system for regular complex Minimal-RC

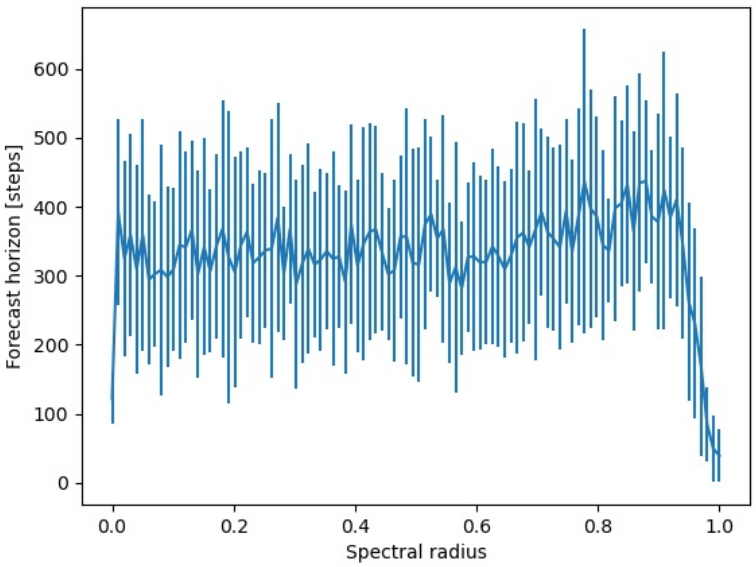
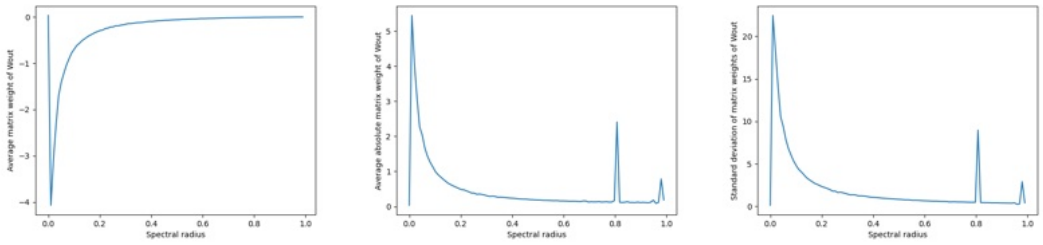


Figure B.29: FH vs. spectral radius for Lorenz system for specifically Lorenz tuned complex Minimal-RC



(a) Average weight (b) Average absolute weight (c) Average standard deviation

Figure B.30: Average matrix weights vs. spectral radius for Lorenz system for specifically Lorenz tuned complex Minimal-RC

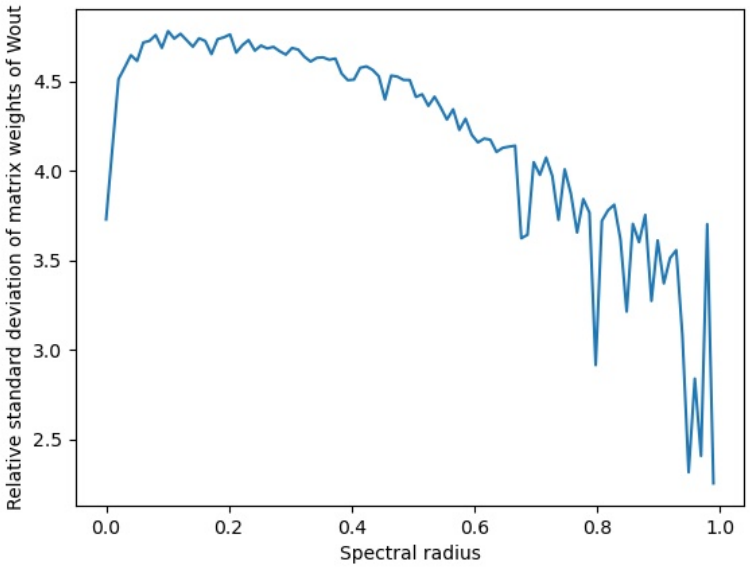


Figure B.31: Relative standard deviation vs. spectral radius for Lorenz system for specifically Lorenz tuned complex Minimal-RC

B.7 Fixed Points

B.7.1 Time Delay Embedde Halvorsen System

This example, involving the time-delay embedded x-coordinate of the Halvorsen system, reaches a fixed point extremely fast. Whether a fixed point occurs or not depends on the starting point. The same behavior is observed: the reservoir state values, as well as the values for $\boldsymbol{A}\vec{r}(t)$ and $\boldsymbol{W}_{in}\vec{u}(t)$, remain constant. In this case, the term $\boldsymbol{A}\vec{r}(t)$ is zero from the very beginning of the prediction, further confirming its role in the formation of fixed points.

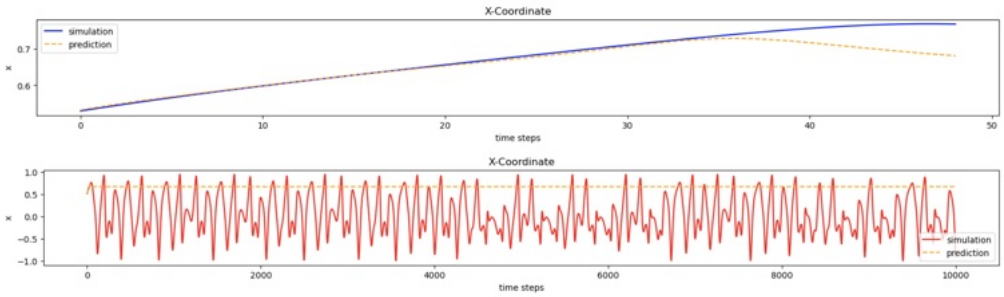
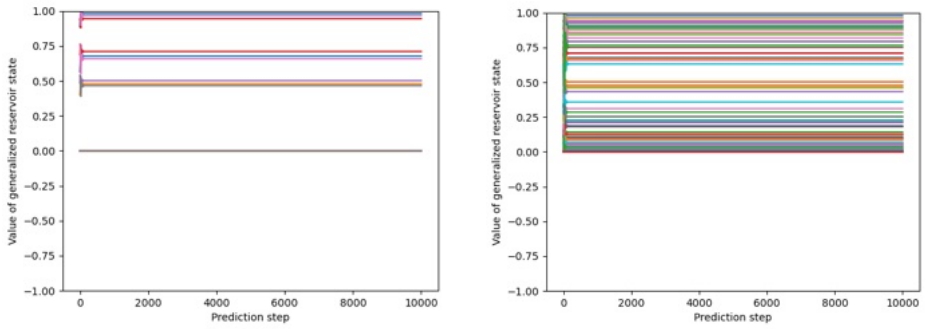


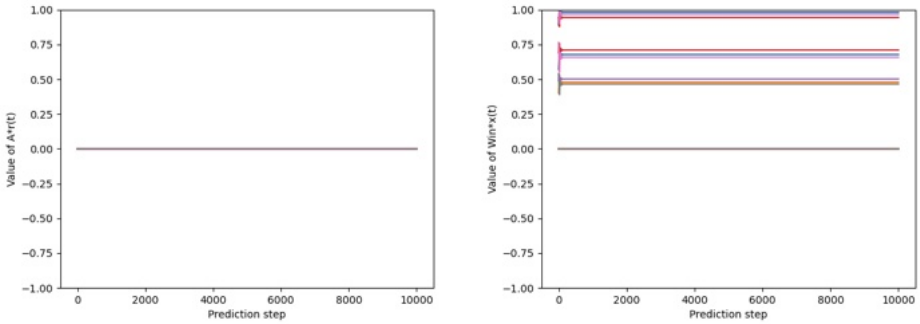
Figure B.32: X coordinate of prediction for time delay embedded Halvorsen system for 10.000 steps



(a) Linear reservoir state

(b) Generalized reservoir state

Figure B.33: All values of the reservoir state vector plotted for the prediction of time delay embedded Halvorsen system

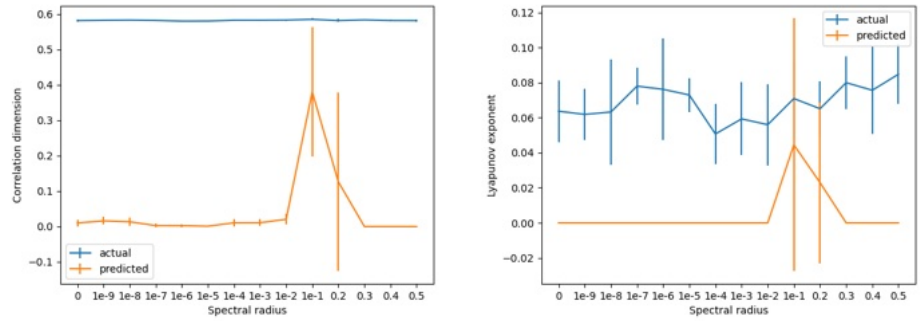


(a) $\boldsymbol{A}\vec{r}(t)$ (b) $\boldsymbol{W}_{in}\vec{u}(t)$

Figure B.34: All values of $\boldsymbol{A}\vec{r}(t)$ and $\boldsymbol{W}_{in}\vec{u}(t)$ plotted for the prediction for the time delay embedded Halvorsen system

B.7.2 Aizawa System: Lyapunov Exponent and Correlation Dimension

The differences in Lyapunov exponent and correlation dimension, as well as the forecast horizons for spectral radius values of $\rho = 0.1$ and $\rho = 0.2$, indicate that these are the best performing spectral radii for the Aizawa system. Some predictions did not yield a Lyapunov exponent—likely due to poor performance or unintended effects in the code—so the exponent was set to 0 in these cases.



(a) Correlation dimension (b) Lyapunov exponent

Figure B.35: Predicted and actual correlation dimension and lyapunov exponent for different spectral radii for the Aizawa system