Master's Thesis

Interpreting Reservoir Computing with PCA and Exploring Physics-Informed RC

Dennis Duncan

Ludwig-Maximilians-Universität München
Department: Physics
Supervisor: PD Dr. Christoph Räth

February 8, 2023

Masterarbeit

Interpretation von Reservoir Computern mittels PCA und Erforschung von physik-informierten RC-Ansätzen

Dennis Duncan

Ludwig-Maximilians-Universität München
Fakultät: Physik
Betreuer: PD Dr. Christoph Räth

8. Februar 2023

Contents

Chapter 1

Introduction

Whether it is the Earth's climate, the spread of pandemics, or financial systems, in many areas, the ability to forecast the behaviour of dynamical systems is immensely useful. When knowledge about the system's behaviour is available in form of governing equations, such as in many complex physical systems, one approach to forecasting is to use numerical methods to solve these equations. As these methods require knowledge of the underlying system, they are known as *knowledge-based models*. However, in many cases, the underlying equations are either not known or only imperfectly known, or their simulation is too computationally expensive, making accurate predictions difficult. In such cases, machine learning models can be a useful approach for forecasting time series, as they rely solely on data and do not require any knowledge of the system's governing equations. Additionally, a *knowledge-based model* can be combined with a data-driven machine learning model to archive even better prediction results. These combined approaches, known as *hybrid* or *physics-informed* machine learning models are considered a part of the interesting field of *theory-guided data science* [1]. One pure machine learning approach that has been shown to be highly effective for the task of predicting chaotic dynamical systems is *reservoir computing* (RC). Chaotic systems, such as a simple double pendulum or the complex Earth's weather system, represent one type of dynamical systems that are inherently difficult to predict. For chaotic systems, small variations in the past lead to largely different behaviours in the future. This property, commonly known as the "butterfly effect", prohibits accurate predictions for arbitrarily long time periods. A rough estimate of the maximal time duration for which an accurate prediction is possible, is given by the so-called *Lyapunov time*. Nevertheless, *reservoir computing* has repeatedly demonstrated the ability to make predictions that are highly accurate for short-time periods and are successful in replicating the long-term properties of the chaotic system [2]. Apart from good prediction capabilities for chaotic time series, the RC approach has other advantages, such as a conceptually simple architecture, fast training, and the ability to be realized by physical systems [3].

Reservoir computing, which originated from two independent works from the field of machine learning [4] and from the field of computational neuroscience [5], couples the time series to a high-dimensional, nonlinear, and fixed system called the *reservoir*. The exact implementation of the *reservoir* can take many forms, ranging from simple recurrent neural networks [4], to physical systems such as photonic systems [6], biologic systems [3], or even a bucket of water [7]. The main importance of the *reservoir* is its ability to embed its input nonlinearly in a high-dimensional space, so that the desired output can be learned by a simple learning algorithm, such as ridge regression, leaving the *reservoir* untouched during training. The interesting physical implementations of the *reservoir* can be considered as novel neuromorphic computing systems, which have the potential for efficiently processing large amounts of digital information with much lower power consumptions compared to conventional processors [8]. The RC approach is not limited to the task of time series prediction, but can also be applied to tasks like classification, anomaly detection, and system control [8]. In the case of time series prediction, the reservoir's inputs are given by the previous states of the time series, and the desired output corresponds to the time series' next step.

In this thesis, the simple recurrent neural network approach to RC, also known as *echo state networks* [4], is used. Being a software-based approach, it can be easily implemented on an ordinary computer. The objective of this thesis is twofold. Similar to many machine learning and artificial

intelligence systems, the inner workings of the *reservoir* in RC are not well understood and are often treated as a black box. A better understanding of these mechanisms can assist in selecting an effective *reservoir* for a given machine learning task. As a first part of this thesis, to investigate the path towards *explainable RC*, the *reservoir* and corresponding output-training are analyzed using a *principal component analysis* (PCA). The analysis presented in this thesis is similar to a previous study [9], but extends it substantially. Secondly, this thesis further investigates a *hybrid reservoir computing* approach, originally introduced by Pathak et al. [10]. This approach combines a *knowledge-based model* with RC, for which Pathak et al. have demonstrated significant improvements in predicting chaotic dynamical systems. This approach is further examined by using different types of knowledge-based models, revealing some of its strengths and limitations. As testing data, comparable simple three-dimensional chaotic systems are used, with a focus on the well-known Lorenz system, which was originally proposed as an approximate model for atmospheric convection [11].

The remainder of this thesis is organized as follows: In chapter 2, the basics of chaotic dynamical systems are introduced, which will provide the testing data for the rest of the thesis. Chapter 3 lays out the basic theory of PCA and ridge regression and highlights their strong connection. The usual recurrent neural network implementation of RC is presented in chapter 4, along with some simple extensions. Here, predictions of the Lorenz system are also presented, demonstrating the established methods and highlighting the influence of the individual hyperparameters. In chapter 5, the application of PCA on RC is presented and results are shown and discussed. Chapter 6 introduces the hybrid RC approach by Pathak et al. [10], and further investigates its strengths and limitations by applying it to nine different three-dimensional chaotic dynamical systems. Finally, the results of this thesis are summarized in chapter 7.

Chapter 2

Chaotic Dynamical Systems

This chapter introduces the basic concepts of chaotic dynamical system. The time series generated by numerically simulating these system will then be used as synthetic data for reservoir computing in the later chapters.

2.1 Definition

The area of chaotic dynamical systems is a subclass of nonlinear dynamics. Nonlinear dynamics, in turn, is a branch withing the broader field of dynamical systems, which encompasses all systems that evolve in time [12]. Furthermore, this thesis considers only deterministic dynamical systems, where the systems future is uniquely determined by its past according to certain governing equations. These systems are distinct from stochastic systems, which involve random processes. In addition, the dynamical system can evolve either continuously in time or in discrete time steps, where the state of the system at a given moment, whether it be time t or step i, is represented by a u_{dim}-dimensional vector $\boldsymbol{u}(t)$ or $\boldsymbol{u}[i]$. Once an initial state $\boldsymbol{u}_0$ is specified, the future state of the system evolves in time according to deterministic equations. For discrete dynamical systems, the state evolves according to a map $M(\cdot)$:

$$\boldsymbol{u}[i+1] = M(\boldsymbol{u}[i]) \tag{2.1}$$

The evolution of the system $\boldsymbol{u}[i]$ can be directly generated by simply iterating the map Eq. (2.1) multiple times. For continuous-time dynamical systems, the state's time evolution is governed by a first-order differential equation in time, defined by the flow $F(\cdot)$:

$$\dot{\boldsymbol{u}}(t) = F(\boldsymbol{u}(t)) \tag{2.2}$$

In contrast to the discrete time case, the true evolution of the system in the continuous-time case, represented by $\boldsymbol{u}(t)$, can only be determined by analytically solving the first-order differential equation in time as defined by the flow Eq. (2.2). Given that most interesting continuous-time dynamical systems are described by flow equations that are not analytically solvable, the common approach is to find an approximate solution by solving Eq. (2.2) numerically. In the upcoming section, the process of numerically solving flow equations using the forth-order Runge-Kutta method (RK4) is described.

When the functions $M(\boldsymbol{u})$ and $F(\boldsymbol{u})$, which define the governing equations, are nonlinear in the system's variables, u_i, one enters the field of nonlinear dynamics. These systems often result in rich and complex dynamics, and in some cases, they exhibit *chaotic* behavior. Chaotic dynamics are characterized by being non-periodic, sensitive with respect to initial conditions, and bounded [12]. A system is considered non-periodic if each state in its infinite time evolution is visited only once. It is referred to as sensitive to initial conditions if two initially highly similar states, $\boldsymbol{u}_0 \approx \tilde{\boldsymbol{u}}_0$, diverge significantly after being evolved for some time. It is interesting to note that while chaos can occur

already in one-dimensional discrete deterministic dynamical systems, continuous systems require at least three dimensions to be able to generate chaotic dynamics [12]. Useful measures to analyze the deviation of initially nearby states in a dynamical system are given by its *Lyapunov exponents*, described in Sec. 2.3, which serve as an indicator of whether a given system behaves chaotically or not. Often, the trajectories of chaotic systems are drawn to small subspaces of the whole state space, denoted as *strange attractors*, which are fractal objects with interesting shapes [12].

2.2 Numerical simulation using RK4

In this section, the numerical simulation of a continuous-time dynamical system, Eq. (2.2), using the forth-order Runge-Kutta method (RK4) [13, 14] is described. This method discretizes time into small time steps of size Δt which effectively approximates the flow Eq. (2.2) by a map $M_{\mathrm{rk4}}(\cdot)$:

$$\boldsymbol{u}[i+1] = M_{\mathrm{rk4}}(\boldsymbol{u}[i]) \tag{2.3}$$

Each discrete time step $i = 0, 1, \ldots$ corresponds to the real time $t_0 + i\Delta t$. The RK4 iterator $M_{\mathrm{rk4}}(\cdot)$ is defined as:

$$M_{\mathrm{rk4}}(\boldsymbol{u}[i]) = \boldsymbol{u}[i] + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.4}$$

The terms k_j in Eq. (2.4) are given by:

$$k_1 = F(\boldsymbol{u}[i])\Delta t \tag{2.5}$$
$$k_2 = F(\boldsymbol{u}[i] + k_1/2)\Delta t \tag{2.6}$$
$$k_3 = F(\boldsymbol{u}[i] + k_2/2)\Delta t \tag{2.7}$$
$$k_4 = F(\boldsymbol{u}[i] + k_3)\Delta t \tag{2.8}$$

In this thesis, all time series that are used as data in reservoir computing are generated using the RK4 method.

2.3 Maximal Lyapunov exponents

The number of Lyapunov exponents in a system is equivalent to the dimensionality of its state space, u_{dim}. Of these exponents, the maximal Lyapunov exponent (MLE) is particularly significant, as a positive MLE value, $\lambda_{\mathrm{max}} > 0$, indicates that the system is chaotic. The MLE is a global quantity that measures the overall behaviour of the dynamical system's attractor, but its definition is rooted in a local quantity, known as the local MLE $\lambda_{\mathrm{local}}(\boldsymbol{u}_0)$, which measures the divergence of nearby trajectories at a specific point, $\boldsymbol{u}_0$, on the attractor. To calculate the local MLE, one can take an initial state, $\boldsymbol{u}_0$, and a slightly perturbed version of it, $\tilde{\boldsymbol{u}}_0 = \boldsymbol{u}_0 + \boldsymbol{\delta}_0$ with $\|\boldsymbol{\delta}_0\| = \delta_0 \approx 0$, evolve both for a short period of time, and measure the exponential growth rate of the difference between the final states: After evolving both states, $\boldsymbol{u}_0$ and $\tilde{\boldsymbol{u}}_0$, for a small number of n steps, each corresponding to a time step Δt, the length of the difference of $\boldsymbol{\delta}_n = \tilde{\boldsymbol{u}}_n - \boldsymbol{u}_n$ can be written as:

$$\delta_n = \delta_0 e^{\lambda_{\mathrm{local}} n \Delta t} \tag{2.9}$$

The global MLE, λ_{max}, is determined by averaging the local MLE, $\lambda_{\mathrm{local}} = \log(\delta_n/\delta_0)/(n\Delta t)$, for all initial conditions $\boldsymbol{u}_0$ that lie on the attractor. The MLE for chaotic systems is positive, $\lambda_{\mathrm{max}} > 0$, which explains its sensitivity to small variations in the initial conditions, as the resulting trajectories diverge exponentially. This property makes chaotic systems, such as the Earth's weather system, unpredictable over long periods of time. One can define the Lyapunov time, $\tau = 1/\lambda_{\mathrm{max}}$, as the

characteristic timescale of a chaotic system, which represents the time at which initial perturbations have increased by a factor e. Systems with larger Lyapunov times typically allow for more accurate predictions over longer periods.

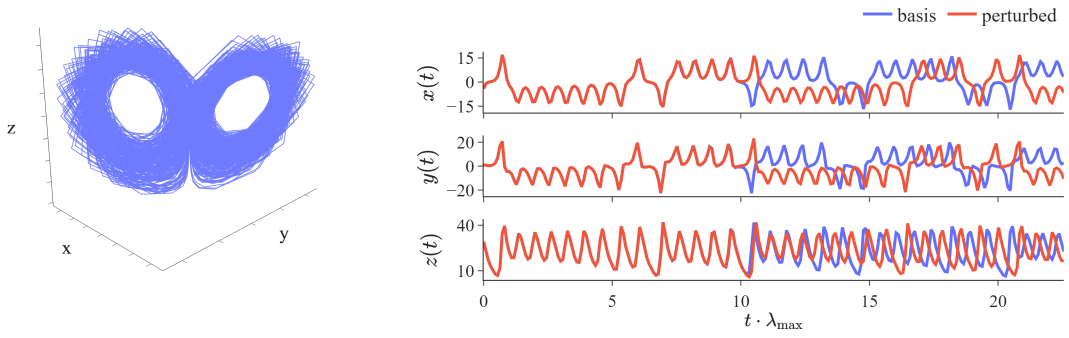
2.4 Lorenz system

The Lorenz system, first introduced as a simplified model for atmospheric convection [11], is a well-known example of a three-dimensional continuous-time system that displays chaotic dynamics in certain parameter ranges. The system's state, represented by $\boldsymbol{u}(t) = [x(t), y(t), z(t)]^T$, evolves according to the flow Eq. (2.10):

$$\begin{aligned} \dot{x}(t) &= \sigma(y - x) \\ \dot{y}(t) &= x(\rho - z) - y \\ \dot{z}(t) &= xy - \beta z \end{aligned} \tag{2.10}$$

For the typical parameter values, $\rho = 28$, $\sigma = 10$, $\beta = 8/3$, the system evolves chaotically, resulting in the iconic butterfly shape shown in Fig. 2.1a. The chaotic nature of the Lorenz system can be quantified by its positive MLE, which is given as $\lambda_{\max} \approx 0.9059$ [12]. The system's sensitivity to small variations in the initial conditions is illustrates in Fig. 2.1b, which shows the time series for the three dimensions, $x(t)$, $y(t)$, and $z(t)$, for two initial states, one of which is perturbed by the vector $\boldsymbol{\delta}_0 = [10^{-4}, 0, 0]$. After approximately 10 Lyapunov times, the differences between the two time series become noticeable, and the subsequent evolutions are completely desynchronized.

In this thesis, the Lorenz system serves as the primary benchmark for evaluating the performance of reservoir computing in predicting time series. In chapter 6, the performance of hybrid reservoir computing will also be evaluated on eight additional three-dimensional chaotic systems, whose flow equations are listed in appendix A. For the illustration in Fig. 2.1b, and throughout this thesis, the Lorenz system is simulated using the RK4 method with a time step of $\Delta t = 0.1$, and an initial condition of $\boldsymbol{u}(0) = [0, -0.01, 9]^T$.



(a) Lorenz attractor

(b) Time series of initially nearby states

Figure 2.1: Evolution of the Lorenz system. (a) 3D-Trajectory of 4000 time steps which illustrates the Lorenz attractor. (b) First 250 time steps of the Lorenz time series of two initially nearby states perturbed by $\boldsymbol{\delta}_0 = [10^{-4}, 0, 0]$.

Chapter 3

Principal Component Analysis and Ridge Regression

The method of *ridge regression* (RR) plays an integral part in reservoir computing, as will be outlined in chapter 4, and is closely related to the concept of *principal components* (PCs). In chapter 5, *principal component analysis* (PCA) is utilized to gain insights into the inner workings of reservoir computing. This chapter establishes the theoretical foundations for PCA and RR, and highlights how ridge regression can be understood through principal components.

3.1 Principal component analysis

PCA is a simple unsupervised learning technique that allows for the analysis of a dataset, represented by $\mathbf{K} = [\boldsymbol{k}[1], \boldsymbol{k}[2], \ldots, \boldsymbol{k}[N]] \in \mathbb{R}^{k_{\text{dim}} \times N}$, containing N observations, $\boldsymbol{k}[j]$. The principal components of the dataset are orthogonal directions, $\boldsymbol{p}_i$, within the data's space, $\mathbb{R}^{k_{\text{dim}}}$, that are ordered by the data's variance along these directions. There are an equal number of principal components as there are mutually orthogonal directions in the data's space, which is given by k_{dim}.

The first PC, $\boldsymbol{p}_1$, points in the direction of the greatest variance within the centered dataset $\mathbf{K}_\text{c}$. The centered dataset, $\mathbf{K}_\text{c}$, can be obtained by subtracting the mean of the data from each observation: $\boldsymbol{k}_\text{c}[j] = \boldsymbol{k}[j] - \overline{\boldsymbol{k}}$. The mean of the data, $\overline{\boldsymbol{k}}$, is given by:

$$\overline{\boldsymbol{k}} = \frac{1}{N} \sum_{j=1}^{N} \boldsymbol{k}[j] \tag{3.1}$$

Mathematically, the first PC, $\boldsymbol{p}_1$, is given by the unit vector which maximizes the variance of the projections $\boldsymbol{p}_1 \cdot \boldsymbol{k}_\text{c}[j]$ over all samples j:

$$\text{Var}(\boldsymbol{p}_1^T \mathbf{K}_\text{c}) = \frac{1}{N} \sum_{j=1}^{N} (\boldsymbol{p}_1 \cdot \boldsymbol{k}_\text{c}[j])^2 \tag{3.2}$$

The second PC, $\boldsymbol{p}_2$, is also determined by maximizing the variance of the centered dataset projected onto it, $\text{Var}(\boldsymbol{p}_2^T \mathbf{K}_\text{c})$, but under the additional constraint of being orthogonal to the first PC, $\boldsymbol{p}_1$. This process is repeated for the subsequent PCs, such that the kth PC, $\boldsymbol{p}_k$, is defined by maximizing $\text{Var}(\boldsymbol{p}_k^T \mathbf{K}_\text{c})$, while being orthogonal to all previous PCs, $\boldsymbol{p}_i$, where $i < k$. The variances, $\phi_k = \text{Var}(\boldsymbol{p}_k^T \mathbf{K}_\text{c})$, of the centered data projected onto the PCs are known as the *explained variances* (EVs).

The problem of finding the PCs by successively maximizing the remaining variance in the data, as described above, is equivalent to finding the eigenvectors of the sample covariance matrix $\mathbf{S} =$

$\mathbf{K}_\mathrm{c}\mathbf{K}_\mathrm{c}^T/N$ [15]: Each eigenvector of $\mathbf{S}$ represents a principal component, and the corresponding eigenvalue equals the explained variance of that component. Therefore, the principal components, $\boldsymbol{p}_i$, and explained variances, ϕ_i, can be efficiently retrieved from the unitary matrix, $\mathbf{P}$, and diagonal matrix, $\boldsymbol{\Phi}$, which constitute the eigendecomposition of the square covariance matrix, $\mathbf{S}$:

$$\mathbf{S} = \mathbf{P}^T\boldsymbol{\Phi}\mathbf{P} \tag{3.3}$$

When the rows of $\mathbf{P}$ and $\boldsymbol{\Phi}$ are ordered by decreasing values in the diagonal elements of $\boldsymbol{\Phi}$, $\phi_1 \geq \phi_2 \geq \ldots \geq \phi_{k_\mathrm{dim}}$, the ith row of $\mathbf{P}$ represents the ith principal component, $\boldsymbol{p}_i$, and the ith diagonal element in $\boldsymbol{\Phi}$ represents the corresponding ith explained variance, ϕ_i. The unitary matrix $\mathbf{P}$ will be referred to as the *principal component matrix*, and the diagonal matrix $\boldsymbol{\Phi}$ as the *explained variance matrix*.

After identifying the principal components, the original data $\boldsymbol{k}$ can be transformed onto the coordinate system defined by the PCs:

$$\boldsymbol{k}_\mathrm{pc} = \mathbf{P}(\boldsymbol{k} - \overline{\boldsymbol{k}}) \tag{3.4}$$

Transforming the data onto the principal components results in the new variables, $\boldsymbol{k}_\mathrm{pc}$, being linearly uncorrelated. This can be observed from the covariance matrix of the transformed data, $\mathbf{K}_\mathrm{pc}\mathbf{K}_\mathrm{pc}^T/N$, which is simply given by the diagonal explained variance matrix, $\boldsymbol{\Phi}$:

$$\mathbf{K}_\mathrm{pc}\mathbf{K}_\mathrm{pc}^T/N = \mathbf{P}\mathbf{S}\mathbf{P}^T = \boldsymbol{\Phi} \tag{3.5}$$

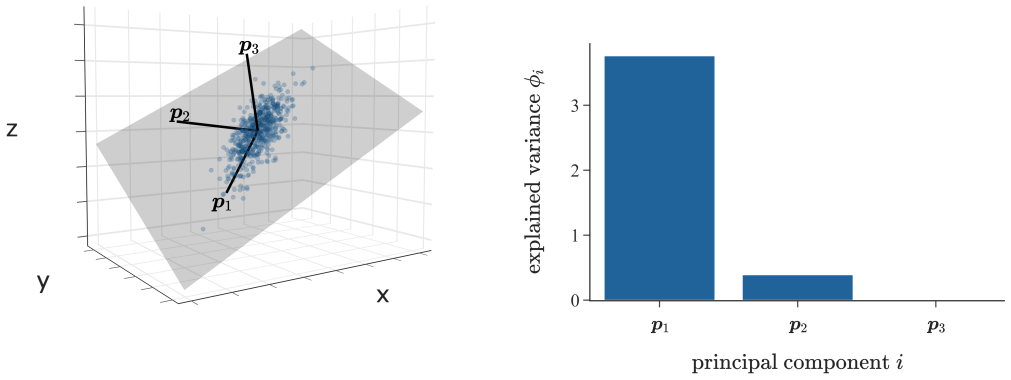
Multicollinearity When the centered data, $\mathbf{K}_\mathrm{c}$, has some variables that are perfectly correlated, meaning that some of the k_dim dimensions are linear functions of each other, it is said to exhibit *multicollinearity*. In such cases, the data lives in a lower-dimensional subspace, whose dimensionality is equal to the rank of the centered data matrix, $\mathrm{rank}(\mathbf{K}_\mathrm{c})$. PCA can be used to detect multicollinearity, as all explained variances, ϕ_i, corresponding to principal components $i > \mathrm{rank}(\mathbf{K}_\mathrm{c})$ vanish. This is shown in Figure 3.1, where $k_\mathrm{dim} = 3$, but the data actually lies on a two-dimensional plane, meaning one variable is a linear function of the other two, hence the explained variance, ϕ_3, of the third principal component is zero. In cases where some dimensions of the dataset are only approximately linear functions of each other, the dataset is said to exhibit approximate multicollinearity. In such scenarios, the explained variances, ϕ_i, corresponding to the higher principal components are not strictly zero, but become very small. Multicollinearity within the data poses problems in linear regression tasks, as will be discussed in Sec. 3.3.

Benefits of using PCA It was demonstrated how the distribution of explained variances, ϕ_i, provides insights into the extent to which the data lives in independent directions, which can be used to detect highly correlated variables. Often, a small number of the first principal components can explain the majority of the variance within the data. By projecting the data onto this subset of principal components, PCA is frequently used as a dimensionality reduction technique. Additionally, the principal components provide a natural coordinate system for the data, similar to how the principal axes of a physical object provide a natural coordinate system. Transforming the data to this uncorrelated principal component space can increase its interpretability.

PCA and SVD The principal components of a dataset $\mathbf{K}$ can also be obtained through a singular value decomposition (SVD) applied to the transposed centered dataset, $\mathbf{K}_\mathrm{c}^T$. The SVD provides an important link when connecting PCA to ridge regression, which will be discussed in Section 3.3. SVD decomposes $\mathbf{K}_\mathrm{c}^T$ into three matrices, $\mathbf{U}$, $\mathbf{D}$, and $\mathbf{V}^T$:

$$\mathbf{K}_\mathrm{c}^T = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{3.6}$$

$\mathbf{U} \in \mathbb{R}^{N\times k_\mathrm{dim}}$ is a semi-orthogonal matrix, meaning $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, but $\mathbf{U}\mathbf{U}^T \neq \mathbf{I}$. Its columns are called *left-singular vectors*. The matrix $\mathbf{V} \in \mathbb{R}^{k_\mathrm{dim}\times k_\mathrm{dim}}$ is an orthogonal matrix, meaning



(a) (b)

Figure 3.1: (a) Scatter plot of three-dimensional data points, $\boldsymbol{k}[i]$, which all lie on the gray plane. The principal components $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$ are displayed. The first two principal components, $\boldsymbol{p}_1, \boldsymbol{p}_2$, both lie in the plane, whereas $\boldsymbol{p}_3$ is perpendicular to it. (b) The explained variances, ϕ_i, corresponding to the principal components. It can be observed that the explained variance, ϕ_3, corresponding to the third PC is 0, due to the multicollinearity in the data. By projecting the data only on the first two principal components, the dimensionality of the data could be reduced without loosing any information.

$\mathbf{V}^T\mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}$, whose columns are called *right-singular vectors*. $\mathbf{D} \in \mathbb{R}^{k_{\text{dim}} \times k_{\text{dim}}}$ is a diagonal matrix with the positive *singular values* $d_1 \geq d_2 \geq \ldots \geq d_{k_{\text{dim}}}$ as its diagonal entries.

By inserting the SVD of $\mathbf{K}_{\text{c}}^T$, represented by Eq. (3.6), into the definition of the covariance matrix, $\mathbf{S} = \mathbf{K}_{\text{c}}\mathbf{K}_{\text{c}}^T/N$, and comparing it with Eq. (3.3), the principal component matrix, $\mathbf{P}$, and explained variances, $\mathbf{\Phi}$, can be linked back to $\mathbf{V}$ and $\mathbf{D}$:

$$\mathbf{P}^T\mathbf{\Phi}\mathbf{P} \overset{\text{PCA}}{=} \mathbf{K}_{\text{c}}\mathbf{K}_{\text{c}}^T/N \overset{\text{SVD}}{=} \mathbf{V}\mathbf{D}^2\mathbf{V}^T/N \tag{3.7}$$

On the right side of Eq. (3.7) the semi-orthogonality of $\mathbf{U}$ was used. By comparing the left and right side of Eq. (3.7), the principal component matrix, $\mathbf{P}$, and explained variance matrix, $\mathbf{\Phi}$, can be linked to the right-singular vectors, $\mathbf{V}$, and the singular values, $\mathbf{D}$, respectively.

$$\mathbf{\Phi} = \mathbf{D}^2/N \tag{3.8}$$

$$\mathbf{P} = \mathbf{V}^T \tag{3.9}$$

3.2 Ridge regression

Ridge regression, also known as *Tikhonov regularized regression* [16], is a simple method of supervised learning that is used to establish a linear relationship between an input variable (also known as the explanatory variable) represented by $\boldsymbol{k} \in \mathbb{R}^{k_{\text{dim}}}$ and an output variable (also known as the dependent variable) represented by $\boldsymbol{y} \in \mathbb{R}^{y_{\text{dim}}}$. This relationship can be either represented by a pure linear model, M1 Eq.(3.10), or a linear model that includes an intercept vector, M2 Eq.(3.11):

$$\text{M1: } \hat{\boldsymbol{y}} = \mathbf{W}\boldsymbol{k} \tag{3.10}$$

$$\text{M2: } \hat{\boldsymbol{y}} = \mathbf{W}\boldsymbol{k} + \boldsymbol{w} \tag{3.11}$$

Ridge regression finds the optimal parameter matrix, $\mathbf{W} \in \mathbb{R}^{y_{\text{dim}} \times k_{\text{dim}}}$, and intercept, $\boldsymbol{w} \in \mathbb{R}^{y_{\text{dim}}}$, if included, that minimize a certain kind of error between the model prediction, $\hat{\boldsymbol{y}}[i]$, and the true output, $\boldsymbol{y}[i]$, for all training observations, $\{\boldsymbol{k}[i], \boldsymbol{y}[i]\}$, where $i = 1, \ldots, N$. Compared to ordinary least squares (OLS) regression, where the error is given as the residual sum of squares, RSS $= \sum_i \|\boldsymbol{y}[i] - \hat{\boldsymbol{y}}[i]\|^2$, RR additionally penalizes large parameter values by adding a regularization term to RSS. The additional regularization becomes useful when the explanatory variables, $\boldsymbol{k}$, are highly correlated. In such cases of multicollinearity, the OLS solutions results in very large parameter values leading to overfitting, or even becomes undetermined in the extreme case of true mutlicollinearity.

The implementation of RR can be specified by selecting either the purely linear model, M1, or the linear model including an intercept, M2. When M2 is chosen, there is an additional option to either only regularize values in the parameter matrix, $\mathbf{W}$, or to additionally penalize large values in the intercept, $\boldsymbol{w}$. The resulting three types of RR, denoted as type-a, type-b and type-c, are defined by the loss functions L_{a}, L_{b}, and L_{c}, respectively:

$$L_{\text{a}}(\mathbf{W}) = \sum_{i=1}^{N} \|\boldsymbol{y}[i] - \mathbf{W}\boldsymbol{k}[i]\|^2 + \beta \,\text{Tr}(\mathbf{W}^T\mathbf{W}) \tag{3.12}$$

$$L_{\text{b}}(\mathbf{W}, \boldsymbol{w}) = \sum_{i=1}^{N} \|\boldsymbol{y}[i] - \mathbf{W}\boldsymbol{k}[i] - \boldsymbol{w}\|^2 + \beta \,\text{Tr}(\mathbf{W}^T\mathbf{W}) \tag{3.13}$$

$$L_{\text{c}}(\mathbf{W}, \boldsymbol{w}) = \sum_{i=1}^{N} \|\boldsymbol{y}[i] - \mathbf{W}\boldsymbol{k}[i] - \boldsymbol{w}\|^2 + \beta \left[\text{Tr}(\mathbf{W}^T\mathbf{W}) + \boldsymbol{w}^T\boldsymbol{w}\right] \tag{3.14}$$

The loss functions $L_{\text{a/b/c}}$ in ridge regression consist of two parts: the left sum represents the residual sum of squares, RSS, and the right term starting with β represents the regularization term. The positive *regularization parameter*, $\beta > 0$, controls the strength of the penalty for large parameter values. If $\beta = 0$, ridge regression reduces to OLS regression. The minimizing parameters for the three types of RR loss functions, $L_{\text{a/b/c}}$, are denoted as $\mathbf{W}_{\text{a}}$, $(\mathbf{W}_{\text{b}}, \boldsymbol{w}_{\text{b}})$, and $(\mathbf{W}_{\text{c}}, \boldsymbol{w}_{\text{c}})$, respectively. The solutions for these minimization problems can be calculated analytically due to the quadratic nature of the loss functions. To do this, it is useful to work with the data matrices $\mathbf{K} = [\boldsymbol{k}[1], \ldots, \boldsymbol{k}[N]] \in \mathbb{R}^{k_{\text{dim}} \times N}$ and $\mathbf{Y} = [\boldsymbol{y}[1], \ldots, \boldsymbol{y}[N]] \in \mathbb{R}^{y_{\text{dim}} \times N}$, which represent the input and output datasets, respectively.

Type-a: no intercept To find the minimizing parameter matrix, $\mathbf{W}_{\text{a}} = \arg\min_{\mathbf{W}}(L_{\text{a}})$, it is useful to rewrite Eq. (3.12) in matrix form:

$$L_{\text{a}}(\mathbf{W}) = (\mathbf{Y} - \mathbf{W}\mathbf{K})^T(\mathbf{Y} - \mathbf{W}\mathbf{K}) + \beta \,\text{Tr}(\mathbf{W}^T\mathbf{W}) \tag{3.15}$$

The solution parameters $\mathbf{W}_{\text{a}}$ can be determined by differentiating Eq. (3.15) with respect to $\mathbf{W}$, and setting the result to zero. Solving for $\mathbf{W}_{\text{a}} = \mathbf{W}$ yields:

$$\mathbf{W}_{\text{a}} = \mathbf{Y}\mathbf{K}^T \left(\mathbf{K}\mathbf{K}^T + \beta\mathbf{I}\right)^{-1} \tag{3.16}$$

Here, $\mathbf{I}$ represents the $k_{\text{dim}} \times k_{\text{dim}}$ identity matrix.

Type-b: unregularized intercept The minimizing parameters, $(\mathbf{W}_{\text{b}}, \boldsymbol{w}_{\text{b}}) = \arg\min_{\mathbf{W},\boldsymbol{w}}(L_{\text{b}})$, for type-$b$ RR can be found by first minimizing the loss function, L_{b}, with respect to the intercept, $\boldsymbol{w}$, and then by minimizing with respect to the parameter matrix $\mathbf{W}$. The partly minimizing intercept, represented by $\boldsymbol{w}^*$, can be obtained by setting the gradient of L_{b} with respect to $\boldsymbol{w}$ to zero and solving for $\boldsymbol{w}^* = \boldsymbol{w}$:

$$\nabla_{\boldsymbol{w}} L_{\text{b}}(\mathbf{W}, \boldsymbol{w}) = -2 \sum_{i=1}^{N} (\boldsymbol{y}[i] - \boldsymbol{w} - \mathbf{W}\boldsymbol{k}[i]) \stackrel{!}{=} 0 \tag{3.17}$$

After solving for $\boldsymbol{w}$, the partly minimizing intercept $\boldsymbol{w}^*$ is given by:

$$\boldsymbol{w}^* = \overline{\boldsymbol{y}} - \mathbf{W}\overline{\boldsymbol{k}} \tag{3.18}$$

Here, $\overline{\boldsymbol{k}}$ and $\overline{\boldsymbol{y}}$ represent the input and output data means, $\overline{\boldsymbol{k}} = \frac{1}{N}\sum_i \boldsymbol{k}[i]$ and $\overline{\boldsymbol{y}} = \frac{1}{N}\sum_i \boldsymbol{y}[i]$, respectively. By inserting the minimizing intercept from Eq. (3.18) into the loss function, L_b in Eq. (3.13), the dependency on the intercept can be eliminated:

$$L_\mathrm{b}(\mathbf{W}) = \sum_{i=1}^{N} \|\boldsymbol{y}_\mathrm{c}[i] - \mathbf{W}\boldsymbol{k}_\mathrm{c}\|^2 + \beta\,\mathrm{Tr}(\mathbf{W}^T\mathbf{W}) \tag{3.19}$$

Here, $\boldsymbol{k}_\mathrm{c} = \boldsymbol{k} - \overline{\boldsymbol{k}}$ and $\boldsymbol{y}_\mathrm{c} = \boldsymbol{y} - \overline{\boldsymbol{y}}$ represent the centered input and output variables, respectively. Comparing L_b in Eq. (3.19) with the definition of the type-a loss function, L_a in Eq. (3.12), one can observe that both have the same form, where the former is acting on the centered data $\{\boldsymbol{k}_\mathrm{c}[i], \boldsymbol{y}_\mathrm{c}[i]\}$, and the latter on the original data $\{\boldsymbol{k}[i], \boldsymbol{y}[i]\}$. Replacing the uncentered data matrices, $\mathbf{K}$ and $\mathbf{Y}$, in the type-a solution, Eq. (3.16), with the centered data matrices, $\mathbf{K}_\mathrm{c}$ and $\mathbf{Y}_\mathrm{c}$, the type-b solution of $\mathbf{W}_\mathrm{b}$ is given by:

$$\mathbf{W}_\mathrm{b} = \mathbf{Y}_\mathrm{c}\mathbf{K}_\mathrm{c}^T\left(\mathbf{K}_\mathrm{c}\mathbf{K}_\mathrm{c}^T + \beta\mathbf{I}\right)^{-1} \tag{3.20}$$

By noting that $\mathbf{Y}\mathbf{K}_\mathrm{c}^T = \mathbf{Y}_\mathrm{c}\mathbf{K}_\mathrm{c}^T$, which holds if $\mathbf{K}_\mathrm{c}$ is centered, the full type-b solution, $(\mathbf{W}_\mathrm{b}, \boldsymbol{w}_\mathrm{b})$, is given by:

$$\mathbf{W}_\mathrm{b} = \mathbf{Y}\mathbf{K}_\mathrm{c}^T\left(\mathbf{K}_\mathrm{c}\mathbf{K}_\mathrm{c}^T + \beta\mathbf{I}\right)^{-1} \tag{3.21}$$

$$\boldsymbol{w}_\mathrm{b} = \overline{\boldsymbol{y}} - \mathbf{W}_\mathrm{b}\overline{\boldsymbol{k}} \tag{3.22}$$

Type-c: regularized intercept The type-c RR solution parameters, $(\mathbf{W}_\mathrm{c}, \boldsymbol{w}_\mathrm{c})$, can be obtained by a simple redefinition of the input variables, $\boldsymbol{k}$. Specifically, by adding one dimension to $\boldsymbol{k}$ with a constant value of one, $\boldsymbol{k} \to \tilde{\boldsymbol{k}} = [k_1, k_2, \ldots, k_{k_\mathrm{dim}}, 1]^T$, the model M2 that includes the intercept, Eq. (3.11), can be effectively rewritten as a model M1 without an intercept, Eq. (3.10):

$$\hat{\boldsymbol{y}} = \mathbf{W}\boldsymbol{k} + \boldsymbol{w} = \tilde{\mathbf{W}}\tilde{\boldsymbol{k}} \tag{3.23}$$

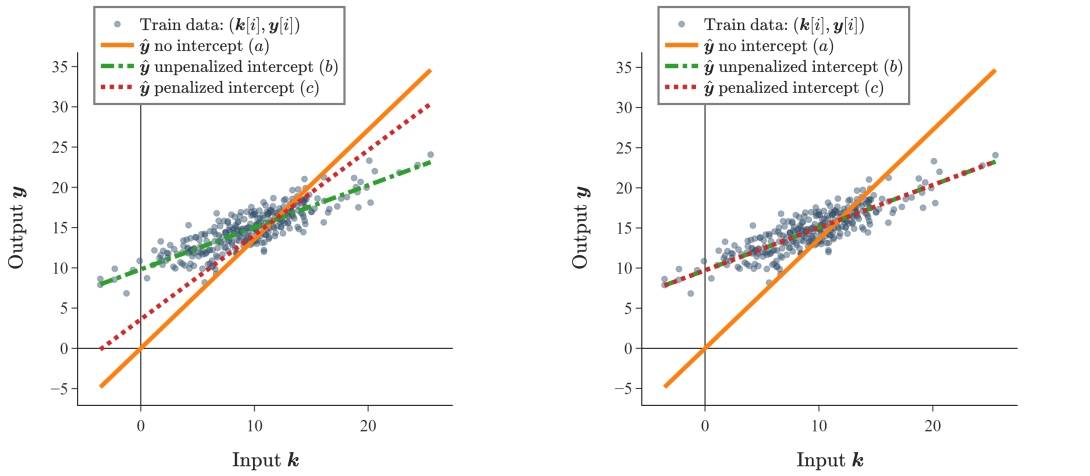
The original intercept, $\boldsymbol{w}$, can be recovered by taking the last column of the matrix $\tilde{\mathbf{W}}$. The original parameter matrix, $\mathbf{W}$, is then formed by collecting all the previous columns of $\tilde{\mathbf{W}}$. Using this transformation, the type-c loss function, L_c in Eq.(3.14), can be again cast into the familiar form of the type-a loss function, L_a in Eq.(3.12), by replacing $\mathbf{W}$ with $\tilde{\mathbf{W}}$ and $\boldsymbol{k}$ with $\tilde{\boldsymbol{k}}$:

$$L_\mathrm{c}(\mathbf{W}) = \sum_{i=1}^{N} \|\boldsymbol{y}[i] - \tilde{\mathbf{W}}\tilde{\boldsymbol{k}}[i]\|^2 + \beta\,\mathrm{Tr}(\tilde{\mathbf{W}}^T\tilde{\mathbf{W}}) \tag{3.24}$$

By using the form of the type-a solution, Eq.(3.16), once more, the type-c solution, $(\mathbf{W}_\mathrm{c}, \boldsymbol{w}_\mathrm{c})$, can be expressed as:

$$\tilde{\mathbf{W}}_\mathrm{c} = \mathbf{Y}\tilde{\mathbf{K}}^T\left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + \beta\mathbf{I}\right)^{-1} \tag{3.25}$$

$$[\mathbf{W}_\mathrm{c}, \boldsymbol{w}_\mathrm{c}] = \tilde{\mathbf{W}}_\mathrm{c} \tag{3.26}$$



(a) Larger regularization parameter, $\beta = 10^2$ (b) Smaller regularization parameter, $\beta = 10^{-1}$

Figure 3.2: The three ridge regression types, *no intercept* (type-a), *unpenalized intercept*, (type-b) and *penalized intercept* (type-c), for one-dimensional input and output data.

Comparison The three types of ridge regression, a, b, and c, introduced above, exhibit different fitting capabilities. These types are compared in Fig. 3.2 for simple one-dimensional input and output data, for both a large and small regularization parameter, β. As shown in Fig. 3.2, type-b ridge regression can produce the best fit, followed by type-c and type-a ridge regression. The superiority of type-b and type-c over type-a is understandable as the inclusion of an intercept $\boldsymbol{w}$ allows the fitting plane/line to not be fixed to the origin, which is necessary for a good fit when the data is uncentered. However, if the input and output data are centered, i.e. $\overline{\boldsymbol{k}} = 0$ and $\overline{\boldsymbol{y}} = 0$, all three types become equivalent. Additionally, as mentioned in [15], regularizing the intercept in type-c does not offer any advantages over type-b, since it results in an unwanted dependence of the fitting procedure on the origins chosen for the input and output data.

In practice, as shown in Fig. 3.2 and also demonstrated for reservoir computing in Sec. 4.6, the differences between type-b and type-c are usually insignificant as long as the regularization parameter is not too large and the data is not highly uncentered. The differences between the three RR types applied to reservoir computing in the case of strongly uncentered output data, $\boldsymbol{y}$, is further discussed in appendix B where the advantages of type-b become more apparent.

Invariance in type-b The main advantage of type-b ridge regression, which will be utilized in this thesis, is its invariance under a transformation of the input data of the following form:

$$\boldsymbol{k} \to \boldsymbol{k}_\mathrm{m} = \mathbf{M}\boldsymbol{k} + \boldsymbol{m} \tag{3.27}$$

Here, $\mathbf{M} \in \mathbb{R}^{k_\mathrm{dim} \times k_\mathrm{dim}}$ is a unitary matrix, and $\boldsymbol{m} \in \mathbb{R}^{k_\mathrm{dim}}$ is an arbitrary vector that shifts the input data. As the principal component transformation, given by Eq. (3.4), corresponds to a transformation of such kind, with $\mathbf{M} = \mathbf{P}$ and $\boldsymbol{m} = -\mathbf{P}\overline{\boldsymbol{k}}$, this property allows for an interpretation of type-b ridge regression in terms of the principal components of the input data, as will be discussed in the following section.

This invariance can be understood by calculating the type-b solutions, $(\mathbf{W}_\mathrm{m}, \boldsymbol{w}_\mathrm{m})$, for the transformed input, $\boldsymbol{k}_\mathrm{m}$, according to Eq. (3.21) and Eq. (3.22), and relating them back to the original solutions, $(\mathbf{W}_\mathrm{b}, \boldsymbol{w}_\mathrm{b})$ as:

$$\mathbf{W}_\mathrm{m} = \mathbf{W}_\mathrm{b}\mathbf{M}^T \tag{3.28}$$

$$\boldsymbol{w}_\mathrm{m} = \boldsymbol{w}_\mathrm{b} - \mathbf{W}_\mathrm{b}\mathbf{M}^T\boldsymbol{m} \tag{3.29}$$

Here, the unitary property of the transformation matrix, $\mathbf{M}\mathbf{M}^T = \mathbf{I}$, was used. Both solutions, $(\mathbf{W}_\mathrm{m}, \boldsymbol{w}_\mathrm{m})$ for $\boldsymbol{k}_\mathrm{m}$ and $(\mathbf{W}_\mathrm{b}, \boldsymbol{w}_\mathrm{b})$ for $\boldsymbol{k}$, are equivalent, as the resulting loss function L_b, given by Eq. (3.13), has the same value in both cases.

3.3 Understanding ridge regression through principal components

In this section, ridge regression will be interpreted using the principal components of the RR input dataset $\mathbf{K}$. As demonstrated in the previous section, type-b ridge regression is invariant under a transformation of the input variables, $\boldsymbol{k} \to \mathbf{M}\boldsymbol{k} + \boldsymbol{m}$, involving a shift and a multiplication with a unitary matrix. Using the PC-transformation given by Eq. (3.4) and transforming the original type-b solutions, $(\mathbf{W}_\mathrm{b}, \boldsymbol{w}_\mathrm{b})$, according to Eq. (3.28) and Eq. (3.29), the original type-b RR output, $\hat{\boldsymbol{y}} = \mathbf{W}_\mathrm{b}\boldsymbol{k} + \boldsymbol{w}_\mathrm{b}$, can be transformed to the PC coordinate frame:

$$\hat{\boldsymbol{y}} = \mathbf{W}_\mathrm{pc}\boldsymbol{k}_\mathrm{pc} + \boldsymbol{w}_\mathrm{pc} \tag{3.30}$$

The PC-transformed quantities, $\boldsymbol{k}_\mathrm{pc}$, $\mathbf{W}_\mathrm{pc}$, and $\boldsymbol{w}_\mathrm{pc}$, are given by:

$$\boldsymbol{k}_\mathrm{pc} = \mathbf{P}(\boldsymbol{k} - \overline{\boldsymbol{k}}) \tag{3.31}$$
$$\mathbf{W}_\mathrm{pc} = \mathbf{W}_\mathrm{b}\mathbf{P}^T \tag{3.32}$$
$$\boldsymbol{w}_\mathrm{pc} = \overline{\boldsymbol{y}} \tag{3.33}$$

The principal component interpretation of type-b ridge regression can be achieved by analyzing the PC-transformed parameter matrix, $\mathbf{W}_\mathrm{pc}$, which now acts on the PC-transformed input data $\boldsymbol{k}_\mathrm{pc}$: The (i,j)th element of $\mathbf{W}_\mathrm{pc}$ connects the jth principal component with the ith output dimension. Through this analysis, described in the following, it is possible to interpret the regularization induced by $\beta > 0$ in terms of the principal components and corresponding explained variances. This can be done by rewriting the original solution of $\mathbf{W}_\mathrm{b}$, as given in Eq. (3.21), in terms of the SVD, $\mathbf{K}_\mathrm{c}^T = \mathbf{U}\mathbf{D}\mathbf{V}^T$, as given in Eq. (3.6).

$$\begin{aligned}\mathbf{W}_\mathrm{b} &= \mathbf{Y}\mathbf{K}_\mathrm{c}^T \left(\mathbf{K}_\mathrm{c}\mathbf{K}_\mathrm{c}^T + \beta\mathbf{I}\right)^{-1} \\ &\overset{\text{SVD}}{=} \mathbf{Y}\mathbf{U}\mathbf{D}\left(\mathbf{D}^2 + \beta\mathbf{I}\right)^{-1}\mathbf{V}^T\end{aligned} \tag{3.34}$$

Here, the unitary properties, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$ and $\mathbf{U}^T\mathbf{U} = \mathbf{I}$, were used. By noting that $\mathbf{V}^T = \mathbf{P}$, as demonstrated in Eq. (3.9), and utilizing the transformation given in Eq. (3.32), the PC-transformed parameter matrix, $\mathbf{W}_\mathrm{pc}$, is given by:

$$\mathbf{W}_\mathrm{pc} = \mathbf{Y}\mathbf{U}\mathbf{D}\left(\mathbf{D}^2 + \beta\mathbf{I}\right)^{-1} = \mathbf{Y}\mathbf{U} \cdot \mathrm{diag}\left(\frac{d_i}{d_i^2 + \beta}\right) \tag{3.35}$$

As $\mathbf{D}$ corresponds to the diagonal matrix holding the singular values, d_i, on its diagonal, the entire component $\mathbf{D}\left(\mathbf{D}^2 + \beta\mathbf{I}\right)^{-1}$ is also diagonal. The effect of regularization becomes apparent, when comparing Eq. (3.35) to its non-regularized, $\beta = 0$, version, corresponding to OLS regression:

$$\mathbf{W}_{\mathrm{pc},\beta=0} = \mathbf{Y}\mathbf{U}\mathbf{D}^{-1} = \mathbf{Y}\mathbf{U} \cdot \mathrm{diag}\left(\frac{1}{d_i}\right) \tag{3.36}$$

When the input data exhibits true multicollinearity, i.e. the explained variances vanish, $\phi_i = 0$ for all $i > \mathrm{rank}(\mathbf{K}_\mathrm{c})$, the singular value matrix $\mathbf{D}$ is not invertible. This is because the elements of $\mathbf{D}$ are related to the explained variances via $\phi_i = d_i^2/N$, as seen in Eq. (3.8). As a result, the

non-regularized OLS solution becomes undetermined. However, when regularization is included, $\beta > 0$, the problematic matrix $\mathbf{D}^{-1}$ is replaced by $\mathbf{D}\left(\mathbf{D}^2 + \beta\mathbf{I}\right)^{-1}$, enabling the inversion even for vanishing singular values. Assuming the calculation of $\mathbf{W}_{\mathrm{pc},\beta=0}$ is possible, which is the case when the input data is not truly multicollinear, the regularized solution, $\mathbf{W}_{\mathrm{pc}}$, can be related to $\mathbf{W}_{\mathrm{pc},\beta=0}$ by:

$$\mathbf{W}_{\mathrm{pc}} = \mathbf{W}_{\mathrm{pc},\beta=0} \cdot \mathrm{diag}\left(\frac{\phi_i}{\phi_i + \beta/N}\right) \tag{3.37}$$

This result is interesting as it shows how regularization in RR suppresses the contributions of principal components corresponding to small explained variances, ϕ_i, by the factor:

$$f(\phi_i) = \frac{\phi_i}{\phi_i + \beta/N} \tag{3.38}$$

The factor $f(\phi_i)$ acts as a low-pass filter on the explained variances ϕ_i. At $\phi_i = \beta/N$, the filter value is $f(\beta/N) = 0.5$, which can be interpreted as the *cutoff explained variance*, represented by ϕ_{co}.

$$\phi_{\mathrm{co}} = \beta/N \tag{3.39}$$

For larger explained variances, $\phi_i > \phi_{\mathrm{co}}$, the filter approaches 1, corresponding to no filtering, and for smaller explained variances, $\phi_i < \phi_{\mathrm{co}}$, the filter approaches 0, corresponding to strong filtering. The principal component index, $1 \leq i_{\mathrm{co}} \leq k_{\mathrm{dim}}$, where $\phi_{i_{\mathrm{co}}} \approx \phi_{\mathrm{co}}$, will be referred to as the *cutoff principal component* and it approximately measures the number of active principal components in the RR output. Mathematically, i_{co} is obtained by:

$$i_{\mathrm{co}} = \arg\min_i |f(\phi_i) - 0.5| = \arg\min_i |\phi_i - \phi_{\mathrm{co}}| \tag{3.40}$$

Chapter 4

Reservoir Computing

4.1 Introduction

Reservoir computing (RC) is a computing framework suited for temporal data processing, such as times series prediction and classification. RC is a unified framework [17] that originated from two independently proposed recurrent neural network (RNN) models, which were introduced as *Echo State Networks* (ESNs) from the field of machine learning in [4], and *Liquid State Machines* (LSMs) from the field of computational neuroscience [5].

In the following, the application of RC will be limited to tasks that model a relationship between temporal input data $\boldsymbol{x}(t)$ and the desired temporal output data $\boldsymbol{y}(t)$. In RC, the temporal input $\boldsymbol{x}(t)$ is coupled to a high-dimensional, nonlinear dynamical system called the *reservoir* so that the state of the reservoir $\boldsymbol{r}(t)$ becomes a function of the current and previous inputs, where the influence of past inputs fades away over time. This is called the *echo state property* which is an important aspect of reservoir computing. Assuming discrete time steps Δt, the state of the reservoir $\boldsymbol{r}(t)$ can be generally written as:

$$\boldsymbol{r}(t) = \text{func}(\boldsymbol{x}(t), \boldsymbol{x}(t - \Delta t), \boldsymbol{x}(t - 2\Delta t), \dots) \tag{4.1}$$

As a result, the reservoir state $\boldsymbol{r}(t)$ encodes the information of the inputs' recent history nonlinearly in a high-dimensional space. With a suitable choice of the reservoir, its high-dimensionality and nonlinearity can be leveraged in such a way that the desired output, $\boldsymbol{y}(t)$, can be efficiently read out from the reservoir states, $\boldsymbol{r}(t)$, by a simple learning algorithm such as ridge regression.

It is important to note that the properties of the reservoir itself are fixed at all times, while only the relationship between $\boldsymbol{r}(t)$ and $\boldsymbol{y}(t)$ is learned. This is a great advantage of RC, as the problematic training of internal reservoir parameters, as done in classical RNNs, can be bypassed. Additionally, besides very fast learning, this property allows for using physical implementations of reservoirs, which do not allow for detailed internal parameter adjustments. For example, in [7] the reservoir is realized by a bucket of water. The input signal is coupled to the reservoir through electric motors that create ripples on the water surface. By video recording the surface, corresponding to creating the digital representation of the reservoir state, $\boldsymbol{r}(t)$, the desired output can be learned by mapping the reservoir states $\boldsymbol{r}(t)$ to $\boldsymbol{y}(t)$. Among other tasks, one task was to classify a spoken digit which could be either "zero" or "one". As reviewed in [3] there are many other more practical physical RC systems such as photonic RC, spintronic RC and quantum RC.

In this thesis, the usual software-based implementation of RC via an artificial recurrent neural network is used, as is described in the following section. Furthermore, this thesis is restricted to time series prediction tasks, which relates to the desired output being the next step of the input data, $\boldsymbol{y}(t) = \boldsymbol{x}(t + \Delta t)$. One type of time series that has been shown to be well predictable with RC [18, 2] are the trajectories of chaotic dynamical systems, which were described in chapter 2. In these cases, RC excels not only at creating accurate short-term predictions, but also at replicating the ergodic properties of the system's attractor.

4.2 Basic implementation

In the usual software-based implementation of RC, the reservoir is realized using a recurrent neural network, with only the last layer being trained. The time dependent input, $\boldsymbol{x}(t)$, which has a dimension of x_{dim}, is dynamically coupled to the high-dimensional reservoir state, $\boldsymbol{r}(t)$, which has a dimension of r_{dim}, through the following equation:

$$\boldsymbol{r}(t) = \tanh\left(\mathbf{W}_{\text{in}}\boldsymbol{x}(t) + \mathbf{A}\boldsymbol{r}(t - \Delta t)\right) \tag{4.2}$$

The input, $\boldsymbol{x}(t)$, is connected to the reservoir through the $r_{\text{dim}} \times x_{\text{dim}}$ input matrix, $\mathbf{W}_{\text{in}}$, and the recurrent connections between consecutive reservoir states, $\boldsymbol{r}(t)$, are introduced through the $r_{\text{dim}} \times r_{\text{dim}}$ adjacency matrix $\mathbf{A}$, also known as the *reservoir network*. The hyperbolic tangent, $\tanh(\cdot)$, is used as the node activation function, introducing the important nonlinearities by acting element-wise on each dimension of its argument. Other activation functions such as the *Sigmoid* or *ReLU* function can also be used, but in this thesis, the usual tanh-function is applied. The non-dynamic quantities $\mathbf{A}$ and $\mathbf{W}_{\text{in}}$ are initialized once before the training and then kept fixed. Their initialization details will be discussed in the final paragraph of this section. From the recursive form of Eq. (4.2) it is clear that the current reservoir state, $\boldsymbol{r}(t)$, depends not only on the current input, $\boldsymbol{x}(t)$, but also on the previous inputs, $\boldsymbol{x}(t - i \cdot \Delta t)$, with $i = 1, 2, \ldots$, mediated through the previous reservoir states, $\boldsymbol{r}(t - \Delta t)$.

The application of RC as a machine learning model for time series prediction can be divided into the following training and prediction phases:

Training To begin the training phase, the initial reservoir state $\boldsymbol{r}(t_0)$ must be initialized to some arbitrary value. Here, $\boldsymbol{r}(t_0) = [0, 0, \ldots, 0]^T$ is used. The reservoir is then dynamically driven by the input $\boldsymbol{x}(t)$ by iteratively applying Eq. (4.2) for $N_{\text{TS}} + N_{\text{T}}$ time steps. The subscripts TS and T corresponds to *train-sync* and *train*, respectively. The first N_{TS} iterations are used to synchronize the reservoir state, $\boldsymbol{r}(t)$, with the driving input, $\boldsymbol{x}(t)$. This eliminates any transient dynamics that result from the arbitrary initial condition, $\boldsymbol{r}(t_0)$, if a suitable reservoir matrix $\mathbf{A}$ has been chosen. After synchronization, the following N_{T} reservoir states will be saved in a $r_{\text{dim}} \times N_{\text{T}}$ matrix $\mathbf{R} = [\boldsymbol{r}[1], \boldsymbol{r}[2], \ldots, \boldsymbol{r}[N_{\text{T}}]]$, where the index $i = 1, 2, \ldots N_{\text{T}}$ corresponds to the time $t = t_0 + (N_{\text{TS}} + i)\Delta t$.

The idea behind reservoir computing is that the high-dimensional reservoir state, $\boldsymbol{r}(t)$, encodes the inputs, $\boldsymbol{x}(t), \boldsymbol{x}(t - \Delta t), \ldots$, in such a way that the desired output, $\boldsymbol{y}(t)$, can be computed by a simple linear function of the reservoir states. If the desired outputs, represented by the matrix $\mathbf{Y} = [\boldsymbol{y}[1], \boldsymbol{y}[2], \ldots, \boldsymbol{y}[N_{\text{T}}]] \in \mathbb{R}^{y_{\text{dim}} \times N_{\text{T}}}$, are available as training data, the parameters of the linear model can be learned through the use of ridge regression, as outlined in Sec. 3.2. Here, the reservoir states, $\boldsymbol{r}$, serve as the explanatory variables in ridge regression, previously denoted by $\boldsymbol{k}$. Usually, the simple linear model M1 without an intercept, as presented in Eq. (3.10), is used. This corresponds to type-a ridge regression. The reservoir's output, denoted as $\boldsymbol{y}_{\text{r}}$, that aims to approximate the true output, $\boldsymbol{y}_{\text{r}} \approx \boldsymbol{y}$, is given by:

$$\boldsymbol{y}_{\text{r}}(t) = \mathbf{W}_{\text{out}}\boldsymbol{r}(t) \tag{4.3}$$

As outlined in section 3.2 for type-a ridge regression, the output matrix $\mathbf{W}_{\text{out}} \in \mathbb{R}^{y_{\text{dim}} \times r_{\text{dim}}}$ can be obtained by:

$$\mathbf{W}_{\text{out}} = \mathbf{Y}\mathbf{R}^T \left(\mathbf{R}\mathbf{R}^T + \beta\mathbf{I}\right)^{-1} \tag{4.4}$$

The regularization parameter, represented by β, is a crucial hyperparameter in reservoir computing. The use of ridge regression in RC is beneficial because the reservoir states tend to be highly correlated, resulting in approximate multicollinearity. This poses a problem for ordinary least squares regression, as discussed in section 3.3.

The RC training procedure that has been developed thus far can handle any type of output, $\boldsymbol{y}(t)$, which may be very different from the input, $\boldsymbol{x}(t)$. In the case of time series prediction, the goal

is to predict the next step of the time series, $\boldsymbol{u}(t + \Delta t)$, based on a section of previous states, $\boldsymbol{u}(t), \boldsymbol{u}(t - \Delta t), \ldots$. In order to apply the task of time series prediction to RC, both the training input and desired output will be given by the true time series data, $\boldsymbol{u}(t)$: Specifically, the training input is equal to the time series at a particular time, $\boldsymbol{x}(t) = \boldsymbol{u}(t)$, and the desired output is equal to the time series at the next step, $\boldsymbol{y}(t) = \boldsymbol{u}(t + \Delta t)$. With $N_{\mathrm{TS}} + N_{\mathrm{T}} + 1$ time steps of the true time series, $\boldsymbol{u}(t)$, being available, the training-synchronization input is given by:

$$\text{train-sync input: } \boldsymbol{x}[i] = \boldsymbol{u}[i] \qquad i = -N_{\mathrm{TS}} + 1, \ldots, 0 \tag{4.5}$$

The training input and output used for ridge regression fitting are given by:

$$\text{train-fit in- and output: } \begin{cases} \boldsymbol{x}[i] & = \boldsymbol{u}[i] \\ \boldsymbol{y}[i] & = \boldsymbol{u}[i+1] \end{cases} \qquad i = 1, \ldots N_{\mathrm{T}} \tag{4.6}$$

Upon determining the output matrix, $\mathbf{W}_{\mathrm{out}}$, through Eq. (4.4), the training phase is completed: $\mathbf{W}_{\mathrm{out}}$ connects the reservoir state, $\boldsymbol{r}(t)$, to the reservoir output, which approximates the next time step in the time series: $\boldsymbol{y}_{\mathrm{r}}(t) \approx \boldsymbol{u}(t + \Delta t)$.

Prediction During the prediction phase, the driving input, $\boldsymbol{x}(t)$, in Eq. (4.2) is replaced with the prediction generated by the reservoir at the preceding time step: $\boldsymbol{x}(t) = \boldsymbol{y}_{\mathrm{r}}(t - \Delta t)$. This results in the prediction phase being defined by the following two equations:

$$\boldsymbol{y}_{\mathrm{r}}(t) = \mathbf{W}_{\mathrm{out}} \boldsymbol{r}(t) \tag{4.7}$$

$$\boldsymbol{r}(t + \Delta t) = \tanh\left(\mathbf{W}_{\mathrm{in}} \boldsymbol{y}_{\mathrm{r}}(t) + \mathbf{A}\boldsymbol{r}(t)\right) \tag{4.8}$$

First, in Eq. (4.7) the prediction of the next time series state $\boldsymbol{y}_{\mathrm{r}}(t) \approx \boldsymbol{u}(t + \Delta t)$ is obtained from the current reservoir state $\boldsymbol{r}(t)$. Then, the next reservoir state $\boldsymbol{r}(t + \Delta t)$ is calculated in Eq. (4.8) by using the prediction $\boldsymbol{y}_{\mathrm{r}}(t)$ as its current driving input. As the RC output is fed back into the reservoir, the reservoir state, $\boldsymbol{r}(t)$, now evolves autonomously in a closed-loop. This means that the subsequent reservoir state, $\boldsymbol{r}(t + \Delta t)$, is dependent only on the preceding reservoir state, $\boldsymbol{r}(t)$, and not on any external input. This can be demonstrated by inserting Eq. (4.7) into Eq. (4.8):

$$\boldsymbol{r}(t + \Delta t) = \tanh\left([\mathbf{W}_{\mathrm{in}} \mathbf{W}_{\mathrm{out}} + \mathbf{A}]\boldsymbol{r}(t)\right) \tag{4.9}$$

Given that Eq. (4.9) corresponds to an iterative map, as described in Eq. (2.1), the dynamics of the reservoir during the prediction phase represent a high-dimensional discrete dynamical system.

In order to initiate the closed-loop, given by Eq. (4.9), an initial synchronized reservoir state, $\boldsymbol{r}(t)$, is required. If the task is to predict the continuation of the training data, the last saved training reservoir state, $\boldsymbol{r}[N_{\mathrm{T}}]$, can be utilized. However, if the task is to predict the continuation of an arbitrary section of the time series, the reservoir must first be synchronized to that section. This can be simply archived by driving the reservoir in the open-loop configuration with the true time series as external input, $\boldsymbol{x}(t) = \boldsymbol{u}(t)$, following Eq. (4.2). The number of time steps used for synchronization is denoted as N_{PS}, where the subscript PS stands for *predict-sync*. The prediction can then be initiated by taking the last resulting reservoir state, which is now synchronized, and repeatedly applying Eq. (4.7) and Eq. (4.8) for the desired number of prediction steps. The number of prediction steps is denoted as N_{P}, where the subscript stands for *prediction*. The resulting reservoir outputs, $\boldsymbol{y}_{\mathrm{r}}(t)$, represent the forecasted N_{P}-long continuation of the N_{PS}-long section of the true time series.

Input matrix $\mathbf{W}_{\mathrm{in}}$ and reservoir matrix $\mathbf{A}$ In the RC framework, the input matrix, $\mathbf{W}_{\mathrm{in}}$, and the reservoir network, $\mathbf{A}$, are important parameters. The elements of the input matrix, $W_{\mathrm{in},ij}$, indicate the strength of the connection between the input variable x_j and the reservoir node r_i.

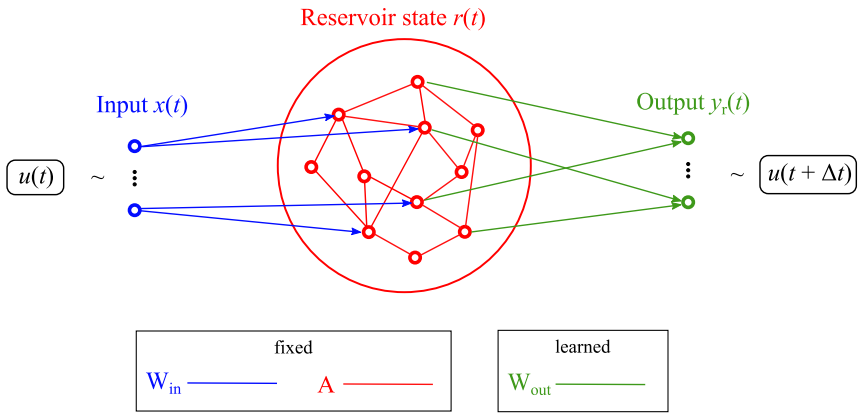


Figure 4.1: Basic RC architecture for time series prediction.

Similarly, the elements of the reservoir network, A_{ij}, represent the strength of the coupling between the reservoir nodes r_i and r_j.

Typically, $\mathbf{W}_{\text{in}}$ is populated either in a sparse or dense way with random numbers drawn from a uniform distribution within the range of $[-\sigma, \sigma]$, where σ represents the *input strength*. A larger input strength, σ, corresponds to a stronger coupling of the input to the reservoir. In the case of a dense population, all elements $W_{\text{in},ij}$ are drawn from the uniform distribution. In contrast, a sparse population, originally introduced in [19], connects each reservoir node r_i to only one randomly chosen input variable, resulting in each row of $\mathbf{W}_{\text{in}}$ having only one non-zero element. In this thesis, the sparse input matrix is used.

Previous studies, such as [20] and [21], have investigated the effect of different network topologies for the reservoir matrix $\mathbf{A}$. In this thesis, the classical approach of creating a large, random, and sparse network is employed, as it has been shown to yield rich reservoir dynamics, as already noted in Jaeger's original work on ESNs [4]. Specifically, corresponding to the method outlined in [19], the network is generated using a weighted Erdős-Rényi random network, created through the following steps: Starting from an empty network, each possible directed network connection between node i and node j, where $i \neq j$, is created with a probability p. This relates to an initially all-zero adjacency matrix, where each element, apart from diagonal elements, is set to 1 with the probability p. In the case of an undirected Erdős-Rényi network, the opposite connection $i \leftarrow j$ is also created upon the creation of $i \rightarrow j$, which makes the adjacency matrix symmetric in that case. In either case, each node is on average connected to $d = p \cdot (r_{\text{dim}} - 1)$ other nodes, where the *average node degree* d represents a hyperparameter of RC. Next, the network is converted into a weighted one by replacing every 1 in the adjacency matrix with a randomly drawn value from the uniform distribution $[-1, 1]$. The resulting adjacency matrix is denoted as $\tilde{\mathbf{A}}$. To obtain the final reservoir matrix $\mathbf{A}$, the previously created matrix, $\tilde{\mathbf{A}}$, is then rescaled to have the desired spectral radius, $\rho_0 > 0$:

$$\mathbf{A} = \rho_0 \frac{\tilde{\mathbf{A}}}{\rho(\tilde{\mathbf{A}})} \tag{4.10}$$

The spectral radius, $\rho(\cdot)$, of a square matrix is equal to its largest absolute eigenvalue, which indicates the maximal scaling produced by a multiplication with that matrix. Therefore, the chosen spectral radius, ρ_0, of the reservoir network controls the strength of the influence of the previous reservoir state on the next one, which represents an important hyperparameter of RC. It can be observed that a large spectral radius, $\rho > 1$, often leads to unstable reservoir dynamics, as the influence of previous states cannot fade away. To ensure the *echo state property*, a spectral radius of $\rho < 1$ is usually chosen. Despite this, a spectral radius $\rho > 1$ may still fulfill the *echo state property* under certain conditions, which may result in successful predictions [4].

The basic reservoir computing architecture outlined in this section is depicted in Fig. 4.1.

4.3 Extensions

There is a significant amount of research dedicated to extending and enhancing the basic RC implementation, such as in [22] which explores deep RC architectures to improve the prediction of time series with multiple time scales, and in [23] and [24], where the RC architecture is parallelized to handle high-dimensional input data, such as in climate simulations. In this section, the focus lies on simple extensions that address some of the limitations of the basic RC implementation discussed in the previous section 4.2.

Node bias One limitation of the basic RC implementation is the issue of point symmetry when using the $\tanh(\cdot)$ activation function, as pointed out in [25]. In this case, the learned output matrix, $\mathbf{W}_{\text{out}}$, is invariant when the reservoir is trained with $\boldsymbol{u}(t)$ or $-\boldsymbol{u}(t)$, resulting in the possibility of the prediction flipping onto the so-called mirror attractor. To break this symmetry, one solution is to add a node bias vector $\boldsymbol{b}$ to the reservoir update equation:

$$\boldsymbol{r}(t) = \tanh\left(\mathbf{W}_{\text{in}}\boldsymbol{x}(t) + \mathbf{A}\boldsymbol{r}(t - \Delta t) + \boldsymbol{b}\right) \tag{4.11}$$

The node bias vector $\boldsymbol{b}$ is an r_{dim}-dimensional vector with elements randomly selected from the uniform distribution $[-\sigma_{\text{b}}, \sigma_{\text{b}}]$. The *node bias scale* σ_{b} represents an additional hyperparameter of RC.

Readout function Another method for breaking the symmetry in the reservoir equations and enhancing the overall prediction capabilities is to apply a readout function, $\psi(\cdot)$, to the reservoir states prior to fitting the output.

$$\boldsymbol{y}_{\text{r}}(t) = \mathbf{W}_{\text{out}}\psi(\boldsymbol{r}(t)) = \mathbf{W}_{\text{out}}\tilde{\boldsymbol{r}}(t) \tag{4.12}$$

The resulting generalized reservoir states, $\tilde{\boldsymbol{r}}(t) = \psi(\boldsymbol{r}(t))$, often represent an extension of the original reservoir states. As an example, the readout function in Lu et al. [19] expands the reservoir state by including its squared values:

$$\tilde{\boldsymbol{r}} = \psi_{\text{ext. Lu}}(\boldsymbol{r}) = [r_1, r_2, \ldots, ...r_{r_{\text{dim}}}, r_1^2, r_2^2, \ldots r_{r_{\text{dim}}}^2] \tag{4.13}$$

A similar readout function that avoids doubling the total number of regressor variables, while still utilizing the squares of the reservoir states, is one that replaces every second element in $\boldsymbol{r}$ with its square:

$$\tilde{\boldsymbol{r}} = \psi_{\text{Lu}}(\boldsymbol{r}) = [r_1, r_2^2, \ldots, r_{r_{\text{dim}}-1}, r_{r_{\text{dim}}}^2] \tag{4.14}$$

Following [25], the full-squaring in Eq. (4.13) is referred to as the *extended Lu readout*, while the alternating squaring in Eq. (4.14) is simply called the *Lu readout*.

As a result of using readout functions, the calculation of the output matrix, $\mathbf{W}_{\text{out}}$, now acts on the generalized reservoir states, $\tilde{\mathbf{R}} = [\tilde{\boldsymbol{r}}[1], \tilde{\boldsymbol{r}}[2], \ldots, \tilde{\boldsymbol{r}}[N_{\text{T}}]]$:

$$\mathbf{W}_{\text{out}} = \left(\tilde{\mathbf{R}}^T\tilde{\mathbf{R}} + \beta\mathbf{I}\right)^{-1}\tilde{\mathbf{R}}^T\mathbf{Y} \tag{4.15}$$

Input standardizer As is common with many machine learning models, standardizing the input data by centering it and scaling it to unit variance can be beneficial. This can be achieved by incorporating an input standardizer into the reservoir computing architecture. The input standardizer is placed between the input and the reservoir, and standardizes the input data $\boldsymbol{x}(t) \to \hat{\boldsymbol{x}}(t)$

before it enters the reservoir, thus modifying the term $\mathbf{W}_{\text{in}}\boldsymbol{x}(t)$ to $\mathbf{W}_{\text{in}}\hat{\boldsymbol{x}}(t)$ in the reservoir update Eq. (4.11). The standardization is performed through the following transformation:

$$\hat{\boldsymbol{x}}(t) = S(\boldsymbol{x}(t)) = \mathbf{s}^{-1} \cdot (\boldsymbol{x}(t) - \overline{\boldsymbol{x}}) \tag{4.16}$$

The quantities $\mathbf{s}$ and $\overline{\boldsymbol{x}}$ are calculated from the training input data $\boldsymbol{x}[i]$, where $i = 1, \ldots, N_{\text{T}}$, prior to the standard training process. The x_{dim}-dimensional vector $\overline{\boldsymbol{x}}$ represents the mean of the training data:

$$\overline{\boldsymbol{x}} = \frac{1}{N_{\text{T}}} \sum_{i=1}^{N_{\text{T}}} \boldsymbol{x}[i] \tag{4.17}$$

The $x_{\text{dim}} \times x_{\text{dim}}$ diagonal matrix $\mathbf{s}$ holds the dimension-wise standard deviations on its diagonal entries:

$$s_{jj}^2 = \frac{1}{N_{\text{T}}} \sum_{i=1}^{N_{\text{T}}} (x_j[i] - \overline{x}_j)^2 \tag{4.18}$$

Ridge regression with intercept In section 3.2, it was discussed that the simple linear model M1, represented by Eq. (3.10), can be improved by including an additional intercept term, $\boldsymbol{w}$, as represented by model M2 in Eq. (3.11). Similarly, this concept can be applied to reservoir computing by extending the previous reservoir-output-connection Eq. (4.3) to include the intercept term, $\boldsymbol{w}$:

$$\boldsymbol{y}_{\text{r}}(t) = \mathbf{W}_{\text{out}}\boldsymbol{r}(t) + \boldsymbol{w} \tag{4.19}$$

As discussed in section 3.2, ridge regression using the linear model M2 can be further divided into type-b and type-c. Type-b does not penalize the intercept while type-c does. While type-c RR is slightly easier to be implemented as it only requires extending the reservoir states by a dimension with a constant value, type-b is more accurate. In the existing literature on reservoir computing, the majority of works that include the intercept also penalize it, as seen in studies such as [26, 25]. However, there are also works that use type-b RR, such as [19]. The widespread use of type-a and type-c RR can be attributed to the fact that, in the typical case of reservoir computing, they perform equivalently to type-b. This will be demonstrated in section 4.6 for the Lorenz system. However, as demonstrated in appendix B, there are extreme cases where the superiority of type-b becomes apparent. The primary reason for employing type-b RR in this thesis is not its theoretically superior fitting capabilities, but rather its mathematically simple connection to principal components, as will be utilized in chapter 5.

4.4 Measuring prediction quality

In section 4.2, it was demonstrated how a reservoir computer can be utilized for time series prediction. To evaluate the quality of the prediction produced by the reservoir, it is useful to predict a section of the time series for which the true data is known. By doing so, the prediction, $\boldsymbol{y}_{\text{r}}(t)$, can be directly compared to the true data, $\boldsymbol{y}(t)$. In this section, a measure called the *valid time* is introduced, which quantifies the duration for which the predicted time series matches the true time series. Additionally, as the reservoir setup contains three sources of randomness, namely the input matrix $\mathbf{W}_{\text{in}}$, the network matrix $\mathbf{A}$, and the node bias vector $\boldsymbol{b}$, different random realization of these quantities result in different predictions, $\boldsymbol{y}_{\text{r}}(t)$. Therefore, to properly assess the prediction quality of a particular reservoir architecture defined by its hyperparameters, the prediction must be repeated for an ensemble of reservoir realizations. Furthermore, as the prediction quality also depends on the time series sections used for training and prediction, the statistical significance

of the results can be further improved by training and predicting each reservoir realization on multiple sections of the time series. The implementation details of the ensemble experiment will also be discussed in this section.

Valid time t_v The length of time for which the prediction, $\boldsymbol{y}_\mathrm{r}(t)$, closely matches the true time series, $\boldsymbol{y}(t)$, is known as the *valid time* or *prediction horizon*. Following the implementation of [10], the first step to calculating the valid time, t_v, is to compute the normalized time-dependent error, $e(t)$, between the prediction, $\boldsymbol{y}_\mathrm{r}(t)$, and the true time series, $\boldsymbol{y}(t)$:

$$e(t) = \frac{\|\boldsymbol{y}(t) - \boldsymbol{y}_\mathrm{r}(t)\|}{\langle \|\boldsymbol{y}(t)\|^2 \rangle^{1/2}} \tag{4.20}$$

The average over all prediction steps, N_P, is represented by $\langle\cdot\rangle$, and $\|\cdot\|$ denotes the L2-norm. The valid time, t_v, is defined as the elapsed prediction time until the time-dependent error, $e(t)$, exceeds a predefined error-threshold, $e_\mathrm{max} > 0$. The valid time is determined by finding the first time step, $k \in [1, N_\mathrm{P}]$, where $e(t) > e_\mathrm{max}$, and calculating $t_\mathrm{v} = k\Delta t$. In chaotic dynamical systems, the valid time is often measured in terms of the Lyapunov time, $\tau = 1/\lambda_\mathrm{max}$, and is then represented as $t_\mathrm{v}\lambda_\mathrm{max}$.

It is important to note that using the time-dependent error defined in Eq. (4.20) may lead to unwanted results if there is a large offset in the time series data. This is because the numerator of the error function is invariant under an offset of $\boldsymbol{y} + \delta$ and $\boldsymbol{y}_\mathrm{r} + \delta$, while the denominator is not. When the offset δ is large, the time-dependent error may never exceed the error-threshold, e_max, even though the prediction may deviate greatly from the true data. To address this issue, a better error function that is invariant under an offset can be calculated using Eq. (B.1). However, for the time series data used in this work, the error defined in Eq. (4.20) is sufficient.

Ensemble experiment To evaluate the ability of a reservoir architecture to predict a time series, $\boldsymbol{u}(t)$, the following ensemble experiment is carried out, which is additionally outlined in the algorithm 1 in more detail. To begin, an ensemble of n_R reservoir computing realizations are created by randomly selecting the elements of $\mathbf{W}_\mathrm{in}$, $\mathbf{A}$, and $\boldsymbol{b}$, as described in sections 4.2 and 4.3. Each RC realization is trained on n_T different training-sections, each of length $N_\mathrm{TD} + N_\mathrm{TS} + N_\mathrm{T}$, which includes N_TS train-synchronization steps and N_T train-fitting steps. To ensure that the different training-sections are independent of each other, the first N_TD steps of each section are discarded, where TD stands for *train-discard*. Next, each reservoir realization, i_R, that was trained on a specific training-section, i_T, is used to make predictions on n_P different prediction-sections, each consisting of $N_\mathrm{PD} + N_\mathrm{PS} + N_\mathrm{P}$ time steps. Following the prediction-process outlined in section 4.2, each trained reservoir is first synchronized with the true time series for N_PS steps, after which it is run in the closed-loop configuration to generate N_P prediction steps. Again, the first N_PD time steps of every prediction-section are discarded, where PD stands for *predict-discard*. The training and prediction process applied to each RC realization is visualized in Fig. 4.2. Finally, for each i_R, i_T and i_P the reservoir prediction is compared to the true time series for that prediction-section, and the valid times, $t_\mathrm{v}(i_\mathrm{R}, i_\mathrm{T}, i_\mathrm{P})$, are recorded. The final result of the prediction quality is determined by using the median value from the set of valid times, with the lower and upper quartiles serving as the range for error.

Experiment parameters The parameters for the ensemble experiment utilized in this study are outlined in table 4.1. The choice of n_R, n_T, and n_P results in an effective ensemble of $15 \times 15 \times 10 = 2250$ valid time calculations.

Training			Prediction			Ensemble			Valid time
N_TD	N_TS	N_T	N_PD	N_PS	N_P	n_R	n_T	n_P	e_max
1000	100	2000	1000	100	2000	15	15	10	0.4

Table 4.1: Default ensemble experiment parameters.

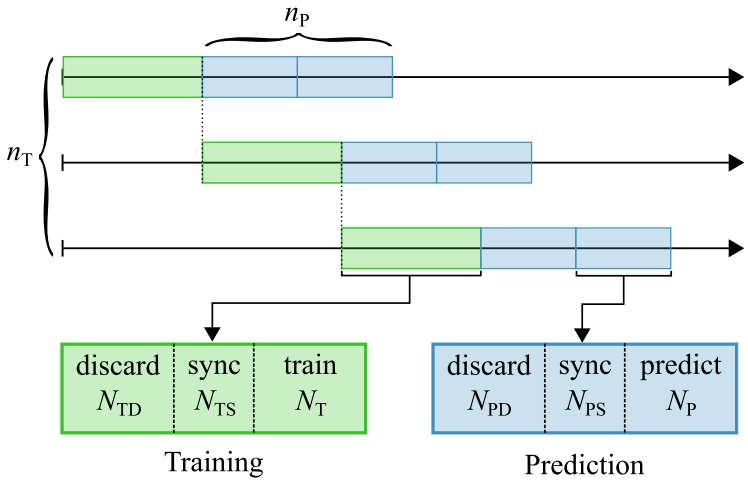


Figure 4.2: Training- and prediction-sections of ensemble experiment as outlined in algorithm 1. For demonstration purposes $n_T = 3$ and $n_P = 2$ is used. Each training-section contains a discard, synchronize and train subsection, and each prediction-section contains a discard, synchronize and predict subsection. The arrow represents the true time series data where the left starting point represents the first data point $\boldsymbol{u}[0]$.

Algorithm 1 Ensemble experiment to assess prediction quality of RC using valid times

$\boldsymbol{u}[0], \boldsymbol{u}[1], \ldots$: time series data.
n_R: number of reservoir realizations.
n_T: number of training-sections.
n_P: number of prediction-sections for each training section.
N_{TD}: time steps to discard before training.
N_{TS}: time steps used for training-synchronization.
N_T: time steps used for training-fit.
N_{PD}: time steps to discard before prediction.
N_{PS}: time steps used for prediction-synchronization.
N_P: time steps to be predicted.

Output:

$t_v(i_R, i_T, i_P)$: Valid time for each combination of reservoir realization i_R, training-section i_T and prediction-section i_P.

for $i_R = 1 : n_R$ **do**
  get reservoir realizations by randomly drawing $\mathbf{W}_{in}$, $\mathbf{A}$, and $\boldsymbol{b}$
  set: TrainStart $\leftarrow N_{TD}$
  for $i_T = 1 : n_T$ **do**
    set: TrainEnd $\leftarrow$ TrainStart $+ N_{TS} + N_T - 1$
    get train-fit input $\mathbf{X}_{train} = \boldsymbol{u}[\text{TrainStart} + N_{TS}], \ldots, \boldsymbol{u}[\text{TrainEnd}]$
    get train-fit output $\mathbf{Y}_{train} = \boldsymbol{u}[\text{TrainStart} + N_{TS} + 1], \ldots, \boldsymbol{u}[\text{TrainEnd} + 1]$
    calculate the input-standardizer, $S(\cdot)$, from $\mathbf{X}_{train}$
    initialize the reservoir state $\boldsymbol{r} = [0, 0, \ldots, 0]^T$
    synchronize the reservoir with $\boldsymbol{u}[\text{TrainStart}], \ldots \boldsymbol{u}[\text{TrainStart} + N_{TS} - 1]$
    train the reservoir with $\mathbf{X}_{train}$ and $\mathbf{Y}_{train}$
    set: TrainStart $\leftarrow$ TrainEnd $+ N_{TD} + 1$
    set: PredStart $\leftarrow$ TrainEnd $+ N_{PD} + 1$
    for $i_P = 1 : n_P$ **do**
      set: PredEnd $\leftarrow$ PredStart $+ N_{PS} + N_P - 1$
      reset the reservoir state $\boldsymbol{r} = [0, 0, \ldots, 0]^T$
      synchronize reservoir with $\boldsymbol{u}[\text{PredStart}], \ldots, \boldsymbol{u}[\text{PredStart} + N_{PS} - 1]$
      predict with reservoir for N_P steps and save prediction $\mathbf{Y}_{pred}$
      get true time series $\mathbf{Y}_{true} = \boldsymbol{u}[\text{PredStart} + N_{PS}], \ldots, \boldsymbol{u}[\text{PredEnd}]$
      calculate valid time, $t_v(i_R, i_T, i_P)$, using $\mathbf{Y}_{pred}$ and $\mathbf{Y}_{true}$
      set: PredStart $\leftarrow$ PredEnd $+ N_{PD} + 1$

4.5 Default RC setup

In the preceding section, the numerical experiment was presented that examines the ability of a specific RC setup to predict a chaotic dynamic system. In this section, the default RC setup, constituted by the default choice of extensions and hyperparameter values, is introduced. This default setup will be used throughout the thesis, unless otherwise specified.

The default RC setup will employ the input standardizer, Eq. (4.16), as it is a standard practice in machine learning and allows for greater comparability between different kinds of time series. To break the symmetry within the reservoir update equation, the default setup will also include the node bias, $\boldsymbol{b}$, as presented in Eq. (4.11). An additional nonlinear readout function, such as *Lu readout*, Eq. (4.14), or *extended Lu readout*, Eq. (4.13), is omitted, as the symmetry is already broken by the node bias. Furthermore, using the reservoir states directly as the input variables for ridge regression allows for greater interpretability in the principal component analysis, which will be presented in chapter 5. For the same reason, type-b RR will be used as it allows for a direct interpretation through the principal components, as described in section 3.3.

The default RC architecture is summarized by the following equations:

Input:

$$\boldsymbol{x}(t) = \tilde{\boldsymbol{u}}(t) \tag{4.21}$$

$$\tilde{\boldsymbol{u}}(t) = \begin{cases} \boldsymbol{u}(t) & \text{for synchronization and training} \\ \boldsymbol{y}_\mathrm{r}(t - \Delta t) & \text{for prediction} \end{cases} \tag{4.22}$$

Input standardizer:

$$\hat{\boldsymbol{x}}(t) = S(\boldsymbol{x}(t)) \tag{4.23}$$

Reservoir update equation:

$$\boldsymbol{r}(t) = \tanh\left(\mathbf{W}_\mathrm{in}\hat{\boldsymbol{x}}(t) + \mathbf{A}\boldsymbol{r}(t - \Delta t) + \boldsymbol{b}\right) \tag{4.24}$$

Ridge regression input:

$$\boldsymbol{k}(t) = \boldsymbol{r}(t) \tag{4.25}$$

Ridge regression output:

$$\boldsymbol{y}_\mathrm{r}(t) = \mathbf{W}_\mathrm{out}\boldsymbol{k}(t) + \boldsymbol{w} \tag{4.26}$$

The seemingly superfluous equations, Eq. (4.21) and Eq. (4.25), are listed here as they constitute the entry point for the hybrid reservoir computing approach described in chapter 6.

The chosen hyperparameter values are listed in table 4.2:

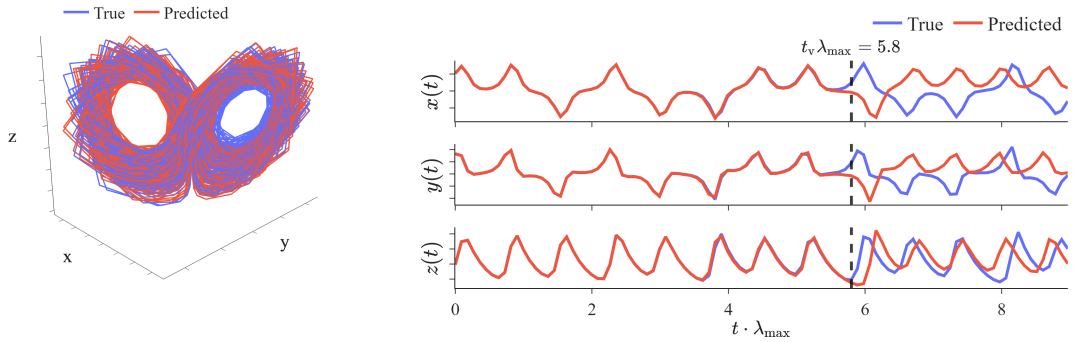
Reservoir				Input		Node bias	Ridge regression	
r_dim	d	ρ_0	$\mathbf{A}$-type	σ	$\mathbf{W}_\mathrm{in}$-type	σ_b	β	type
500	5.0	0.4	undirected	1.0	sparse	0.4	10^{-7}	b

Table 4.2: Default RC hyperparameters.

4.6 Prediction demonstration: Lorenz system

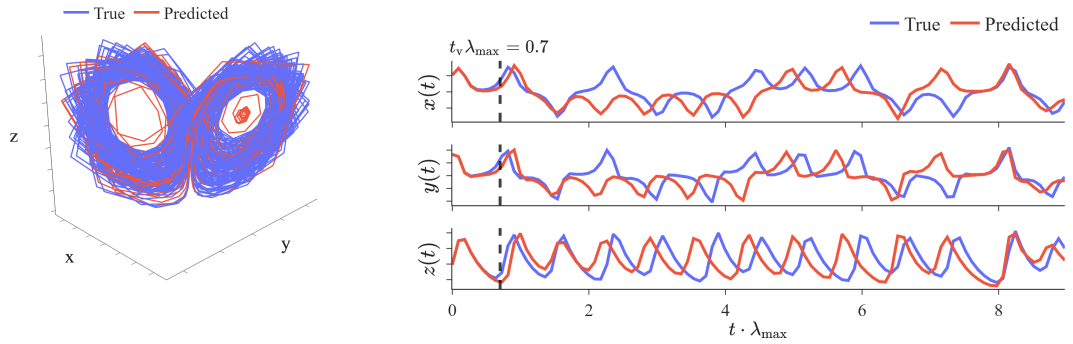
To showcase the RC framework presented in this chapter, the default RC setup outlined in the previous section will be applied to predict the three-dimensional Lorenz system, whose flow is given in Eq. (2.10). Additionally, variations from the default setup will be explored in order to demonstrate the impact of the RC extensions and hyperparameter values.

Trajectory plots Figures 4.3a and 4.3b show the true and predicted trajectories of the Lorenz system generated by a single reservoir realization using the default setup. As seen in Fig. 4.3b, the predicted time series aligns very closely with the true time series until a valid time of $t_v\lambda_{max} \approx 5.8$. From the three-dimensional trajectory in Fig. 4.3a, where all $N_P = 2000$ prediction steps are plotted, one can see that the reservoir is able to accurately replicate the true Lorenz attractor. Whereas the prediction shown in figures 4.3a and 4.3b utilizes the default reservoir dimension of $r_{dim} = 500$, the prediction shown in figures 4.3c and 4.3d are generated by a smaller reservoir of $r_{dim} = 50$. In this case, the prediction only archives a valid time of $t_v\lambda_{max} \approx 0.7$. From the three-dimensional trajectory in Fig. 4.3c, one can observe clear differences between the predicted and the true Lorenz attractor: The prediction collapses into a single point located in the middle of the right wing of the true Lorenz attractor.



(a) 3D-Trajectory ($r_{dim} = 500$)

(b) Time series ($r_{dim} = 500$)



(c) 3D-Trajectory ($r_{dim} = 50$)

(d) Time series ($r_{dim} = 50$)

Figure 4.3: Demonstration of a good and bad prediction of the Lorenz system using a larger and a smaller reservoir. The left column shows the three-dimensional trajectories of the true and predicted time series for all $N_P = 2000$ prediction steps, while the right column displays the first 100 steps of the time series for each dimension. (a) and (b): A good prediction using the default RC setup, which uses $r_{dim} = 500$. (c) and (d): A bad prediction using the same setup but with a smaller reservoir dimension of $r_{dim} = 50$.

Violin plots The ensemble experiment described in Sec. 4.4 generates a total of $n_R \times n_T \times n_P = 2250$ valid time calculations. The distributions of these valid times for the Lorenz system are represented in Fig. 4.4 through the use of violin plots, where the default RC setup is highlighted in green. Fig 4.4a compares the three different types of ridge regression, a, b, and c, and shows that there are no significant differences between them. However, when dealing with extremely uncentered time series data, as will be discussed in Appendix B, notable differences between the different RR types can be observed. The influence of the nonlinear readout functions, *Lu readout* and *extended Lu readout*, compared to using no readout is depicted in Fig. 4.4b. It can be observed

that both readouts lead to higher valid times compared to using no readout, with the *extended Lu readout* performing the best. The superiority of the *extended Lu readout* can be explained by its incorporation of all the regressor variables of normal RC, as well as their squared values, which doubles the number of regressor variables from 500 to 1000. The advantage of the squaring can also be observed for the non-extended *Lu readout*, which also results in higher valid times compared to the normal linear readout, while using the same number of regressor variables. It is noteworthy that the distributions of valid times for all cases shown in Fig. 4.4 are very broad. For the default RC setup, they range from $t_v\lambda_{max} \approx 2.5$ to $t_v\lambda_{max} \approx 14$.

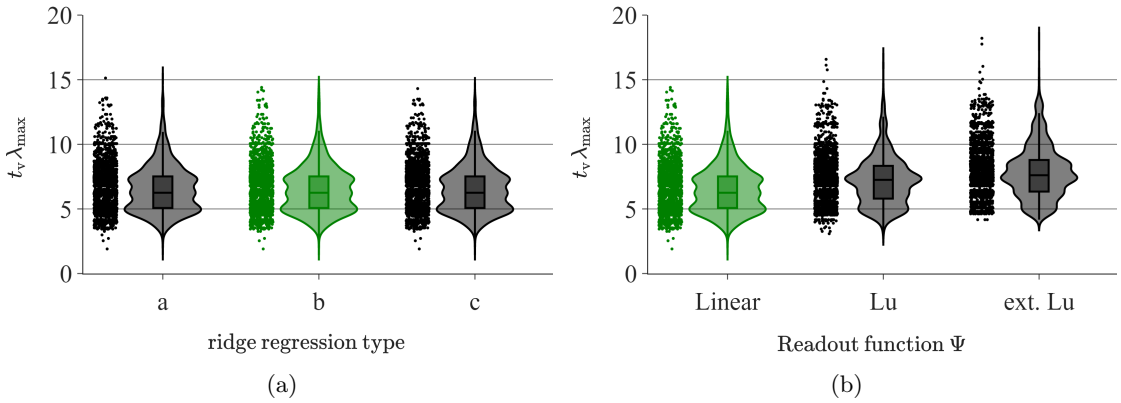


Figure 4.4: Violin plots for the ensemble of $n_R \times n_T \times n_P = 2250$ valid time calculations according to the ensemble experiment, defined in Sec. 4.4, applied to the Lorenz system. The default RC setup is shown in green. The box in the middle of the violin shows the median value, and lower and upper quartile. (a) Valid times for the different ridge regression types *a*, *b*, and *c* defined in Sec. 3.2. (b) Valid times for the nonlinear readout functions, defined in Sec. 4.3.

Hyperparameter sweeps Fig. 4.5 illustrates how variations in the hyperparameter values affect the median value of the ensemble of valid times. The error bars, which represent the lower and upper quartiles, indicate the valid times for which 25% of the data points fall below, and 75% fall above, respectively. For each hyperparameter sweep, all other hyperparameter values are fixed according to the default values listed in Tab. 4.2. The purpose of this paragraph is to provide a simple understanding of the influence of the hyperparameters, and to justify the chosen default hyperparameter values. These hyperparameters will be discussed in more depth through the lens of principal components in the following chapter.

As seen in Fig. 4.5a, a sufficiently large reservoir dimension, r_{dim}, is necessary for achieving a high prediction quality. As the reservoir dimension is increased, there is a steep increase in valid times, however, this improvement saturates for larger dimensions: A reservoir dimension of $r_{dim} = 2000$ only yields a small improvement compared to $r_{dim} = 1000$, but there is a significant increase when moving from $r_{dim} = 200$ to $r_{dim} = 400$.

Fig. 4.5b demonstrates the influence of the regularization parameter, β, on the valid times. It can be observed that an initial decrease of the regularization parameter from $\beta = 10^{-1}$ to $\beta = 10^{-8}$ significantly increases the valid times. However, any further reduction in the regularization parameter reduces the valid times again. As described in chapter 3, a low regularization parameter leads to larger values in the output matrix, $\mathbf{W}_{out}$, which relates to overfitting the training data. The large values in the output matrix can lead to unstable predictions, resulting in lower valid times.

The influence of the input strength, σ, is illustrated in Fig. 4.5c. It can be seen that there is a sharp increase in the prediction accuracy when moving from $\sigma = 0.2$, representing the first data point that is shown, to $\sigma = 1.0$. However, for larger input strengths, $\sigma > 2.5$, the prediction quality starts to decrease again. The poor prediction quality for small input strengths can be explained by the fact that the input strength controls the range of values for the arguments of the tanh activation function. For small arguments, $|x| \approx 0$, the hyperbolic tangent function is approximately linear: $\tanh(x) \approx x$. However, a certain degree of nonlinearity is necessary for good prediction capabilities, as was also discussed in [27].

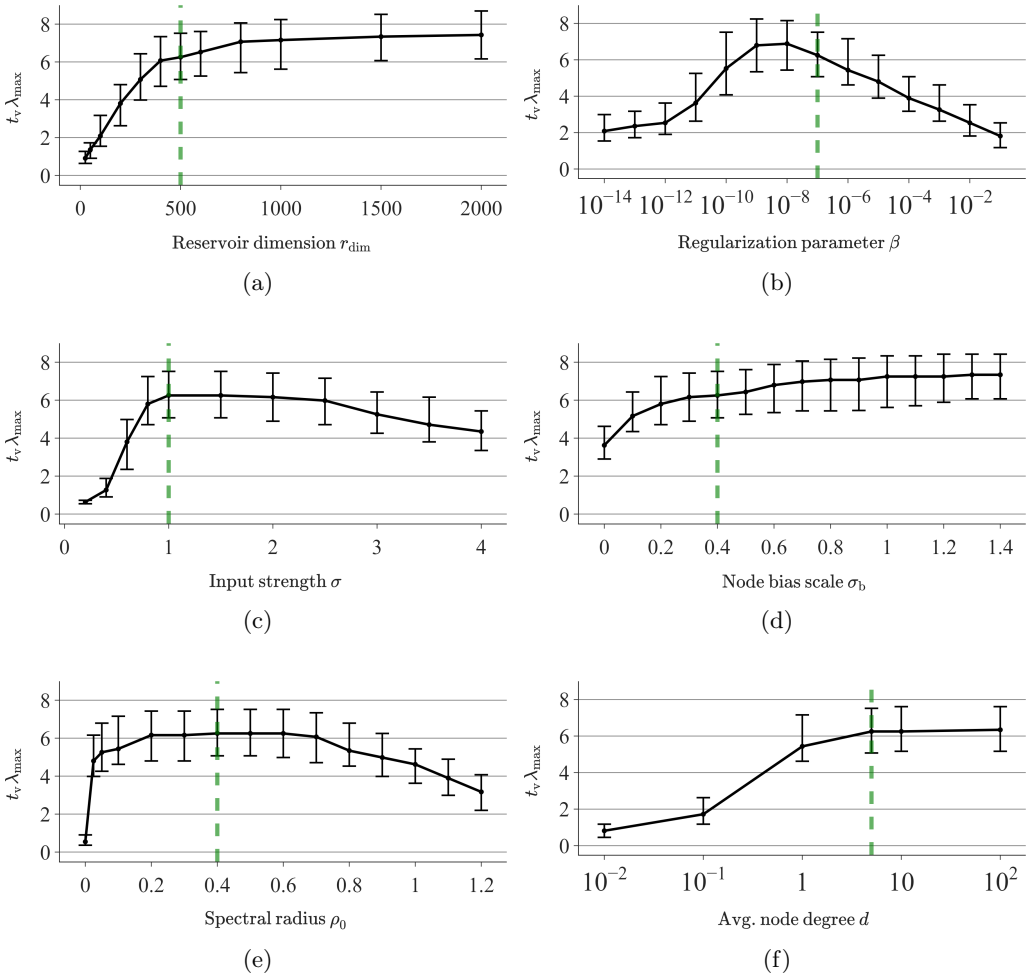


Figure 4.5: Median value with lower and upper quartile error bars for the ensemble of $n_R \times n_T \times n_P = 2250$ valid time calculations for the Lorenz system, under variations of hyperparameters. The green dashed line indicates the default hyperparameter choice.

Fig. 4.5d demonstrates the positive impact of including a node bias vector, $\boldsymbol{b}$, into the default RC setup: When the default node bias is added with $\sigma_b = 0.4$, the valid time that corresponds to no node bias, $\sigma_b = 0$, is almost doubled. An increase in the node bias to $\sigma_b = 1.4$ further improves the prediction quality, but the effect is relatively small.

The figures, Fig. 4.5e and Fig. 4.5f, demonstrate the impact of the spectral radius, ρ_0, and average node degree, d, of the reservoir network, $\mathbf{A}$, respectively. In Fig. 4.5e, which shows the spectral radius, it can be observed that there are comparable valid times across a wide range of values, $\rho_0 \in [0.2, 0.7]$. However, in the case where the spectral radius is set to $\rho_0 = 0$, effectively removing the reservoir network, $\mathbf{A} = 0$, the prediction capabilities of the reservoir are nullified, as shown by the leftmost data point. A small increase in the spectral radius to $\rho_0 = 0.25$, as represented by the second data point, already leads to a substantial increase in valid times. At the other end of the spectrum, when the spectral radius exceeds $\rho_0 > 0.7$, the prediction quality begins to decrease again. As shown in Fig. 4.5f, a variation of the average node degree, d, from an extremely small value, $d = 0.01$, to the default value, $d = 5.0$, increases the the valid times substantially. Interestingly, there is no difference in performance between $d = 5$ and $d = 100$.

Chapter 5

Towards Explainable RC using Principal Components

In the previous chapter, the software-based implementation of reservoir computing for the task of time series prediction was introduced. It was shown how the input, $\boldsymbol{x}(t)$, is embedded in the high-dimensional reservoir states, $\boldsymbol{r}(t)$, through the use of the recurrent update function, Eq. (4.24), whose parameters, $\mathbf{W}_{\text{in}}$, $\mathbf{A}$, and $\boldsymbol{b}$, are randomly generated and fixed throughout training. It was demonstrated that variations in the hyperparameter values that control the random generation of these RC parameters can result in better or worse predictions for the Lorenz system. Despite extensive research on determining which RC parameters lead to optimal performance, such as in [21, 25, 27, 28, 29, 30, 31], the reservoir states, $\boldsymbol{r}(t)$, remain a black box with limited interpretability. This chapter aims to improve the interpretability of RC through the use of a principal component analysis on the reservoir states, $\boldsymbol{r}(t)$. Due to the tight connection between principal components and ridge regression, as described in chapter 3, the PCA not only provides a tool to investigate the dynamics of the reservoir, but also allows for an in-depth analysis of the RR output-learning. In a previous study, Gallicchio et al. [9] also used PCA as a tool to analyze the reservoir states. They demonstrated a high level of redundancy in the reservoir states when driven with a discrete one-dimensional time series consisting of 10 possible values. In addition, they found an interesting match between the first few principal components and the most recent reservoir inputs. In [32], the authors used PCA as a dimensionality reduction technique for the reservoir states before performing ridge regression, which results in additional regularization. The work presented here can be considered an extensions of the work in [9], incorporating an analysis of ridge regression using the Lorenz system as data.

5.1 PCA-procedure for reservoir computing

The theory developed in section 3.3, which allows for an interpretation of type-b ridge regression through the principal components of the RR input variables, $\boldsymbol{k}$, is now applied to reservoir computing. In RC, the RR input variables, $\boldsymbol{k}$, can either be given by the pure reservoir states, $\boldsymbol{k} = \boldsymbol{r}$, or by a nonlinear readout function of the latter, $\boldsymbol{k} = \psi(\boldsymbol{r})$, as described in section 4.3. In the following, only the case where the RR input variables are given by the pure reservoir states, $\boldsymbol{k} = \boldsymbol{r}$, is considered, enabling a direct analysis of the reservoir states through the principal components. The PCA-procedure for reservoir computing can be defined by the following steps:

PCA-procedure:

1. **Train RC:**
   Perform standard RC training by conducting type-b ridge regression on the set of N_{T} reservoir states, $\mathbf{R}$, and output data, $\mathbf{Y}$, as outlined in section 4.2. This results in the type-b RR solutions given by the output matrix, $\mathbf{W}_{\text{out}}$, and intercept, $\boldsymbol{w}$.

2. **Perform PCA:**
   Calculate the principal components, $\mathbf{P}$, and explained variances, $\mathbf{\Phi}$, of the collected reservoir states, $\mathbf{R}$, by diagonalizing the covariance matrix, $\mathbf{S}$, of the centered reservoir states, $\mathbf{R}_\mathrm{c}$:

$$\mathbf{S} = \mathbf{R}_\mathrm{c}\mathbf{R}_\mathrm{c}^T/N = \mathbf{P}^T\mathbf{\Phi}\mathbf{P} \tag{5.1}$$

   The N_T centered reservoir states, $\boldsymbol{r}_\mathrm{c}[j]$, in $\mathbf{R}_\mathrm{c}$ are given by:

$$\boldsymbol{r}_\mathrm{c}[j] = \boldsymbol{r}[j] - \overline{\boldsymbol{r}}, \qquad \text{with } \overline{\boldsymbol{r}} = \frac{1}{N_\mathrm{T}}\sum_j^{N_\mathrm{T}} \boldsymbol{r}[j] \tag{5.2}$$

   Once the principal components, $\mathbf{P}$, and mean of reservoir states, $\overline{\boldsymbol{r}}$, are obtained, the reservoir states can be transformed onto the PC coordinate system by:

$$\boldsymbol{r}_\mathrm{pc}(t) = \mathbf{P}(\boldsymbol{r}(t) - \overline{\boldsymbol{r}}) \tag{5.3}$$

3. **RC output through PCs:**
   The reservoir output, Eq. (4.26), can now be transformed to act on the PC-transformed reservoir states, $\boldsymbol{r}_\mathrm{pc}(t)$:

$$\boldsymbol{y}_\mathrm{r}(t) = \mathbf{W}_\mathrm{out}\boldsymbol{r}(t) + \boldsymbol{w} \quad \overset{\text{PC}}{\Longrightarrow} \quad \boldsymbol{y}_\mathrm{r}(t) = \mathbf{W}_\mathrm{pc}\boldsymbol{r}_\mathrm{pc}(t) + \overline{\boldsymbol{y}}, \qquad \text{with } \mathbf{W}_\mathrm{pc} = \mathbf{W}_\mathrm{out}\mathbf{P}^T \tag{5.4}$$

   The PC-transformed output matrix, $\mathbf{W}_\mathrm{pc}$, now acts on the principal components rather than on the original reservoir nodes, as was previously the case for $\mathbf{W}_\mathrm{out}$. The regularization imposed by the RR regularization parameter, β, can now be interpreted directly in terms of the principal components:

$$\mathbf{W}_\mathrm{pc} = \mathbf{W}_{\mathrm{pc},\beta=0} \cdot \mathrm{diag}\left(f(\phi_i)\right), \qquad \text{with } f(\phi_i) = \frac{\phi_i}{\phi_i + \phi_\mathrm{co}} \tag{5.5}$$

   The cutoff explained variance, ϕ_co, is given by $\phi_\mathrm{co} = \beta/N_\mathrm{T}$. The PC-filter, $f(\phi_i) \in (0, 1]$, leads to a suppression of output contributions belonging to principal components, i, with small explained variances, $\phi_i < \phi_\mathrm{co}$. The resulting number of active principal components in the reservoir output, Eq. (5.4), can be estimated by the PC-cutoff, i_co. This cutoff is represented by the PC index, i, where $f(\phi_i) \approx 0.5$, or equivalently, where $\phi_i \approx \phi_\mathrm{co}$.

4. **Variance of $\boldsymbol{r}_\mathbf{pc}$ during prediction:**
   Thus far, the steps described have only taken into account quantities generated during the RC training phase. Additionally, the reservoir states, $\boldsymbol{r}(t)$, generated during the prediction phase can also be PC-transformed using $\mathbf{P}$ and $\overline{\boldsymbol{r}}$ from the training phase, as per Eq. (5.3).

   A comparison of the variances of the PC-transformed reservoir states, $\boldsymbol{r}_\mathrm{pc}(t)$, generated when driven with the true system during training, and those generated during the prediction phase when the reservoir is run in a closed-loop configuration, can be used to assess the prediction quality of the reservoir. To remind, the variances of $r_{\mathrm{pc},i}(t)$ generated during the training phase are simply the explained variances, ϕ_i:

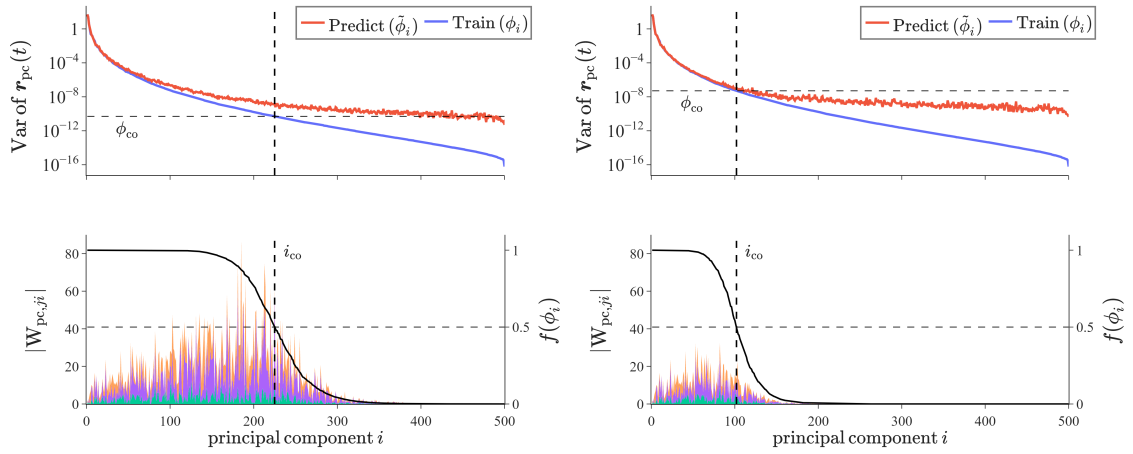
$$\phi_i = \frac{1}{N_\mathrm{T}}\sum_j^{N_\mathrm{T}} (r_{\mathrm{pc},i}[j] - \langle r_{\mathrm{pc},i}\rangle_\mathrm{T})^2 \tag{5.6}$$

   Similarly, the variances of $r_{\mathrm{pc},i}(t)$ generated during the prediction phase, denoted as $\tilde{\phi}_i$, can be obtained by:

$$\tilde{\phi}_i = \frac{1}{N_\mathrm{P}}\sum_j^{N_\mathrm{P}} (r_{\mathrm{pc},i}[j] - \langle r_{\mathrm{pc},i}\rangle_\mathrm{P})^2 \tag{5.7}$$

It is noteworthy that while the mean of $\boldsymbol{r}_{\mathrm{pc}}(t)$ over the training samples vanishes, $\langle r_{\mathrm{pc},i}\rangle_{\mathrm{T}} = 0$, the mean of $\boldsymbol{r}_{\mathrm{pc}}(t)$ over the prediction samples may not be strictly zero, $\langle r_{\mathrm{pc},i}\rangle_{\mathrm{P}} \neq 0$.

Demonstration In this paragraph, the PCA-procedure for RC will be demonstrated by displaying the explained variances, ϕ_i, and prediction variances, $\tilde{\phi}_i$, as well as the PC-transformed output matrix, $\mathbf{W}_{\mathrm{pc}}$, that is regularized by the PC-filter, $f(\phi_i)$. To do so, the same reservoir realization of the default setup is once trained with the default regularization parameter of $\beta = 10^{-7}$, and once with a stronger regularization of $\beta = 10^{-4}$. The experiment corresponds to the first iteration, $i_{\mathrm{R}} = i_{\mathrm{T}} = i_{\mathrm{P}} = 1$, of the ensemble experiment described in section 4.4. Corresponding to the default experiment parameters, Tab. 4.1, the reservoir is trained and predicted for $N_{\mathrm{T}} = N_{\mathrm{P}} = 2000$ time steps, using the Lorenz system as the data. The results are depicted in Fig. 5.1.



(a) $\beta = 10^{-7}$ (default)

(b) $\beta = 10^{-4}$

Figure 5.1: Explained variances, ϕ_i, and prediction variances, $\tilde{\phi}_i$ (upper plots), as well as the PC-transformed output matrix, $\mathbf{W}_{\mathrm{pc}}$, and PC-filter, $f(\phi_i)$ (lower plots), for the Lorenz system prediction using one realization of the default RC setup. The left plots, (a), use the default regularization parameter, $\beta = 10^{-7}$, and the right plots, (b), a stronger regularization of $\beta = 10^{-4}$. The PC-transformed output matrix is visualized as a stacked area plot of its absolute values, $|\mathbf{W}_{\mathrm{pc}}|$, where the three colors indicates the three output dimensions. The PC-cutoff, i_{co}, where $f(\phi_i) = 0.5$, or equivalently, where $\phi_i = \phi_{\mathrm{co}}$, is shown as a vertical dashed line.

The explained variances, ϕ_i, which are not affected by the regularization parameter, β, are shown in the upper plots in Fig. 5.1 using a logarithmic y-axis. They show a rapid decline when moving to higher principal components, indicating strong correlations in the reservoir states, which confirms the findings in [9]. The explained variance of the 50th PC is already as low as $\phi_{50} \approx 10^{-5}$, demonstrating that a small number of principal components explains almost all the variance in the reservoir states. The EV curves in the semi-logarithmic plot appear to converge to a linear form, indicating an exponential decrease in the explained variances, ϕ_i, in that range. Interestingly, the prediction variances, $\tilde{\phi}_i$, match the explained variances, ϕ_i, very accurately up to $i \approx 100$ for both cases of the regularization parameter. However, for higher components, clear differences between the two variances can be observed. Here, it would be interesting to examine the relationship between the degree of matching between $\tilde{\phi}_i$ and ϕ_i and the corresponding prediction qualities.

The lower plots in Fig. 5.1 demonstrate the effect of regularization in ridge regression. In the case of stronger regularization, $\beta = 10^{-4}$, the PC-transformed output matrix, $\mathbf{W}_{\mathrm{pc}}$, incorporates fewer principal components, compared to weaker regularization with $\beta = 10^{-7}$. This behaviour can be directly explained by the PC-filters, $f(\phi_i) \in (0, 1]$, which are also shown in the lower plots: The approximately exponential decrease in explained variances, ϕ_i, results in the PC-filter, $f(\phi_i)$, to resemble a softened step-function that goes from 1 to 0. Contributions to $\mathbf{W}_{\mathrm{pc}}$ on the right side of the softened step-function are suppressed to 0, according to Eq. (5.5). The location of the step is given by the PC-cutoff, i_{co}, which represents the principal component, i, where $f(\phi_i) = 0.5$, or equivalently, where $\phi_i = \phi_{\mathrm{co}}$. When the regularization parameter, β, is increased, the location of

the PC-cutoff, i_{co}, moves to the left, to the point where the horizontal line given by $\phi_{\text{co}} = \beta/N_{\text{T}}$ intersects the explained variance curve, ϕ_i. The width of the step-function, $f(\phi_i)$, is related to the decrease in explained variances, ϕ_i, on either sides of the PC-cutoff, i_{co}. A stronger decrease in ϕ_i leads to a sharper step in $f(\phi_i)$, while a flatter EV-curve results in a softer step. In the analysis that follows, the number of active principal components will be estimated by the PC-cutoff, i_{co}, regardless of the softness of the step in the PC-filter, $f(\phi_i)$.

5.2 Explained variances, active components and prediction quality

This section investigates the impact of variations in the RC hyperparameters on the reservoir's explained variances, ϕ_i. Additionally, the relationship between the resulting number of active components, as measured by the PC-cutoff, i_{co}, and the prediction quality, as measured by the valid times, t_{v}, is analyzed.

To obtain accurate statistical results, the ensemble experiment outlined in Sec. 4.4 is employed to calculate the explained variances, ϕ_i, corresponding PC-cutoffs, i_{co}, and valid times, t_{v}, for an ensemble of n_{R} reservoir realizations, n_{T} training-sections, and n_{P} prediction-sections. As the EVs, ϕ_i, and PC-cutoffs, i_{co}, can be obtained from the training phase alone, only the calculation of the valid times, t_{v}, requires the additional n_{P} prediction-sections. The final datapoints and error spans for EVs, ϕ_i, and PC-cutoffs, i_{co}, correspond to the median value and lower/upper quartile of the $n_{\text{R}} \times n_{\text{T}} = 225$ ensemble calculations, respectively. Similarly, the valid time datapoints and errors are also formed from the median and lower/upper quartile across all $n_{\text{R}} \times n_{\text{T}} \times n_{\text{P}} = 2250$ ensemble calculations. As in all previous experiments, all hyperparameter values which are not specifically changed are given by the values in Tab. 4.2, and the time series data is provided by the Lorenz system.

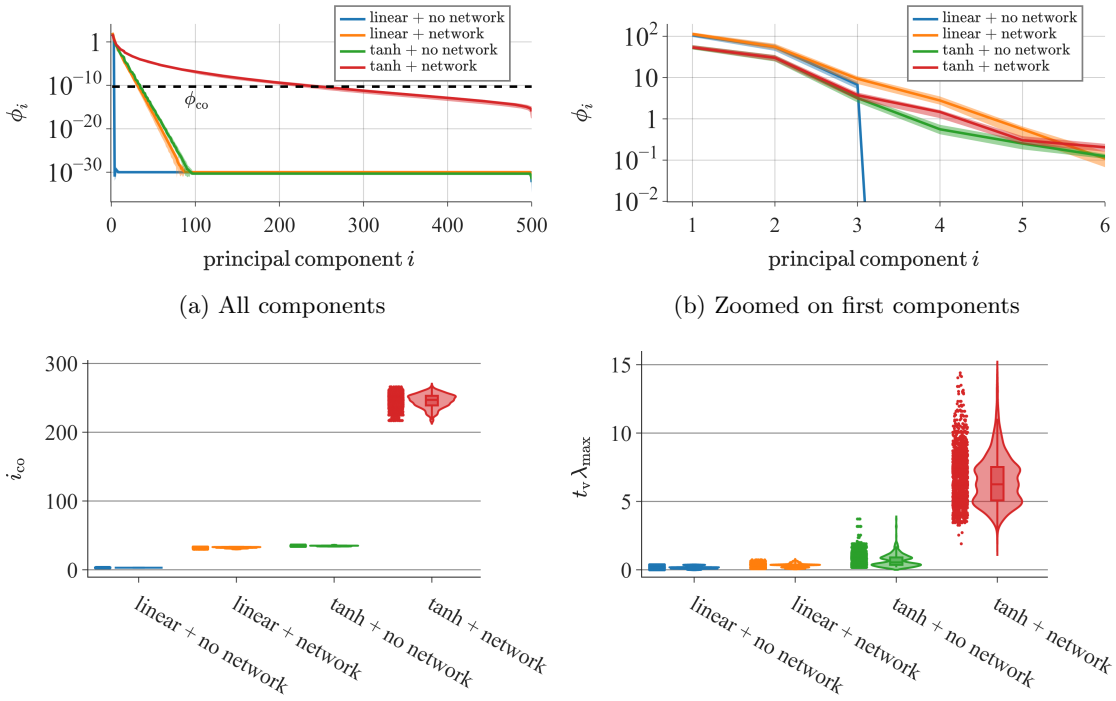
The first experiment investigates the scenarios in which either the tanh-activation function or the reservoir network, $\mathbf{A}$, are individually removed, as well as the scenario in which both are removed from the reservoir update equation, Eq. (4.24). The subsequent experiments examine individual variations in the hyperparameters: reservoir dimension, r_{dim}, input strength, σ, spectral radius, ρ_0, average node degree, d, and regularization parameter, β. In all experiments except for the final one, which investigates the influence of the regularization parameter, the regularization parameter is fixed at its default value of $\beta = 10^{-7}$, resulting in an EV-cutoff of $\phi_{\text{co}} = 10^{-7}/2000 = 5.0 \cdot 10^{-11}$.

Activation function and reservoir network: Fig. 5.2a demonstrates the variations in explained variances, ϕ_i, depending on the use of the tanh activation function, labeled as "linear" or "tanh", and the presence of the reservoir network, $\mathbf{A}$, labeled as "network" or "no network". The second plot, Fig. 5.2b, is equivalent to the first but is zoomed in on the first six principal components. The third and forth plots, Fig. 5.2c and Fig. 5.2d, show the distribution of the PC-cutoffs, i_{co}, and valid times, t_{v}, respectively, through the use of violin plots.

An interesting initial observation of all explained variance curves is their extremely narrow error bands, which represent where 50% of the data points fall. This demonstrates a very small variability of ϕ_i across the ensemble of $n_{\text{R}} \times n_{\text{T}} = 225$ different combinations of reservoir realizations and training-sections. The EV curve for the case where both the reservoir network, $\mathbf{A}$, and the tanh activation function are removed, drops below machine precision after the first three principal components. This corresponds to the expected behaviour, as the resulting reservoir update function,

$$\boldsymbol{r}(t) = \mathbf{W}_{\text{in}}\hat{\boldsymbol{x}}(t) + \boldsymbol{b}, \tag{5.8}$$

is just a linear function of the standardized input, $\hat{\boldsymbol{x}}(t)$. As the input, $\boldsymbol{x}(t)$, provides three dimensions which are not collinear, any linear function of it can also contain at most three non-collinear dimensions. By either incorporating nonlinearity through the use of the tanh activation function, labeled as "tanh + no network", or enabling memory through the addition of the reservoir network, labeled as "linear + network", the EV curves no longer drop below machine precision after the first



(a) All components

(b) Zoomed on first components

(c) PC-cutoff, violin plot

(d) Valid time, violin plot

Figure 5.2: Explained variances, ϕ_i, PC-cutoffs, i_{co}, and valid times, $t_v \lambda_{max}$, for the four cases: "linear + no network", "linear + network", "tanh + no network", and "tanh + network".

three principal components. Interestingly, both curves show a similar, near-perfect exponential decrease, which corresponds to a near-perfect line on the logarithmic y-axis. This trend continues until they both fall below machine precision after around 90 principal components. In contrast, when both the nonlinearity and memory are enabled, as is the case in the default RC setup labeled "tanh + network", the resulting reservoir states contain far less multicollinearity, as indicated by the much broader explained variance curve. As described in the previous section, the intersection of the horizontal line given by the EV-cutoff, ϕ_{co}, and the explained variance curve, ϕ_i, defines the PC-cutoff, i_{co}, which measures the number of active components in the reservoir output. As a result, the stronger declines in the modified RC setups lead to far less active components, compared to the standard RC setup, as is visualized in Fig. 5.2c. Additionally, the only setup capable of producing accurate predictions is the standard RC setup that includes both nonlinearity and memory, as evident from the distribution of valid times shown in Fig. 5.2d. This raises the question of whether the number of active components obtained from the training phase can be used as an early indicator of prediction quality. The following experiments, which analyze the impact of variations in the RC hyperparameters, will investigate this question in greater depth.

Reservoir dimension r_{dim}: The left plot in Fig. 5.3 shows the explained variances, ϕ_i, for seven different reservoir dimensions, ranging from $r_{dim} = 100$ to $r_{dim} = 1500$. Since the total number of principal components is equal to the reservoir dimension, $i_{max} = r_{dim}$, each curve only extends up to its related r_{dim}. The right plot depicts the valid times, t_v, against PC-cutoffs, i_{co}, with the colors of the points indicating the reservoir dimensions, r_{dim}. An additional five reservoir dimensions are considered in this plot.

All explained variance curves share a similar shape, characterized by a steep drop for small principal components, followed by a relatively flat region that appears linear on a logarithmic y-axis, and ending with another stronger drop as i approaches r_{dim}. While the initial steep drops appear to be similar for all reservoir dimensions, the following flatter declines are slightly stronger for smaller reservoir dimensions. The initial similarities in the curves can persist for many principal components, as is demonstrated by comparing $r_{dim} = 1000$ and $r_{dim} = 1500$, which are nearly identical up to $i \approx 500$. Interestingly, due to the initial decline being relatively similar for all

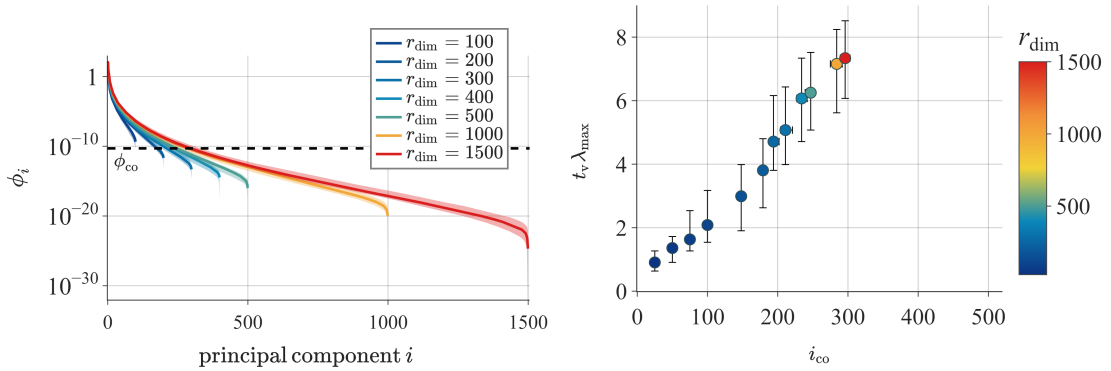


Figure 5.3: Reservoir dimensions, $r_{\mathrm{dim}} \in (25, 50, 75, 100, 150, 200, 250, 300, 400, 500, 1000, 1500)$.

reservoir dimensions, the range of different PC-cutoffs, i_{co}, resulting from the intersections of ϕ_i with ϕ_{co} is small compared to the range of reservoir dimensions tested: For small reservoir dimensions, such as $r_{\mathrm{dim}} = 100$, the minimal explained variance is still above the EV-cutoff, $\phi_{r_{\mathrm{dim}}} > \phi_{\mathrm{co}}$, so the RR regularization has no effect and the number of active components equals the reservoir dimension, $i_{\mathrm{co}} = r_{\mathrm{dim}}$. On the other hand, transitioning from a large to an even larger reservoir dimension, the initial explained variances up to ϕ_{co} remain unchanged, as shown by $r_{\mathrm{dim}} = 1000$ and $r_{\mathrm{dim}} = 1500$, both of which have a similar PC-cutoff of $i_{\mathrm{co}} \approx 300$. As a result, with all other hyperparameters being fixed, the number of active components saturates for large reservoir dimensions.

The plot on the right in Fig. 5.3 displays a remarkable nearly linear correlation between the number of active components (measured by i_{co}) and the prediction quality (measured by t_{v}) for all tested reservoir dimensions. The saturation in valid times with increasing reservoir dimensions, previously shown in Fig. 4.5a, aligns well with the saturation in the number of active components discussed here. This interesting result supports the previous conjecture that the number of active components derived solely from the combination of driven reservoir states, $\mathbf{R}$, and EV-cutoff, $\phi_{\mathrm{co}} = \beta / N_{\mathrm{T}}$, serves as an indicator for the prediction quality. Moreover, the matching saturations in the number of active components and in the prediction quality demonstrate that increasing the reservoir dimension only remains effective at a certain point if the EV-cutoff is lowered accordingly to enable more active components. Due to the low variability in the explained variances, ϕ_i, across reservoir realizations and training-sections, an accurate statistical estimation of the resulting number of active components, i_{co}, requires far less computational power compared to accurately estimating the median value of the valid times. This can be observed from the very narrow (hardly visible) error bars for i_{co} compared to the larger error bars for the valid times, t_{v}. Therefore, an investigation of the number of active components, i_{co}, using only a few reservoir realizations and training-sections might provide a fast estimation of the reservoir dimension, r_{dim}, where the valid times begin to saturate. However, this procedure only works if all other hyperparameters, including the regularization parameter β, are kept fixed.

Input strength σ: The plots in Fig. 5.4 show the effect of input strength, ranging from $\sigma = 0.1$ to $\sigma = 4.0$, on the explained variances, ϕ_i, (left plot) and on the t_{v} vs. i_{co} scatter plot (right plot). The EV curves display only five input strengths for easier visualization, while the scatter plot on the right shows 14 values for more detailed results.

Contrary to the previous scenario where the initial drop in the explained variance curves was similar across different reservoir dimension, r_{dim}, in this case, the initial drop for lower input strengths is more pronounced than for higher input strengths. Interestingly, after the initial drop in ϕ_i, the flatter regions of the plots show a similar slope for the various input strengths, corresponding to an approximately equivalent exponential rate of decline in ϕ_i. The lower overall explained variances for smaller input strengths indicate higher correlations among the nodes of the reservoir states, $\boldsymbol{r}(t)$, compared to higher input strengths. This can be related to the input strength controlling the amount of nonlinearity induced in the reservoir: The input strength, σ, controls the size of the elements in the input matrix, $\mathbf{W}_{\mathrm{in}}$. The combination of input matrix, $\mathbf{W}_{\mathrm{in}}$, reservoir network,

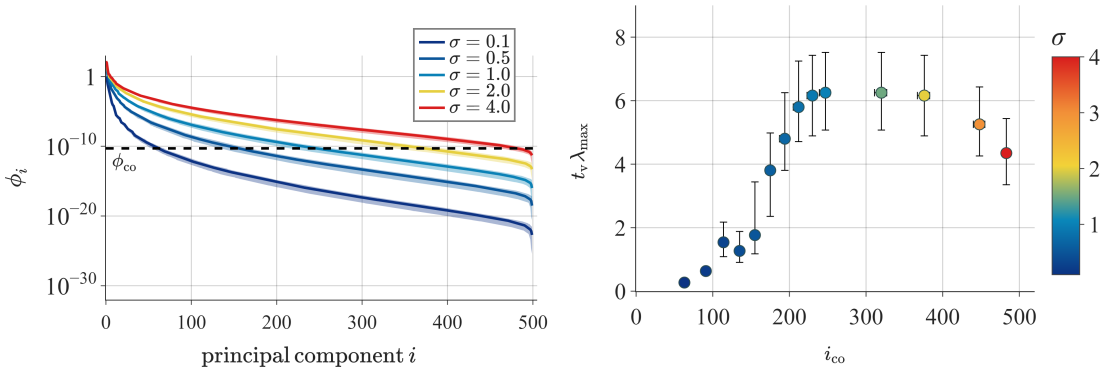


Figure 5.4: Input strengths, $\sigma \in (0.1, 0.2, \ldots, 0.9, 1.0, 1.5, 2.0, 3.0, 4.0)$.

A, and node bias, $\boldsymbol{b}$, determines the overall size of the arguments passed to the nonlinear tanh activation function in the reservoir update equation, Eq. (4.24). With the selected spectral radius of the reservoir network, $\rho_0 = 0.4$, and the scale of the node bias, $\sigma_b = 0.4$, being not too large, a smaller input strength moves the arguments of the activation function closer to its linear regime, $\tanh(x) \approx x$ for $|x| \ll 1$. The lower extent of nonlinearity induced in the reservoir states results in higher correlations, as reflected by their lower explained variance curves, compared to the case of larger input strengths. This results in fewer active components, i_{co}, for smaller input strengths, σ, and in more active components for larger values of σ. For instance, a small input strength of $\sigma = 0.1$ results in only $i_{co} \approx 60$ active components, while the default value of $\sigma = 1.0$ leads to $i_{co} \approx 250$. For the lower end of input strengths tested, $\sigma \in [0.1, 1.0]$, the resulting number of active components, i_{co}, correlates well with the prediction qualities, as seen from the right plot in Fig. 5.4. As the input strength increases beyond $\sigma > 1.0$, the number of active components, i_{co}, also increases, but the corresponding valid times start to decrease again. The decrease in valid times for larger input strengths was previously demonstrated in Fig. 4.5c. Therefore, compared to the results discussed in the previous experiment for varying reservoir dimensions, r_{dim}, the correlation between the number of active components and prediction quality is only observed at the lower end of the active components in this case.

Spectral radius ρ_0: The plots in Fig. 5.5 show the impact of the reservoir network's spectral radius, ranging from $\rho_0 = 0.0$ to $\rho_0 = 1.0$, on the explained variances, ϕ_i, (left plot) and on the t_v vs. i_{co} scatter plot (right plot). The spectral radii tested, apart from the extreme case of $\rho_0 = 0.0$, were varied by adjusting the exponent, x, in $\rho_0 = 10^x$, uniformly from $x = -4$ to $x = 0$ with a step size of 1 for the EV plots and a smaller step size of 0.5 for the PC-cutoff vs. valid times plot.

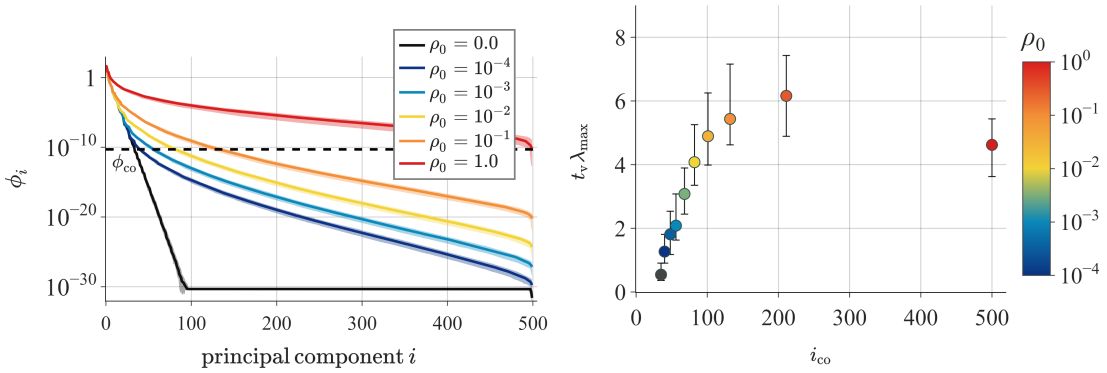


Figure 5.5: Spectral radii, $\rho_0 \in (0.0, 10^{-4}, 10^{-3.5} \ldots, 10^{-0.5}, 1.0)$.

The explained variances, ϕ_i, demonstrate the strong exponential decay in the case of a vanishing spectral radius, $\rho_0 = 0$, as already observed in the "tanh + no network" case in figure 5.2a. Interestingly, for increasing spectral radii, $\rho_0 > 0$, the EV curves still follow the line corresponding

to $\rho_0 = 0$ for small principal components, but eventually deviate into the previously seen flatter regime. Smaller spectral radii result in a later deviation from the $\rho_0 = 0$ curve. The larger explained variances, ϕ_i, for larger spectral radii, ρ_0, can be explained as follows: The spectral radius, ρ_0, regulates the influence of the reservoir network, $\mathbf{A}$, and hence the influence of past inputs, on the reservoir dynamics, through the term $\mathbf{A}\boldsymbol{r}(t - \Delta t)$ in Eq. (4.24). A larger spectral radius, which increases the memory of the reservoir, leads to more independent dynamics in the reservoir states, $\boldsymbol{r}(t)$, and therefore, higher overall explained variances, ϕ_i, compared to a smaller spectral radius. Additionally, as all EV curves display a similar behavior for smaller principal components regardless of the spectral radius, ρ_0, it suggests that these first principal components are mainly affected by the reservoir's immediate input, $\boldsymbol{x}(t)$. This finding was previously noted in [9] and will be further studied in section 5.3.

The plot on the right of Fig. 5.5 displays the relationship between the number of active components, i_{co}, and their corresponding valid times, t_{v}, resulting from the variations in the spectral radius, ρ_0. As in the previous cases for varying reservoir dimensions, r_{dim}, and input strengths, σ, a strong positive correlation can be observed for the lower end of active components. With a large spectral radius of $\rho_0 = 1.0$, the explained variances, ϕ_i, are so high that all the principal components act as active components, $i_{\text{co}} \approx r_{\text{dim}} = 500$. However, the prediction quality does not improve in proportion to the correlation seen for the lower end of active components and actually decreases. The decrease in valid times for larger spectral radii was already demonstrated in Fig. 4.5e. The steeper slope in the lower-end positive correlation between t_{v} and i_{co} compared to the previous cases with varying r_{dim} and σ reveals that the same number of active components, i_{co}, leads to varying prediction qualities, depending on the source of the active components. For instance, in the case of varying spectral radii the median valid time for $i_{\text{co}} \approx 100$ active components results in $t_{\text{v}}\lambda_{\max} \approx 5$, while the same number of active components in the case of varying input strengths yields only $t_{\text{v}}\lambda_{\max} \approx 1$. This shows that it is not just the number of active components, but also their effectiveness that matters.

Avg. node degree d: The plots in Fig. 5.6 show the effect of the reservoir network's average node degree, ranging from $d = 0.01$ to $d = 100$, on the explained variances, ϕ_i, (left plot) and on the t_{v} vs. i_{co} scatter plot (right plot). The node degrees tested were varied by adjusting the exponent, x, in $d = 10^x$, uniformly from $x = -2$ to $x = 2$ with a step size of 1 for the EV plots and a smaller step size of 0.5 for the PC-cutoff vs. valid times plot.

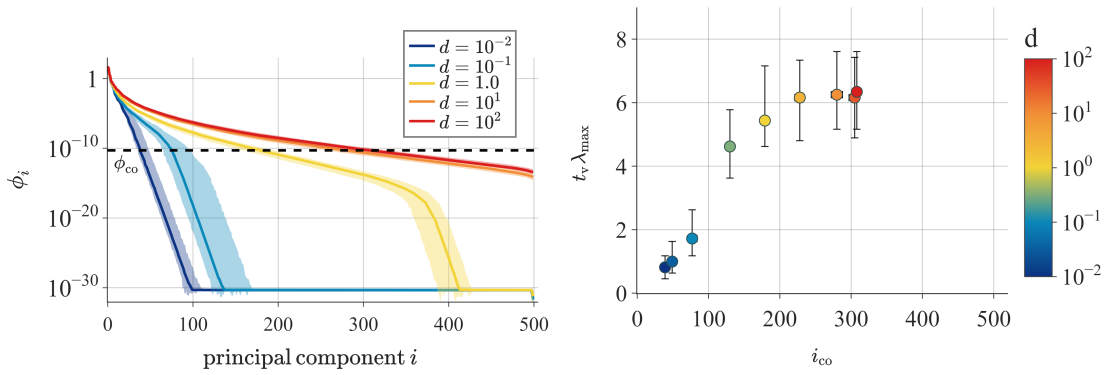


Figure 5.6: Avg. node degree, $d \in (10^{-2}, 10^{-1.5}, \ldots, 10^{1.5}, 10^{2})$.

As previously seen, the input strength, σ, primarily affects the slope of the initial EV curve drop for small principal components, while the spectral radius mainly determines the principal component where the initial drop ends and the middle flat region begins. The average node degree, d, considered here, appears to control the width of the middle flat region before the final steep exponential drop occurs: For the extremely low node degree of $d = 0.01$, the final EV drop occurs immediately after the initial drop, resulting in a similar curve of ϕ_i to the previous case of a removed reservoir network, $\mathbf{A} = 0$. This behavior can be explained by the fact that a node degree of $\rho_0 = 0.01$ leads, on average, to only $\rho_0 \cdot r_{\text{dim}} = 5$ non-zero elements in the reservoir network $\mathbf{A}$. Thus, the network's influence is negligible and the EV curve closely resembles the

"tanh + no network" case in Fig. 5.2a. The next two larger node degrees tested, $d = 0.1$ and $d = 1$, extend the flatter middle region to around $i \approx 70$ and $i \approx 350$, respectively, before the drop occurs. Interestingly, for $d = 10$ and $d = 100$, the EV curves are almost the same, and the hypothetical final EV drop appears to be beyond the limit of $i_{\max} = 500$. This suggests that at one point, increasing the node degree does not introduce new independent dynamics into the reservoir.

As in the previous cases, the number of active components, this time resulting from variations in the average node degree, d, aligns well with the corresponding valid times, as shown in the right plot of Fig. 5.6. Interestingly, the similar number of active components for the node degrees, $d = 10$, $d = 10^{1.5} \approx 32$, and $d = 100$, is reflected in the similar prediction qualities they produce.

Regularization parameter β: This final experiment considers variations in the ridge regression regularization parameter, β. Unlike in previous experiments, changes in β do not affect the explained variances, ϕ_i, but do alter the EV cutoffs according to: $\phi_{\text{co}} = \beta / N_{\text{T}}$. The left plot in Fig. 5.7 displays the explained variance curve of the default RC setup, along with the EV cutoffs, ϕ_{co}, resulting from seven different regularization parameters in the range $\beta = 10^{-13}$ to $\beta = 10^{-1}$. The right side of Fig. 5.7 presents the resulting i_{co} vs. t_{v} scatter plot using more detailed variations in the regularization parameter represented by the values of $\beta = 10^{-13}, 10^{-12}, \ldots, 10^{-2}, 10^{-1}$.

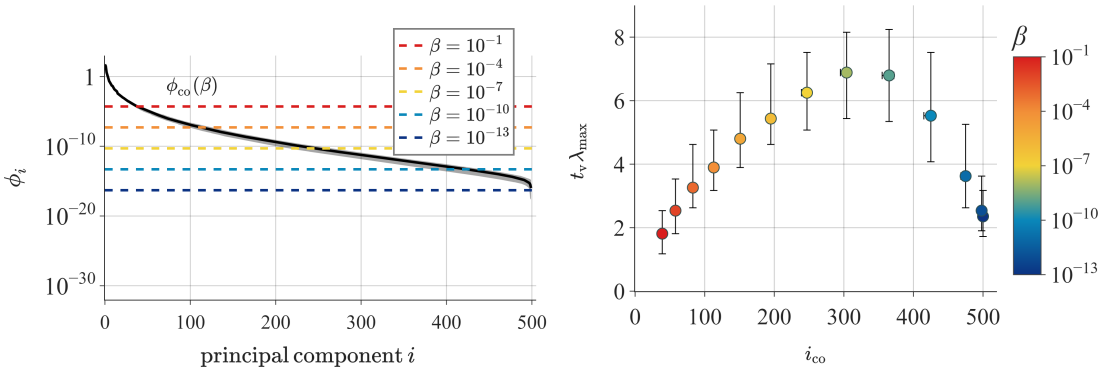


Figure 5.7: Regularization parameter, $\beta \in (10^{-13}, 10^{-12}, \ldots, 10^{-2}, 10^{-1})$.

As shown in the explained variance plot, different ϕ_{co} values intersect the EV curve at different points, allowing direct control of the active components, i_{co}, through the choice of the regularization parameter, β. Similar to the prior cases, up to $i_{\text{co}} \approx 300$, corresponding to $\beta = 10^{-8}$, the number of active components aligns well with the corresponding valid times, t_{v}. Even though a further decrease in the regularization parameter results in even more active components, the corresponding valid times decrease. The decrease in valid times for very small regularization parameters was already demonstrated in Fig. 4.5b, and can be related to overfitting the training data: When the output matrix, $\mathbf{W}_{\text{out}}$, is overly fine-tuned to the training data, the reservoir during the prediction phase, which by itself is a high-dimensional chaotic dynamical system, may converge to an attractor unrelated to the Lorenz system. Therefore, these potential instabilities in the closed-loop reservoir dynamics result in a lower prediction quality.

Conclusion The experiments presented in this section resulted in two interesting observations:

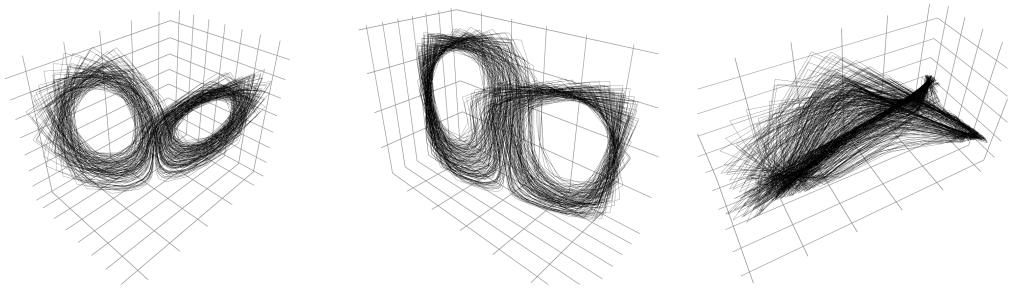
Firstly, the embedding of the input signal, $\boldsymbol{x}(t)$, into the high-dimensional reservoir states, $\boldsymbol{r}(t)$, was explored through the lens of the reservoir state's explained variances, ϕ_i. The explained variances, ϕ_i, provide a simple tool to explore the richness of the reservoir states [9], as they measure the amount of linear correlation within the reservoir's nodes: Smaller values indicate fewer independent dynamics, while larger values indicate more linearly independent dynamics in the reservoir states. It was demonstrated that only by combining memory and nonlinearity can the resulting reservoir states, $\boldsymbol{r}(t)$, achieve a significant amount of linearly independent dynamics, which is necessary for a successful prediction. A more detailed investigation of the explained variance curves, plotted on a logarithmic scale, revealed three approximately distinct regions: The first region, defined by the first few principal components, exhibits a steep initial decline in explained variance, followed by

a flatter decline in the broader middle region, and ending with another steep drop in the third region. Interestingly, the individual RC hyperparameters appear to directly control the properties of these regions: The input strength, σ, appears to determine the slope of the initial drop in the first region, with a smaller value of σ corresponding to a steeper decline. The spectral radius, ρ_0, affects the point at which the first region transitions into the flatter middle region, with a smaller value of ρ_0 resulting in a later transition. The combination of the reservoir dimension, r_{dim}, and average node degree, d, appears to determine the principal component at which the middle region transitions into the final drop, with smaller values of r_{dim} and d resulting in an earlier transition. In the case of the reservoir dimension, r_{dim}, this control is due to its definition of the total width of the EV curve. The overall slope of the broader middle region appears to be influenced by all the hyperparameters, σ, ρ_0, d, and r_{dim}, with smaller values in each leading to a stronger decrease. This corresponds to the expected behaviour, as larger values of σ result in more nonlinearity, larger values of ρ_0 and d result in stronger memory, and a larger reservoir dimension, r_{dim}, results in overall more available space for independent dynamics. Therefore, the explained variances have been demonstrated as an interesting tool for exploring the amount of independent dynamics in the reservoir generated by each hyperparameter.

The second interesting observation is the relationship between the number of active components, as measured by the PC-cutoff i_{co}, and the prediction quality, as measured by the valid times t_{v}. The number of active components, i_{co}, which is determined by the intersection between the explained variances, ϕ_i, and the EV-cutoff, $\phi_{\mathrm{co}} = \beta/N_{\mathrm{T}}$, measures the number of linearly independent dynamics in the reservoir states, $\boldsymbol{r}(t)$, that contribute to the output through the equation $\boldsymbol{y}_{\mathrm{r}}(t) = \mathbf{W}_{\mathrm{out}}\boldsymbol{r}(t) + \boldsymbol{w}$. The value of i_{co} can change by either varying the hyperparameters that control the reservoir update equation, Eq. (4.24), causing variations in the explained variances, ϕ_i, or by changing the value of the EV-cutoff, ϕ_{co}, through variations in the regularization parameter, β, or in the training size, N_{T}. Interestingly, it was demonstrated in all tested cases that a small number of active components results in a low prediction quality. Furthermore, it was shown that the prediction quality improves as the number of active components increases, regardless of the hyperparameter causing the increase in i_{co}. This positive correlation between i_{co} and t_{v} held for at least up to $i_{\mathrm{co}} \approx 250$ principal components in all tested cases. This experiment clearly shows that the number of active components is important for the prediction quality, but it does not imply that it is the only factor. This can be seen, as the positive correlations between i_{co} and t_{v} for the lower end of active components show different slopes depending on the hyperparameter considered. Nevertheless, an investigation of the number of active components appears to explain why some hyperparameter values perform well and others do not. This is particularly interesting because the estimation of the number of active components for a given set of hyperparameters can be done efficiently due to their low error and the fact that calculations do not need a prediction phase.

5.3 Meaning of principal components

In this section, the meaning of the principal component vectors, $\boldsymbol{p}_i$, and the PC-transformed reservoir states, $r_{\mathrm{pc},i}(t)$, is investigated through two simple experiments. In the previous study [9] the meaning of the PC-transformed states, $r_{\mathrm{pc},i}(t)$, was also investigated using a random one-dimensional input sequence with 10 possible values. The authors found that the states corresponding to the first PC, $r_{\mathrm{pc},1}(t)$, correlate strongly with the immediate input, the states of the second, $r_{\mathrm{pc},2}(t)$, with the input one step before, and so on, until the correlation diminishes. In this study, this behaviour is further investigated by using the Lorenz system as the reservoir's input, which includes the additional complications arising from a three-dimensional input. Fig. 5.8 shows the three-dimensional trajectories of the PC-transformed reservoir states generated during training, $[r_{\mathrm{pc},1}(t), r_{\mathrm{pc},2}(t), r_{\mathrm{pc},3}(t)]^T$ and $[r_{\mathrm{pc},4}(t), r_{\mathrm{pc},5}(t), r_{\mathrm{pc},6}(t)]^T$, alongside the training input provided by the Lorenz system. From the trajectory of the first three principal components, a slightly deformed Lorenz system trajectory can be recognized, whereas the following three principal component's trajectory shows a different shape. This already suggests a close relationship between the first x_{dim} principal components and the driving input time series $\boldsymbol{x}(t)$.



(a) Lorenz input (b) PCs, $i = 1, 2, 3$ (c) PCs, $i = 4, 5, 6$

Figure 5.8: Three-dimensional trajectories (a) of the reservoir input $\boldsymbol{x}(t)$ given by the Lorenz system, (b, c) of the PC-transformed reservoir states $r_{\mathrm{pc},i}(t)$ generated during training.

Network influence on $r_{\mathbf{pc},i}$ and $\boldsymbol{p}_i$ In this numerical experiment, the influence of the reservoir's memory, induced by the reservoir network, $\mathbf{A}$, on the individual PC-transformed reservoir states, $r_{\mathrm{pc},i}(t)$, and corresponding PC vectors, $\boldsymbol{p}_i$, is investigated. To do this, two RC configurations are compared: one with a non-zero reservoir network $\mathbf{A}$ given by the default RC setup with the spectral radius $\rho_0 = 0.4$, and the other using the same default RC setup with the network removed by setting $\rho_0 = 0$. Apart from the reservoir network, both RC setups are identical in terms of input matrix, $\mathbf{W}_{\mathrm{in}}$, and node bias, $\boldsymbol{b}$. As a result, the modified reservoir's update equation, Eq. (5.9), is solely dependent on the immediate standardized input $S(\boldsymbol{x}(t))$, thereby eliminating any influence of previous inputs:

$$\boldsymbol{r}(t) = \tanh(\mathbf{W}_{\mathrm{in}} S(\boldsymbol{x}(t)) + \boldsymbol{b}) \tag{5.9}$$

Eq. (5.9) corresponds to the "tanh + no network" case from the previous section 5.2. After training both reservoirs on the same Lorenz time series, the resulting PC vectors, $\boldsymbol{p}_i$ and $\hat{\boldsymbol{p}}_i$, and PC-transformed reservoir states, $r_{\mathrm{pc},i}(t)$ and $\hat{r}_{\mathrm{pc},i}(t)$, are compared, where $\hat{\boldsymbol{p}}_i$ and $\hat{r}_{\mathrm{pc},i}(t)$ correspond to the modified RC setup without network. By comparing $\boldsymbol{p}_i$ with $\hat{\boldsymbol{p}}_i$ and $r_{\mathrm{pc},i}(t)$ with $\hat{r}_{\mathrm{pc},i}(t)$, the aim is to identify any similarities between the two cases for some principal components i. This would suggest that these PCs are independent of the reservoir's memory, and are thus defined by the reservoir's input $\boldsymbol{x}(t)$ alone. The comparison is performed by using the Pearson correlation coefficient (PCC), $R(a, b) \in [-1, +1]$, between two sequences, a and b. The PCC is defined as:

$$R(a, b) = \frac{C(a, b)}{C(a, a)C(b, b)} \tag{5.10}$$

The function $C(a, b)$ denotes the usual covariance between two N-long sequences a and b:

$$C(a, b) = \sum_{k=1}^{N} (a[i] - \overline{a})(b[i] - \overline{b}) \tag{5.11}$$

For the comparison of the PC-transformed reservoir states, $r_{\mathrm{pc},i}(t)$ and $\hat{r}_{\mathrm{pc},i}$, the sequences a and b are given by:

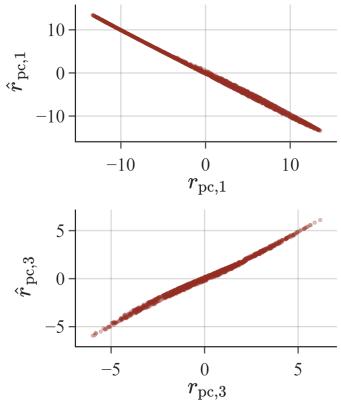
$$a = [r_{\mathrm{pc},i}[1], r_{\mathrm{pc},i}[2], \ldots, r_{\mathrm{pc},i}[N_{\mathrm{T}}]] \tag{5.12}$$
$$b = [\hat{r}_{\mathrm{pc},i}[1], \hat{r}_{\mathrm{pc},i}[2], \ldots, \hat{r}_{\mathrm{pc},i}[N_{\mathrm{T}}]] \tag{5.13}$$

For the comparison of the PC vectors, $\boldsymbol{p}$ and $\hat{\boldsymbol{p}}$, the sequences a and b are given by:

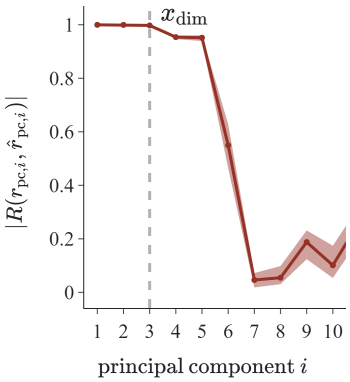
$$a = [p_{i,1}, p_{i,2}, \ldots, p_{i,r_{\mathrm{dim}}}] \tag{5.14}$$
$$b = [\hat{p}_{i,1}, \hat{p}_{i,2}, \ldots, \hat{p}_{i,r_{\mathrm{dim}}}] \tag{5.15}$$

Here, $p_{i,j}$ and $\hat{p}_{i,j}$ denote the jth element of the ith principal component vectors, $\boldsymbol{p}_i$ and $\hat{\boldsymbol{p}}_i$, respectively. As only the strengths and not the signs of the correlations are important, only

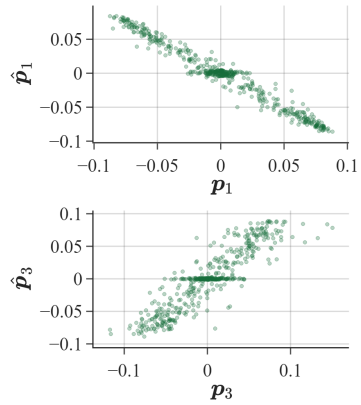
the absolute values of the PCCs are considered. To obtain statistically significant results, the experiment is repeated according to the ensemble experiment parameters outline in Table 4.1. As before, the median value and lower, upper quartile of the resulting absolute PCCs are presented. The results are presented and discussed in the following:



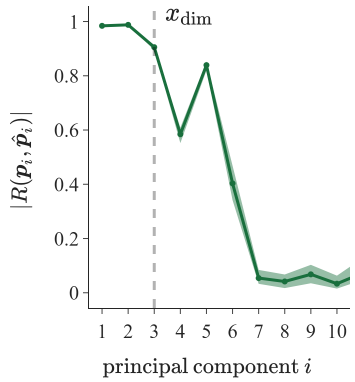
(a) Scatter plots of $r_{\mathrm{pc},i}(t)$ vs. $\hat{r}_{\mathrm{pc},i}(t)$ for $i = 1, 2, 3, 4$



(b) $|R(r_{\mathrm{pc},i}(t),\, \hat{r}_{\mathrm{pc},i}(t))|$



(c) Scatter plots of $\boldsymbol{p}_i$ vs. $\hat{\boldsymbol{p}}_i$ for $i = 1, 2, 3, 4$

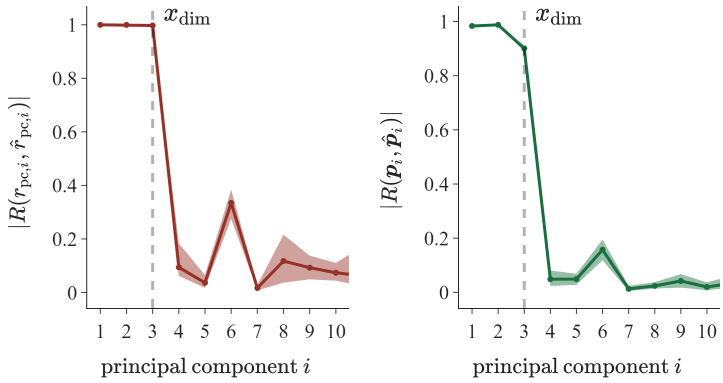


(d) $|R(\boldsymbol{p}_i,\, \hat{\boldsymbol{p}}_i)|$

Figure 5.9: Network influence on principal component quantities, where $(r_{\mathrm{pc},i}(t), \boldsymbol{p}_i)$ correpond to the default RC setup with $\rho_0 = 0.4$, and $(\hat{r}_{\mathrm{pc},i}(t), \hat{\boldsymbol{p}}_i)$ correspond to the same RC setup without network $\rho_0 = 0.0$. (a, c) Scatter plots of $r_{\mathrm{pc},i}(t)$ vs. $\hat{r}_{\mathrm{pc},i}(t)$, and $\boldsymbol{p}_i$ vs. $\hat{\boldsymbol{p}}_i$ for the first four principal components for a single reservoir realization and training-section. (b, d) Absolute value of Pearson correlation coefficient between $r_{\mathrm{pc},i}(t)$ and $\hat{r}_{\mathrm{pc},i}(t)$, and between $\boldsymbol{p}_i$ and $\hat{\boldsymbol{p}}_i$. The curve correponds to the median accross all n_{R} reservoir realizations and n_{T} training-sections according to Tab. 4.1, and the error band corresponds to the lower and upper quartile. Only the first 10 principal components indices i are shown.

Fig. 5.9b and Fig. 5.9d depict the absolute PCC's dependency on the principal component i for the PC-transformed states and PC vectors, respectively. Additionally, Fig. 5.9a presents scatter plots of $r_{\mathrm{pc},i}(t)$ vs. $\hat{r}_{\mathrm{pc},i}(t)$, and Fig. 5.9c shows scatter plots of $\boldsymbol{p}_i$ vs. $\hat{\boldsymbol{p}}_i$ for the first four principal components $i = 1, 2, 3, 4$, obtained from a single reservoir realization and training-section. The scatter plots in Fig. 5.9a illustrate that the first three PC-transformed reservoir states are largely identical, up to a sign, whether the RC architecture includes a reservoir network or not. Even for the fourth component, there is still a high correlation, although it is not as strong as it is for the first three components. From the absolute PCCs shown in Fig. 5.9b, a strong correlation can be observed even up to $i = 5$. For the following principal components, $i > 5$, the PC states are only weakly correlated, but here only the first ten PCs are shown. The error bands are also very narrow up to $i = 5$, indicating that this behavior is consistent regardless of the specific realization of $\mathbf{W}_{\mathrm{in}}$, $\boldsymbol{b}$, $\mathbf{A}$, and training-sections used. A similar behaviour can be observed for the principal component vectors $\boldsymbol{p}_i$ and $\hat{\boldsymbol{p}}_i$, even though the corresponding correlations are less pronounced in this case.

Specifically, for $i = 4$, differences between the two PCC curves can be observed, with the PCC between the PC vectors being lower compared to the corresponding PCC of the PC-transformed reservoir states. Furthermore, it is interesting to note that the scatter plots of $\boldsymbol{p}_i$ vs. $\hat{\boldsymbol{p}}_i$ all show some points lying on a horizontal line at the origin. This line visualizes that some elements in the PC vectors $\hat{\boldsymbol{p}}_i$ of the reservoir without network are close to zero, whereas the corresponding elements in $\boldsymbol{p}_i$ are not.



(a) $|R(r_{\text{pc},i}(t),\, \hat{r}_{\text{pc},i}(t))|$ (b) $|R(\boldsymbol{p}_i,\, \hat{\boldsymbol{p}}_i)|$

Figure 5.10: Same as Fig. 5.9b and Fig. 5.9d, but here the node bias is ommited, i.e. $\sigma_{\text{b}} = 0$.

To study the effect of the node bias, $\boldsymbol{b}$, the same experiment as described above, which uses the default node bias scale of $\sigma_{\text{b}} = 0.4$, is repeated, this time with the node bias removed from both RC setups. The resulting PCC curves for the PC-transformed reservoir states and PC vectors are shown in Fig. 5.10. In this case both RC setups, with and without network, correlate strongly only for the first three components, corresponding to the dimensionality of the Lorenz system. This suggests, that a non-zero node bias, $\boldsymbol{b} \neq 0$, has a substantial influence on the number of principal components, which are solely related to the reservoir's input, $\boldsymbol{x}(t)$.

To conclude, the main results of this study show that at least the first x_{dim} principal components are largely independent of the reservoir's memory induced by its reservoir network, $\mathbf{A}$, and therefore are only related to the reservoir's immediate input, $\boldsymbol{x}(t)$. Furthermore, as the first principal components correspond to the largest explained variances, the results also demonstrate that the reservoir's immediate input, $\boldsymbol{x}(t)$, is responsible for the directions of the largest variance within the reservoir states, $\boldsymbol{r}(t)$. These results are in general agreement with those previously obtained in section 5.2, which showed that the input strength, σ, influenced the explained variances of the first few principal components the most, while variations in the spectral radius, ρ_0, and average node degree, d, left the first few components mostly untouched.

Memory in higher PCs The previous experiment revealed that at least the first x_{dim} principal components can be attributed primarily to the reservoir's current input, $\boldsymbol{x}(t)$. The simple experiment presented here, aims to examine the impact of the previous inputs, $\boldsymbol{x}(t-\Delta t), \boldsymbol{x}(t-2\Delta t), \ldots$, on the PC-transformed reservoir states, $r_{\text{pc},i}(t)$. In order to do so, the experiment examines the responses in the PC-transformed reservoir states, $r_{\text{pc},i}(t)$, when the reservoir's input, $\boldsymbol{x}(t)$, is abruptly removed. The sudden removal of the inputs causes a modification in the reservoir update equation according to:

$$\boldsymbol{r}(t) = \tanh(\mathbf{A}\boldsymbol{r}(t-\Delta t) + \boldsymbol{b}) \tag{5.16}$$

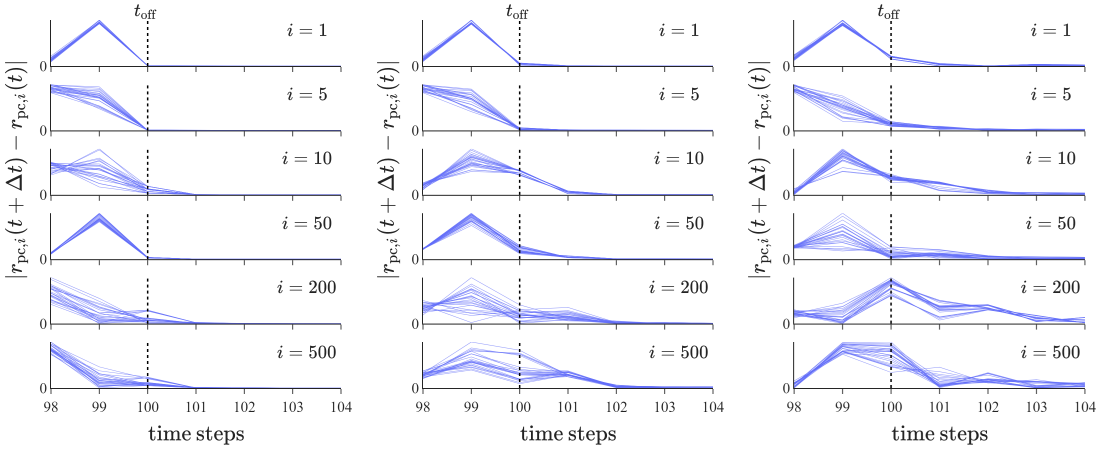
In detail, the experiment is conducted using the following procedure: Initially, the reservoir is driven with the Lorenz system, as it is done in the usual training process, in order to obtain the principal component matrix, $\mathbf{P}$, and mean, $\overline{\boldsymbol{r}}$, which define the PC-transformation, $\boldsymbol{r}_{\text{pc}} = \mathbf{P}(\boldsymbol{r} - \overline{\boldsymbol{r}})$. Next, the reservoir is again driven with a separate section of the Lorenz time series for $t_{\text{off}} - 1$ steps, after which, starting from the t_{off}'s step, the input is removed from the reservoir's update equation according to Eq. (5.16). The resulting reservoir states are then transformed onto the principal components using the previously determined PC matrix, $\mathbf{P}$, and state mean, $\overline{\boldsymbol{r}}$. This allows for an examination of the responses in the individual PC-transformed reservoir states, $r_{\text{pc},i}(t)$. To

visualize the responses, the absolute difference between two consecutive time steps of the PC-transformed reservoir states, $r_{\mathrm{pc},i}(t)$, is calculated:

$$\mathrm{RESP}_i(t) = |r_{\mathrm{pc},i}(t + \Delta t) - r_{\mathrm{pc},i}(t)| \tag{5.17}$$

If a principal component index, i, is solely determined by the reservoir's immediate input, $\boldsymbol{x}(t)$, the corresponding response in $r_{\mathrm{pc},i}(t)$ will be instantaneous, and thus $\mathrm{RESP}_i(t) = 0$ for all $t \geq t_{\mathrm{off}}$. In contrast, principal components, which incorporate a greater portion of the input's history are expected to show a longer echo.

Besides the default RC setup with a spectral radius of $\rho_0 = 0.4$, two additional setups with a lower, $\rho_0 = 0.1$, and higher spectral radius, $\rho_0 = 0.9$, are also tested. An ensemble of $n_{\mathrm{R}} = 5$ different reservoir realizations trained on $n_{\mathrm{T}} = 5$ different training-sections are investigated. As mentioned above, each training-section is only used to obtain the PC-transformation parameters, $\mathbf{P}$ and $\overline{\boldsymbol{r}}$, meaning an output fit using ridge regression is not required. For all trained reservoirs, the final response collection is based on the same section of the time series.



(a) $\rho_0 = 0.1$ (b) $\rho_0 = 0.4$ (c) $\rho_0 = 0.9$

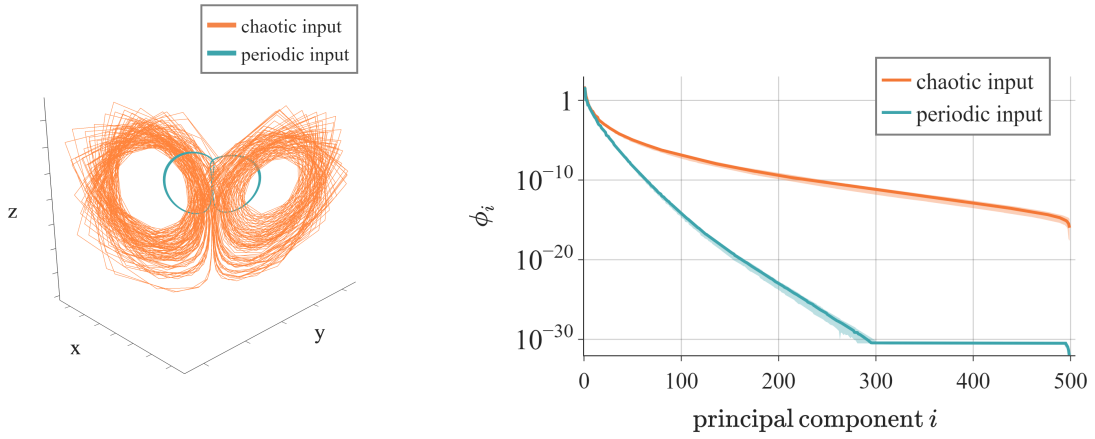
Figure 5.11: Responses in PC-transformed reservoir states $r_{\mathrm{pc},i}(t)$, when driven with an abruptly stopping signal.

The responses for the selection of six principal components, $i = 1, 5, 10, 50, 200, 500$, and for the three tested spectral radia, $\rho_0 = 0.1$, $\rho_0 = 0.4$, and $\rho_0 = 0.9$, are shown in Fig. 5.11. It is important to note, that the scales of the individual y-axes are not equivalent, instead, they are selected to match their corresponding data. The results indicate that the duration of the echo increases as either the principal component index i (plots from top to bottom) or the spectral radius ρ_0 (plots from left to right) increases. The responses in the first principal component, $i = 1$, are approximately instantaneous, especially for the two smaller spectral radii, $\rho_0 = 0.1$ and $\rho_0 = 0.4$. This demonstrates once again that the first principal component is primarily influenced by the reservoir's immediate input. The longer echo within the higher principal components indicates that inputs from further in the past have a greater impact on these components. This is consistent with the findings reported in [9]. The longer echo for larger spectral radii demonstrates the greater impact of past inputs on the current reservoir state through the larger elements in the reservoir network, $\mathbf{A}$.

5.4 Possible further studies

The experiments presented in this chapter were only applied to the synthetically generated three-dimensional Lorenz system. Investigating the influence of inputs with different characteristics on the reservoir state's explained variances, ϕ_i, could provide valuable insights. For example, the effects of noisy inputs, as seen in real data, or higher-dimensional inputs such as the Kuramoto-Sivashinsky or Lorenz-96 systems could be studied. Additionally, the impact of simpler periodic

inputs, as opposed to chaotic ones, can be investigated. As a simple demonstration of the difference between chaotic and periodic input data, Fig. 5.12 illustrates the explained variances resulting from a chaotic solution of the Lorenz system using Lorenz-$\beta = 8/3$ and a periodic solution using Lorenz-$\beta = 0.2$. Such experiments suggest that periodic inputs lead to faster declines in explained variances compared to chaotic inputs. Therefore, exploring the potential of using the explained variances, ϕ_i, as a tool to probe the nature of the inputs, $\boldsymbol{x}(t)$, could lead to interesting insights.



(a) Input data trajectory

(b) Explained variances

Figure 5.12: Explained variances resulting from chaotic (Lorenz-$\beta = 8/3$) and periodic (Lorenz-$\beta = 0.2$) Lorenz system input.

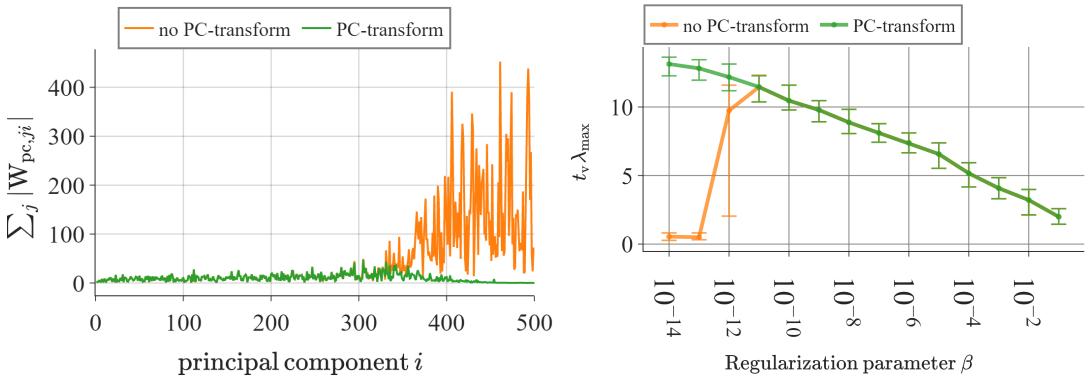
Additionally, the effect of different modifications to the reservoir update equation on the explained variances, ϕ_i, beyond those considered in this chapter, may also yield interesting findings. For instance, the effect of different activation functions, such as the *Sigmoid* or *ReLU* function, or the impact of different reservoir network topologies, such as small-world, scale-free or densely-connected networks, can be studied. Furthermore, the explained variances of the generalized reservoir states resulting from a nonlinear readout, $\psi(\cdot)$, such as the extended or alternating *Lu readout* can be investigated.

Considering the prediction phase of the reservoir, the possibilities of using a comparison between the prediction variances, $\tilde{\phi}_i$, and the explained variances, ϕ_i, as a tool to assess the prediction quality of the reservoir can also be further explored. Intuitively, one might suggest that strong similarities between $\tilde{\phi}_i$ and ϕ_i indicate good prediction qualities, but proper statistical experiments also considering long-term ergodic prediction measures such as Lyapunov exponents and correlation dimensions are needed.

This chapter showed that the regularization in ridge regression occurs through the PC-filter, $f(\phi_i)$, which resembles a soft step-function going from 1 to 0 and suppresses all principal components located on the right side of the step. The principal component index, i_{co}, that marks the step location in the PC-filter was referred to as the "number of active components", but in reality, components $i > i_{\text{co}}$ still contribute weakly to the output due to the soft nature of the PC-filter. This suggests the use of a sharp step function, which is strictly 1 for all principal components $i \leq i_{\text{co}}$ and strictly 0 for all $i > i_{\text{co}}$. The resulting regression scheme is known as as *principal component regression* (PCR) [15]. To implement PCR for reservoir computing, similarly as done in [32], one can transform the reservoir states, $\boldsymbol{r}(t)$, into their PC-coordinate frame, yielding $\boldsymbol{r}_{\text{pc}}(t)$, and then keeping only the first i_{co} principal components, resulting in a dimensionality reduction. The resulting states are then used as the input variables, $\boldsymbol{k}$, in ordinary least squares regression: $\boldsymbol{k} = [r_{\text{pc},1}, r_{\text{pc},2}, \ldots, r_{\text{pc},i_{\text{co}}}]^T$. In [32], the authors used ridge regression for the final output fit instead of OLS regression. However, they did not mention that both ridge regression and principal component regression result in a suppression of larger principal components. An exploration of the potential advantages of using PCR instead of RR can be conducted.

Furthermore, it was observed in some cases that applying a full PC-transformation to the reservoir states prior to performing type-b ridge regression resulted in greater numerical stability for the RR

output fit, when using an extremely low regularization parameter, β. For instance, when comparing the results of the default RC setup and the same setup with the additional PC-transformation, applied to the Lorenz system with a time step of $\Delta t = 0.05$ and a low regularization parameter, $\beta \leq 10^{-12}$, differences between the two can be observed. This is illustrated in Fig. 5.13, where it is shown that without using a PC-transformation before RR, the PC-transformed output matrix, $\mathbf{W}_{\text{pc}}$, displays significant contributions from higher principal components, despite the fact that they should be suppressed. This shows that while both methods are mathematically equivalent, the prior PC-transformation seems to enhance numerical stability during ridge regression. This improved numerical stability is reflected in the significantly longer valid times, while the standard RC setup becomes unstable. For the larger Lorenz time step of $\Delta t = 0.1$, which is used throughout this thesis, no differences in the valid times could be observed. A further exploration into this enhanced numerical stability when using an additional PC-transformation could yield noteworthy result.



(a) Regularization parameter, $\beta = 10^{-12}$

(b) Valid times

Figure 5.13: Influence of PC-transformation layer on (a) PC-transformed output matrix, $\mathbf{W}_{\text{pc}}$, and (b) valid times, t_{v}, for Lorenz system with $\Delta t = 0.05$.

The results demonstrated in this chapter revealed that the first few principal components of the reservoir states, $\boldsymbol{r}(t)$, are mostly influenced by their immediate input, $\boldsymbol{x}(t)$. These results were obtained by comparing a standard RC setup with a modified setup where the influence of past inputs was removed by setting the reservoir network to zero, $\mathbf{A} = 0$. Denoting the reservoir states from the setup without network as $\boldsymbol{r}^{(0)}(t)$, influenced by $n = 0$ past input states, and the reservoir states from the standard setup as $\boldsymbol{r}^{(\infty)}(t)$, influenced by all past inputs, $n = \infty$, one can also define "n-step reservoir states", $\boldsymbol{r}^{(n)}(t)$, by limiting the influence to the nth step into the past:

$$
\begin{aligned}
\boldsymbol{r}^{(0)}(t) &= \tanh\left\{\mathbf{W}_{\text{in}}\boldsymbol{x}(t)\right\} \\
\boldsymbol{r}^{(1)}(t) &= \tanh\left\{\mathbf{W}_{\text{in}}\boldsymbol{x}(t) + \mathbf{A}\tanh\left[\mathbf{W}_{\text{in}}\boldsymbol{x}(t-\Delta t)\right]\right\} \\
\boldsymbol{r}^{(2)}(t) &= \tanh\left\{\mathbf{W}_{\text{in}}\boldsymbol{x}(t) + \mathbf{A}\tanh\left[\mathbf{W}_{\text{in}}\boldsymbol{x}(t-\Delta t) + \mathbf{A}\tanh(\mathbf{W}_{\text{in}}\boldsymbol{x}(t-2\Delta t))\right]\right\} \\
&\vdots \\
\boldsymbol{r}^{(\infty)}(t) &= \tanh\left\{\mathbf{W}_{\text{in}}\boldsymbol{x}(t) + \mathbf{A}\boldsymbol{r}^{(\infty)}(t-\Delta t)\right\}
\end{aligned}
$$

For simplicity, the node bias, $\boldsymbol{b}$, and the input standardizer, $S(\cdot)$, have been omitted. A comparison of the principal components of the n-step reservoir states, $\boldsymbol{r}^{(n)}(t)$, with those of the standard reservoir states, $\boldsymbol{r}(t) = \boldsymbol{r}^{(\infty)}(t)$, could provide more insights into the memory capacities of the individual principal components.

Chapter 6

Hybrid Reservoir Computing

6.1 Introduction

The reservoir computing approach described so far was purely data-driven which is a useful approach if there is no knowledge about the governing equations producing the data. In many cases though some knowledge about the underlying system is known and its inclusion into the data-driven methods can usually be helpful in improving the prediction quality. These combinations of machine learning and knowledge-based models are often called *hybrid* or *physics-informed* machine learning models. These models can be seen as a part of an important emerging field called *Theory-guided data science* which "seeks to exploit the promise of data science without ignoring the treasure of knowledge accumulated in scientific principles" [1].

One example area of science which benefits well from hybrid models is the field of extreme events in complex dynamical systems, which studies phenomena as epidemics, tsunamis, droughts and more [33]. These systems are usually extremely complex with a multitude of different interacting parts [34]. As such, a detailed description of their dynamics by the means of governing equations is often not feasible even if the underlying equations are known in principle. In these cases approximate knowledge-based models, which often reduce the complexity of the system in order to make it computationally feasible, are used to predict the behaviour of the system. Due to these approximations, interesting nonlinear phenomena of these systems as extreme events are often difficult to predict. Purely data-driven models such as reservoir computing can often predict the systems main attractor quite well, but also face challenges dealing with extreme events which only occur rarely in the data [33]. The combination of knowledge-based equations with data-driven machine learning methods often leads to an improvement of the prediction quality compared to a purely data-driven prediction.

In the field of reservoir computing different ways to implement physical knowledge into the RC framework were explored. Doan et al. [35, 36] introduced the PI-ESN (*physics-informed echo state network*) which was then extended by [37]. The PI-ESN uses the time series' true physical equations in order to fine-tune the RC output matrix, $\mathbf{W}_{\text{out}}$, in a second training phase, after being normally trained with the data only. Consequently, the PI-ESN architecture corresponds to a normal RC architecture, as only the values of the output matrix, $\mathbf{W}_{\text{out}}$, are adjusted using the physical equations of the system. Their example predictions on the Lorenz system and on chaotic flows related to the Navier-Stokes equation showed improved robustness to noise and overall better prediction qualities. Another hybrid approach was introduced by Pathak et al. [10], and then utilized and extended in [38, 26, 39, 40], where, in contrast to the PI-ESN, the physical knowledge becomes an integral part of the hybrid RC architecture. The physical knowledge of the system, as introduced in [10], is given as an imperfect knowledge-based model, $K(\cdot)$, which is a function of the current state of the time series, $\boldsymbol{u}(t)$, that outputs an approximation of the next state:

$$K(\boldsymbol{u}(t)) \approx \boldsymbol{u}(t + \Delta t) \tag{6.1}$$

By implementing the knowledge-based model, $K(\cdot)$, into the RC architecture, Pathak et al. were

able to show large improvements in prediction qualities of the Lorenz and Kuramoto-Sivashinsky system over a normal data-driven RC approach [10].

In this chapter, the hybrid RC approach by Pathak et al. is studied by comparing different variations in the architecture, and by investigating the effect of different kinds of knowledge-based models, $K(\cdot)$.

6.2 Hybrid RC architectures

The hybrid approach introduced by Pathak et al. [10] consists of two independent entry points of the knowledge-based model (KBM), $K(\cdot)$, into the RC framework, which will be denoted here as *input hybrid* (IH) and *output hybrid* (OH). If both *input hybrid* and *output hybrid* are used, as originally in [10], the resulting RC architecture will be denoted as *full hybrid* (FH).

The dimension of the knowledge-based output, $K(\boldsymbol{u}(t))$, is denoted as K_{dim}. In the original implementation, the knowledge-based model, $K(\cdot)$, is taken to be an approximate next-step predictor, i.e. Eq. (6.1) holds. In this case, the dimension of the KBM output equals the dimension of the time series: $K_{\text{dim}} = u_{\text{dim}}$. In the course of this chapter the equation Eq. (6.1) will be relaxed so that $K(\cdot)$ is just some function containing physical knowledge of the system, which does not necessarily have to be an approximate next-step predictor. In these cases $K_{\text{dim}} \neq u_{\text{dim}}$ is also possible, but the hybrid RC architectures described below remain the same. In the following the same notation as in Eq. (4.22) is used, where $\tilde{\boldsymbol{u}}(t)$ stands for the true time series, $\boldsymbol{u}(t)$, during synchronization and training, as well as for the previous reservoir prediction, $\boldsymbol{y}_{\text{r}}(t - \Delta t)$, during the prediction phase.

Input hybrid In the *input hybrid* approach the reservoir input, $\boldsymbol{x}(t)$, is given by the concatenation of the time series, $\tilde{\boldsymbol{u}}(t)$, and the knowledge-based prediction, $K(\tilde{\boldsymbol{u}}(t))$. Mathematically, Eq. (4.21) belonging to normal reservoir computing is replaced by Eq. (6.2):

$$\boldsymbol{x}(t) = \begin{bmatrix} \tilde{\boldsymbol{u}}(t) \\ K(\tilde{\boldsymbol{u}}(t)) \end{bmatrix} \tag{6.2}$$

This extends the former input dimension of normal RC, $x_{\text{dim}} = u_{\text{dim}}$, to $x_{\text{dim}} = u_{\text{dim}} + K_{\text{dim}}$. Consequently, the input matrix, $\mathbf{W}_{\text{in}}$, now becomes a $r_{\text{dim}} \times (u_{\text{dim}} + K_{\text{dim}})$ matrix, where the first u_{dim} columns connect the pure input, $\tilde{\boldsymbol{u}}(t)$, to the reservoir nodes and the last K_{dim} columns connect the knowledge-based predictor, $K(\tilde{\boldsymbol{u}}(t))$, to the reservoir nodes, $\boldsymbol{r}(t)$, respectively.

Output hybrid In the output hybrid approach, the input variables for ridge regression, $\boldsymbol{k}$, which are given by the pure reservoir states, $\boldsymbol{k} = \boldsymbol{r}$, or a nonlinear readout of the latter, $\boldsymbol{k} = \psi(\boldsymbol{r})$, are simply extended by the knowledge based model, $K(\tilde{\boldsymbol{u}})$. Assuming here that no readout function $\psi(\cdot)$ used, the output hybrid architecture is implemented by replacing $\boldsymbol{k}(t) = \boldsymbol{r}(t)$ from Eq. (4.25) with:

$$\boldsymbol{k}(t) = \begin{bmatrix} \boldsymbol{r}(t) \\ K(\tilde{\boldsymbol{u}}(t)) \end{bmatrix} \tag{6.3}$$

As a result, the output matrix, $\mathbf{W}_{\text{out}}$, now has the shape of $u_{\text{dim}} \times (r_{\text{dim}} + K_{\text{dim}})$, where the first r_{dim} rows connect the reservoir states, $\boldsymbol{r}(t)$, to the output and the last K_{dim} connect the knowledge-based model, $K(\tilde{\boldsymbol{u}}(t))$, to the output, respectively. The reservoir output, $\boldsymbol{y}_{\text{r}}$, can then be rewritten as:

$$\boldsymbol{y}_{\text{r}}(t) = \mathbf{W}_{\text{out}} \begin{bmatrix} \boldsymbol{r}(t) \\ K(\tilde{\boldsymbol{u}}(t)) \end{bmatrix} + \boldsymbol{w} = \mathbf{W}_{\text{res}} \boldsymbol{r}(t) + \mathbf{W}_{\text{kbm}} K(\tilde{\boldsymbol{u}}(t)) + \boldsymbol{w} \tag{6.4}$$

The partial output matrices, $\mathbf{W}_{\text{res}}$ and $\mathbf{W}_{\text{kbm}}$, are of the shape $u_{\text{dim}} \times r_{\text{dim}}$ and $u_{\text{dim}} \times K_{\text{dim}}$, respectively. The mean output contributions, $\Omega_{\text{res},j}$ and $\Omega_{\text{kbm},j}$, of the two matrices, $\mathbf{W}_{\text{res}}$ and

$\mathbf{W}_{\text{kbm}}$, can be defined by averaging the absolut elements of $\mathbf{W}_{\text{res}}$ and $\mathbf{W}_{\text{kbm}}$ over the reservoir nodes and KBM-output dimensions, respectively:

$$\Omega_{\text{res},j} = \frac{1}{r_{\text{dim}}} \sum_{i=1}^{r_{\text{dim}}} |W_{\text{res},ji}| \tag{6.5}$$

$$\Omega_{\text{kbm},j} = \frac{1}{K_{\text{dim}}} \sum_{i=1}^{K_{\text{dim}}} |W_{\text{kbm},ji}| \tag{6.6}$$

Full hybrid In the *full hybrid* approach, as it was initially introduced in [10], both equations Eq. (6.2) and Eq. (6.3) are used.

The three possible hybrid RC approaches, IH, OH, and FH, are visualized in Fig. 6.1. The main idea of these hybrid approaches is to make the physical knowledge, $K(\cdot)$, available within the reservoir prediction. The *output hybrid* framework works in the straight forward manner, where the ridge regression finds the optimal weights in $\mathbf{W}_{\text{out}}$ which combine the reservoir states, $\boldsymbol{r}(t)$, with the KMB output, $K(\tilde{\boldsymbol{u}}(t))$, in order to learn the desired output. In the *input hybrid* approach the KMB output, $K(\tilde{\boldsymbol{u}}(t))$, is just fed into the reservoir as additional input, making it less straight forward to interpret.

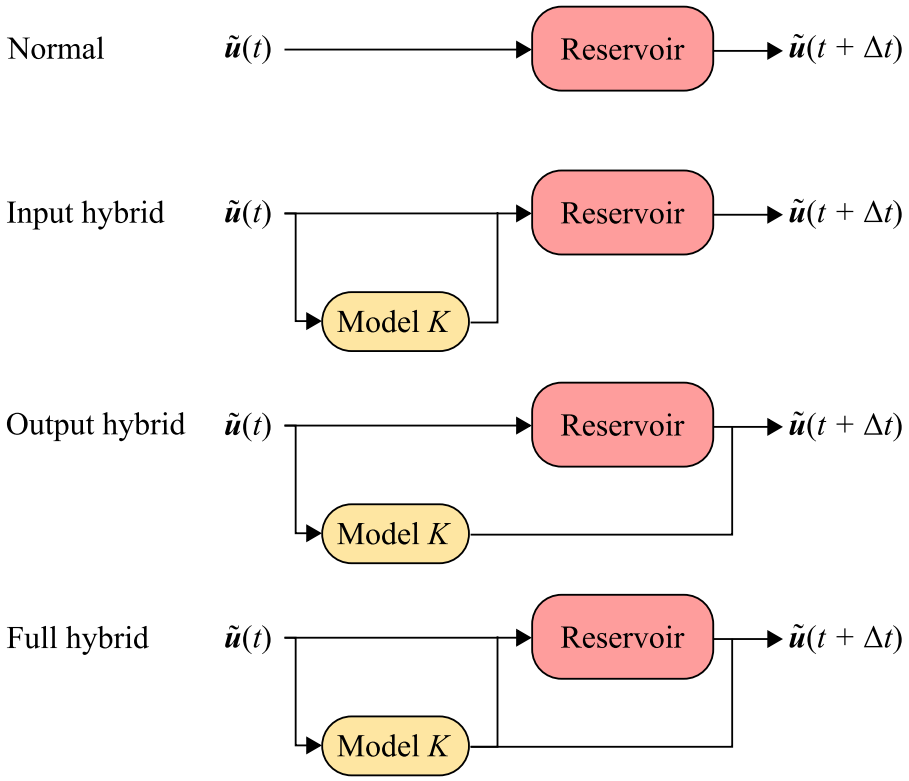


Figure 6.1: Hybrid RC architectures.

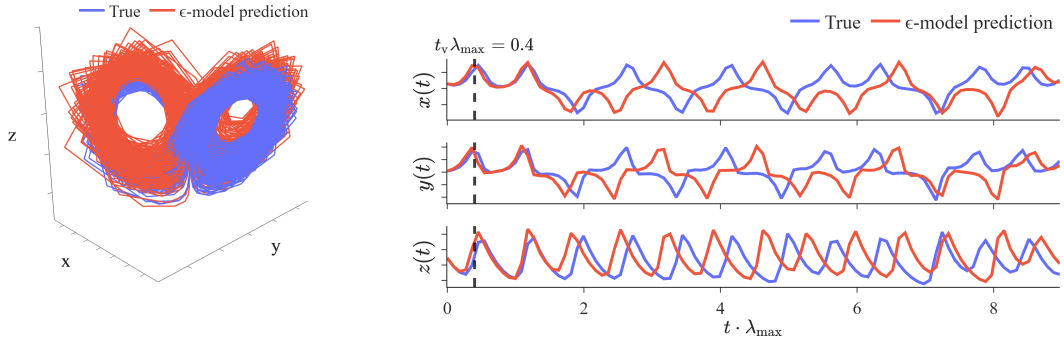
6.3 Knowledge-based models

In this section the different knowledge-based models, $K(\cdot)$, which will be used to test the hybrid architectures IH, OH, and FH are introduced. As before, it is assumed that the time series, $\boldsymbol{u}(t)$, to be predicted is resulting from a numerical solution, using RK4 Eq. (2.3), of a continuous time

dynamical system given by a flow equation, Eq. (2.2): $\dot{\boldsymbol{u}}(t) = F(\boldsymbol{u}(t))$. Denoting the RK4 iterator as $M(\cdot)$, the true time series is created by iteratively applying:

$$M(\boldsymbol{u}(t)) = \boldsymbol{u}(t + \Delta t) \tag{6.7}$$

ϵ-model K_ϵ The ϵ-model, $K_\epsilon(\cdot)$, as was used in [10], is created by modifying one parameter value of the true dynamical system's flow, $F(\cdot)$, and then performing the same numerical integration method, i.e. RK4, in order to artificially create an imperfect predictor, $K_\epsilon(\cdot)$. The modified parameter in the flow is obtained by multiplying the true systems' parameter by a factor $(1 + \epsilon)$, where ϵ is denoted here as the *model error*. If the model error, ϵ, is not too large, $K_\epsilon(\boldsymbol{u}(t)) \approx \boldsymbol{u}(t + \Delta t)$ holds. For example, in the Lorenz system, Eq. (2.10), the parameter $\rho = 28$ can be modified to $\rho(1 + \epsilon) = 28(1 + \epsilon)$. In Fig. 6.2 the true Lorenz time series, created using the default parameters $\rho = 28, \sigma = 10, \beta = 8/3$, is compared with the ϵ-model prediction where σ and β remain the same, but ρ is modified to $\tilde{\rho} = 28(1 + \epsilon) = 30.8$, using $\epsilon = 0.1$. The pure ϵ-model prediction only archives a valid time of $t_\mathrm{v}\lambda_\mathrm{max} = 0.4$. From the three-dimensional trajectory plot in Fig. 6.2a one can observe that the effect of the modification of ρ results in a slightly rotated Lorenz attractor.



(a) 3D-Trajectory (b) Time series

Figure 6.2: True Lorenz time series vs. ϵ-model with the parameter change: $\rho \to (1 + \epsilon)\rho$. Here, the model error $\epsilon = 0.1$ is used, which modifies the default $\rho = 28$ to $\tilde{\rho} = 30.8$.

Flow-model $K_\mathbf{flow}$ The flow-model, K_flow, is simply given as the flow of the true dynamical system, $K_\mathrm{flow}(\cdot) = F(\cdot)$. This model is used to investigate if a knowledge-based model that does not approximately predict the next state of the system, $K_\mathrm{flow}(\boldsymbol{u}(t)) \not\approx \boldsymbol{u}(t + \Delta t)$, but still contains information about the system is able to improve the prediction quality of the hybrid reservoir.

Dimension-selection-model $K_\mathbf{ds}$ In this case, the knowledge-based model, $K_\mathrm{ds}(\cdot)$, is given as only one dimension of the true systems iterator, $M(\cdot)$. Writing the true iterator dimension-wise, $M(\boldsymbol{u}) = [M_1(\boldsymbol{u}), M_2(\boldsymbol{u}), \ldots, M_{u_\mathrm{dim}}(\boldsymbol{u})]^T$, the dimension-selection-model, $K_\mathrm{ds}(\cdot)$, is given by:

$$K_\mathrm{ds}(\boldsymbol{u}) = M_l(\boldsymbol{u}) \tag{6.8}$$

Here, the dimension l represents the chosen dimension. As before with the flow-model, the dimension-selection-model does not provide a next-step predictor according to Eq. (6.1), but still contains useful information about the dynamical system.

Sinus-model $K_{\sin}$ In order to test the influence of an extremely inaccurate model which incorporates no knowledge of the underlying system, but still produces a nonlinear transformation of the state, the sinus-model $K_{\sin}(\cdot)$ is also tested, which is given as:

$$K_{\sin}(\boldsymbol{u}) = [\sin(u_1), \sin(u_2), \ldots, \sin(u_{u_{\text{dim}}})]^T \tag{6.9}$$

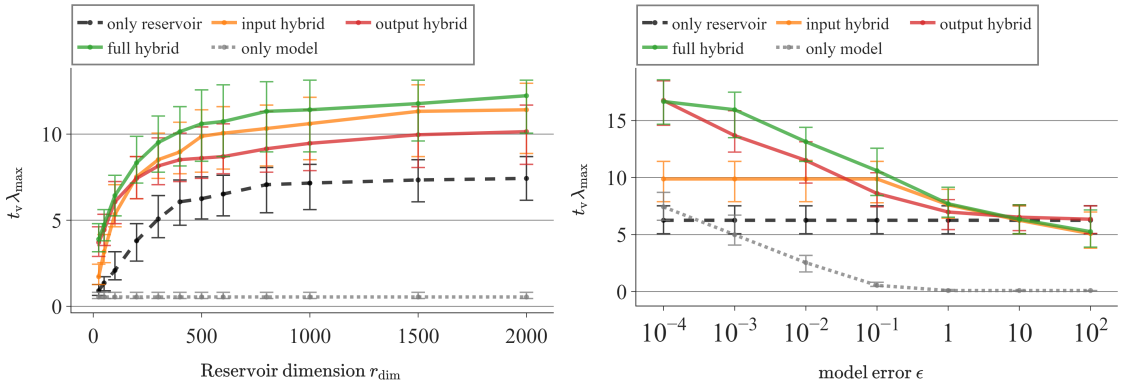
6.4 Experiments and results

In order to investigate the predictive qualities of the three hybrid models, IH, OH, and FH, the valid times for these architectures are compared to each other and to a normal non-hybrid reservoir, for the four kinds of knowledge-based models, ϵ-model, flow-model, dimension-selection-model, and sinus-model, introduced in the previous section. In the case of the ϵ-model, the model-only prediction, using no reservoir, is also included in the comparison. As before, the default RC setup defined in Sec. 4.5 is used, i.e. node bias and input standardization is incorporated, and the nonlinear reservoir readout $\psi(\cdot)$ is omitted. All hyperparameter values which are not explicitly varied are taken from Tab. 4.2, and the ensemble experiment parameters are taken from Tab. 4.1. In addition to the well-known Lorenz system, eight other three-dimensional continuous-time chaotic dynamical systems are studied, whose flow equations and default parameters are listed in appendix A. Table 6.1 lists all nine tested systems, along with the ϵ-parameter which defines the choice of parameter that is modified in the ϵ-model.

System	Eq.	ϵ-param	System	Eq.	ϵ-param
Chen	(A.1)	a	Roessler	(A.5)	a
ChuaCircuit	(A.2)	α	Rucklidge	(A.6)	a
DoubleScroll	(A.3)	a	Thomas	(A.7)	b
Halvorsen	(A.4)	a	Windmi	(A.8)	a
Lorenz	(2.10)	ρ			

Table 6.1: Equation references and selected ϵ-parameter which is modified in the ϵ-model.

ϵ-model Fig. 6.3 shows the influence of reservoir dimension, r_{dim}, and model error, ϵ, on the Lorenz system's valid times for the three hybrid architectures IH, OH, FH. The model-only and reservoir-only predictions are also shown for comparison.



(a) Model error fixed at $\epsilon = 0.1$

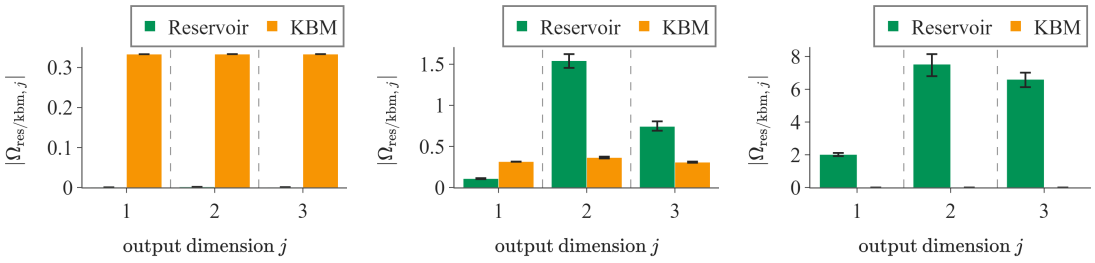
(b) Reservoir dimension fixed at $r_{\text{dim}} = 500$

Figure 6.3: Hybrid RC using the ϵ-model for the Lorenz system.

With the model error fixed at $\epsilon = 0.1$, Fig. 6.3a clearly shows that all three hybrid approaches significantly increase the valid times compared to either a model-only or a reservoir-only prediction. The FH prediction, which uses both the input and output hybrid approaches, provides the best prediction over all reservoir dimensions, r_{dim}, tested. It is interesting to note that for small reservoirs, $r_{\text{dim}} \approx 25$, the predictive qualities corresponding to FH and OH are equivalent and

much higher than those of the IH and the reservoir-only approach. This is due to the fact that the output hybrid approach is independent of the reservoir dimension, r_{dim}, so that the full power of an accurate knowledge-based model can be utilized even for small reservoirs. In contrast, the IH approach doubles the reservoir input size, and therefore requires a sufficiently large reservoir to effectively utilize the KBM effectively. This can be seen from the IH valid times curve surpassing the OH valid times curve at $r_{\text{dim}} \approx 200$. Fig. 6.3b shows the effect of a varying model error ϵ for a fixed reservoir dimension of $r_{\text{dim}} = 500$. As expected, the valid times of all hybrid approaches decrease as the model error increases from an extremely accurate model, $\epsilon = 10^{-4}$, to an extremely inaccurate model, $\epsilon = 100$. For the highly accurate model with $\epsilon = 10^{-4}$, the valid times of FH and OH are equivalent and significantly higher than those of the IH, reservoir-only and model-only approaches. It is interesting to note, that the predictive qualities of IH do not vary within the range from $\epsilon = 10^{-4}$ to $\epsilon = 0.1$. As the model error increases, the OH approach converges to the reservoir-only prediction. The FH and IH approaches, on the other hand, show even lower valid times in the case of the highly inaccurate model with $\epsilon = 100$.

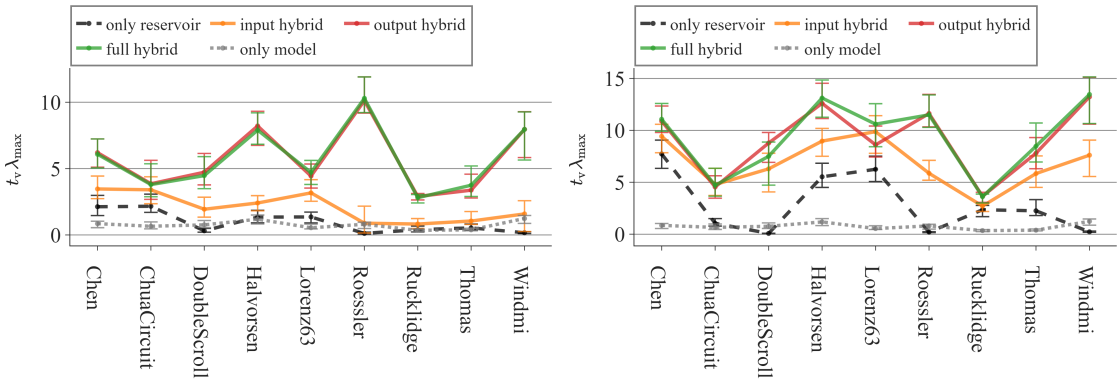
For the OH approach using $r_{\text{dim}} = 500$, the reservoir and KBM output contributions, $\Omega_{\text{res},j}$ and $\Omega_{\text{kbm},j}$, are shown in Fig. 6.4 for three different model errors $\epsilon \in (10^{-4}, 0.1, 100)$.



(a) $\epsilon = 10^{-4}$ (b) $\epsilon = 0.1$ (c) $\epsilon = 100$

Figure 6.4: Output contributions of OH approach with $r_{\text{dim}} = 500$ using the ϵ-model.

In the case of the highly accurate model with $\epsilon = 10^{-4}$, most of the weights in $\mathbf{W}_{\text{out}}$ are contributed to the knowledge-based ϵ-model. For $\epsilon = 0.1$, which corresponds to the model error used in Fig. 6.3a, the contributions of the reservoir nodes and of the KBM are approximately balanced. In the case of the highly inaccurate model with $\epsilon = 100$, most of the output can be attributed to the reservoir nodes. This demonstrates how the output matrix, $\mathbf{W}_{\text{out}}$, in the OH and FH approaches is able to choose between the KBM and the reservoir nodes, depending on the accuracy of the model. This also demonstrates the power of the OH approach, which in the worst case, when the KBM is very inaccurate, just reduces to the reservoir-only prediction. For IH and OH, however, the KBM-output directly influences the reservoir states, $\boldsymbol{r}(t)$, which, in the case of a highly inaccurate model can reduce the predictive ability of the reservoir states, as shown in Fig. 6.3b for $\epsilon = 100$.



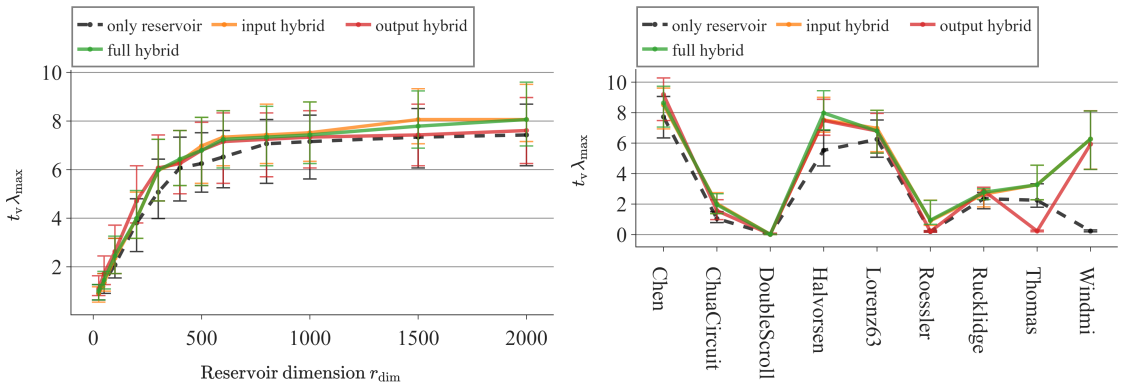
(a) Smaller reservoir $r_{\text{dim}} = 50$ (b) Larger reservoir $r_{\text{dim}} = 500$

Figure 6.5: Hybrid RC using the ϵ-model with $\epsilon = 0.1$ for all nine dynamical systems.

Fig. 6.5 compares the three hybrid architectures with a fixed model error $\epsilon = 0.1$ for the nine different dynamical systems from Tab. 6.1 using a smaller reservoir $r_{\text{dim}} = 50$ and a larger reservoir

$r_{\text{dim}} = 500$. In all cases, the hybrid approaches IH, OH, and FH show higher valid times than the reservoir-only and the model-only prediction, with the OH and FH approaches yielding even better prediction qualities compared to IH. Furthermore, the equivalence of the FH and OH approaches using smaller reservoirs such as $r_{\text{dim}} = 50$, which was already observed for the Lorenz system in Fig. 6.3a, can also be confirmed for the other dynamical systems. Even in the case of the larger reservoir with $r_{\text{dim}} = 500$, the OH valid times are equivalent to the FH valid times for many systems.

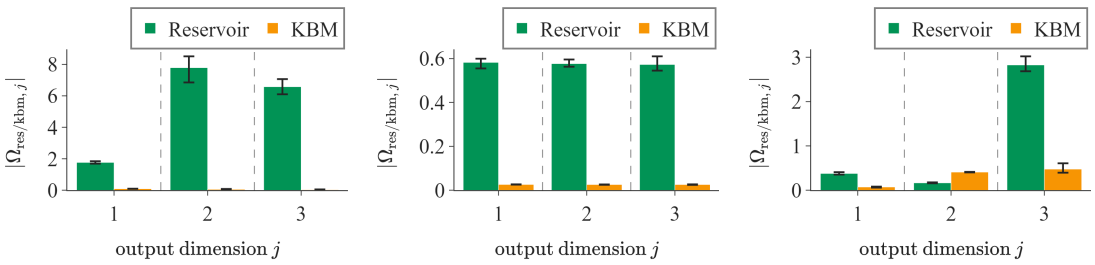
Flow-model As before with the ϵ-model, Fig. 6.6a shows the influence of the reservoir dimension, r_{dim}, on the Lorenz prediction for the three hybrid approaches, this time using the flow-model as the knowledge-based model. All hybrid approaches improve the prediction qualitiy compared to the reservoir-only prediction, but compared to the ϵ-model with $\epsilon = 0.1$, the improvements are only marginal. As shown in Fig. 6.6b, the valid times of the hybrid predictions with $r_{\text{dim}} = 500$ of most of the other dynamical systems also show only small improvements over the reservoir-only prediction. Nevertheless, the Halvorsen system and the Windmi system show significant improvements in all hybrid architectures using the flow-model. The prediction of the Thomas system shows a case where the OH approach actually performs worse than the reservoir-only prediction.



(a) Lorenz system (b) All systems with $r_{\text{dim}} = 500$

Figure 6.6: Hybrid RC using the flow-model.

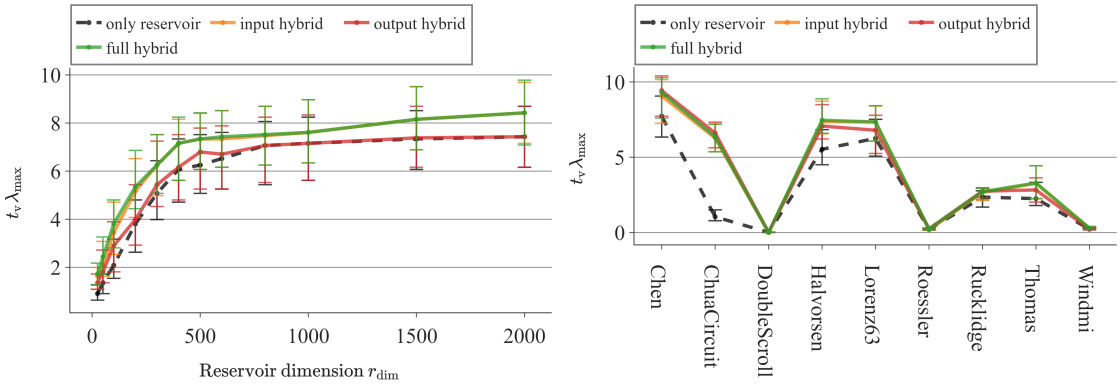
The higher effectiveness of the hybrid predictions using the flow-model in the cases of the Halvorsen and Windmi systems can also be observed in the reservoir and KBM output contributions, $\Omega_{\text{res},j}$ and $\Omega_{\text{kbm},j}$, for the OH approach. In both cases, shown in Fig. 6.7b and Fig. 6.7c, a non-vanishing KBM-contribution is observed, although the KBM-contribution for the Halvorsen system is still rather small compared to the reservoir-contribution. In contrast, for the Lorenz system shown in Fig. 6.7a, the KBM-contribution to the flow-model is only vanishingly small, which corresponds well to the marginal prediction improvement in this case.



(a) Lorenz system (b) Halvorsen system (c) Windmi system

Figure 6.7: Output contributions of OH approach with $r_{\text{dim}} = 500$ using the flow-model.

Dimension-selection-model Fig. 6.8a shows the influence of the reservoir dimension, r_{dim}, on the prediction of the Lorenz system using the dimension-selection-model with $l = 1$ serving as the knowledge-based model. In this scenario, the FH and IH approaches perform similarly well and enhance the prediction compared the OH and reservoir-only approaches for all reservoir dimensions tested. The OH approach only shows a slight improvement over the reservoir-only approach for small reservoir sizes. Fig. 6.8b depicts the valid times for all systems evaluated using a reservoir of $r_{\text{dim}} = 500$ and again setting the selected dimension to $l = 1$ for each system. For certain systems, such as the Chen, ChuaCircuit and Halvorsen systems, the dimension-selection-model significantly improves the prediction. However, for other systems, such as the DoubleScroll, Roessler and Windmi systems, the hybrid approaches are still not able to generate a successful prediction.

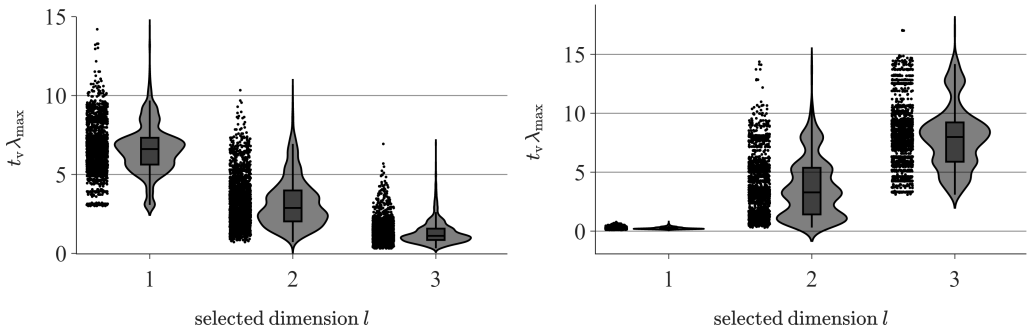


(a) Lorenz system

(b) All systems with $r_{\text{dim}} = 500$

Figure 6.8: Hybrid RC using the dimension-selection-model with $l = 1$ for each system.

The dimension-selection-model's effectiveness is strongly influenced by the chosen selected dimension l, as demonstrated in Fig. 6.9, which shows the distribution of valid times depending on the selected dimension for the ChuaCircuit and Windmi systems, using the OH approach.



(a) ChuaCircuit system

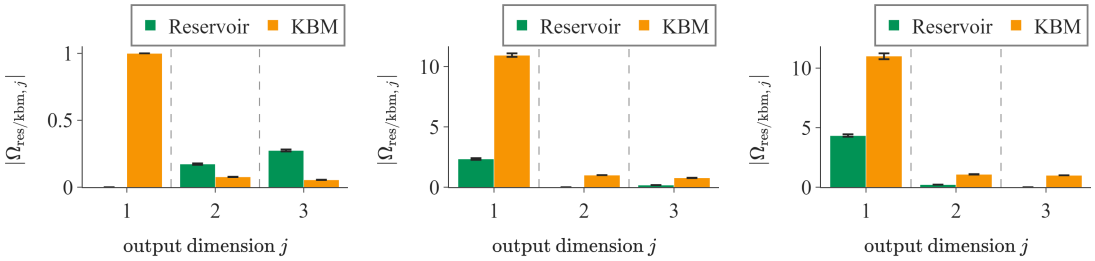
(b) Windmi system

Figure 6.9: Distribution of valid times for OH approach with $r_{\text{dim}} = 500$ depending on selected dimension l in the dimension-selection-model.

In both cases the choice of l leads to significant differences in the valid time distributions. As can be seen from the ChuaCircuit's flow Eq. (A.2), all of its nonlinearities are contained in the first dimension, $\dot{x}(t)$. It follows, that only the first dimension of the RK4 iterator, $x(t + \Delta t) = M_1(x(t), y(t), z(t))$, contains nonlinearities, while the other two dimensions, $M_2(x, y, z)$ and $M_3(x, y, z)$, are just linear functions of x, y, z. As a result, when choosing $K_{\text{ds}}(x, y, z) = M_1(x, y, z)$ as the knowledge-based model, all the complex nonlinearities are handled by the KBM, leaving the reservoir to learn only the linear functions $M_2(x, y, z)$ and $M_3(x, y, z)$. For the Windmi system, all the nonlinearities are contained in the third dimension, resulting in the higher valid times for $l = 3$. Although the effectiveness of the hybrid approaches appears to be optimal when the chosen dimension, l, corresponds to a dimension with high nonlinearity in the system, even if $K_{\text{ds}}(x, y, z) = M_l(x, y, z)$ is linear, the hybrid approaches can still be beneficial. This can be observed, as dimension $l = 2$ for both the ChuaCircuit and Windmi systems results in higher valid

times compared to the reservoir-only prediction shown in Fig. 6.8b, despite the corresponding KBMs, $K_{\mathrm{ds}}(x, y, z) = M_2(x, y, z)$, being linear.

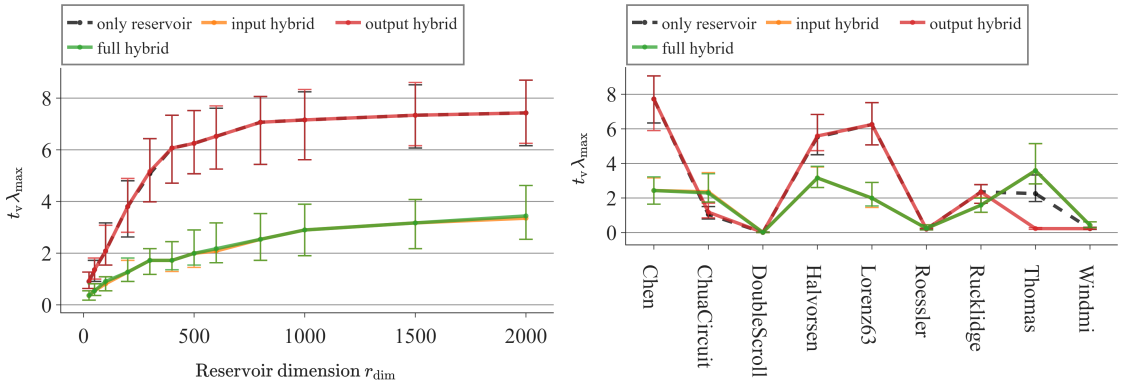
Fig. 6.10 illustrates the effect of the selected dimension, $l \in (1, 2, 3)$, on the output contributions, $\Omega_{\mathrm{res},j}$ and $\Omega_{\mathrm{kbm},j}$, in the OH approach when applied to the ChuaCircuit system. As expected, for all selected dimensions l, the reservoir-contribution $\Omega_{\mathrm{res},j=l}$ vanishes, as it is accurately covered by the KBM. Interestingly, even when the KBM is given by the second or third dimension of the ChuaCircuit's iterator, $M_{l=2}$ or $M_{l=3}$, a majority of the KBM-contributions are still attributed to the nonlinear dimension, $j = 1$.



(a) Selected dimension $l = 1$ (b) Selected dimension $l = 2$ (c) Selected dimension $l = 3$

Figure 6.10: Output contributions of OH approach with $r_{\mathrm{dim}} = 500$ using the dimension-selection-model for the ChuaCircuit system.

Sinus-model This paragraph investigates the sinus-model, which represents a highly inaccurate model containing no information about the dynamical system. The sinus-model is represented by $K_{\mathrm{sin}}(x, y, z) = [\sin(x), \sin(y), \sin(z)]^T$. As for the other KBMs, Fig. 6.11a shows the influence of the reservoir dimension, r_{dim}, on the prediction of the Lorenz system, while Fig. 6.11b depicts the valid times for all dynamical systems that were tested using a reservoir dimension of $r_{\mathrm{dim}} = 500$.



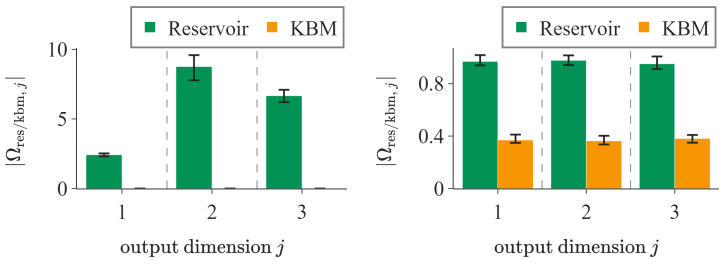
(a) Lorenz system (b) All systems with $r_{\mathrm{dim}} = 500$

Figure 6.11: Hybrid RC using the sinus-model.

In all cases, apart from the Thomas system, the OH approach is equivalent to the reservoir-only approach, and the IH approach is equivalent to the FH approach. Furthermore, only for the ChuaCircuit and Thomas systems do the IH and FH approaches perform better than the OH and reservoir-only approaches. This confirms the previous claim that when the model is highly inaccurate, the OH approach just reduces to the reservoir-only prediction. On the other hand, the IH and FH approaches, which incorporate the inaccurate model into the reservoir states $\boldsymbol{r}(t)$, fail to ignore the inaccurate model, and as a result, the prediction quality decreases. The exceptional behaviour for the Thomas system can be attributed to the fact that its flow equation, Eq. (A.7), includes the terms $\sin(x)$, $\sin(y)$, and $\sin(z)$. Therefore, in this case, incorporating the sinus-model K_{sin} actually introduces some relevant knowledge about the system.

This behaviour of the Thomas system is further supported by investigating the reservoir and KBM output contributions, $\Omega_{\mathrm{res},j}$ and $\Omega_{\mathrm{kbm},j}$, for the OH approach using a reservoir dimension

of $r_{\text{dim}} = 500$, as shown in Fig. 6.12b. In this case, it can be observed that the sinus-model makes a notable contribution to the output matrix, $\mathbf{W}_{\text{out}}$. Nevertheless, the sinus-model in the OH approach seems to generate unstable dynamics, as its valid times are close to zero, as seen in Fig. 6.11b. For the Lorenz system, where the sinus-model does not lead to differences between the OH approach and reservoir-only approach, the KBM-contribution, $\Omega_{\text{kbm},j}$, is insignificant, as shown in figure 6.12a.



(a) Lorenz system (b) Thomas system

Figure 6.12: Output contributions of OH approach with $r_{\text{dim}} = 500$ using the sinus-model.

Conclusion In summary, the *input hybrid*, *output hybrid*, and *full hybrid* approaches were evaluated for nine different chaotic dynamical systems using various artificially created knowledge-based models.

The ϵ-model with a lower model error was used as an accurate knowledge-based model that is able to approximate the next step in the time series, according to Eq. (6.1). It is important to note that for one choice of ϵ the accuracy of the knowledge-based model can vary depending on the chosen ϵ-parameter (Tab. 6.1) and the general characteristics of the individual dynamical systems. For these accurate ϵ-models, it was observed that all hybrid architectures showed a significant improvement in predictive qualities compared to the reservoir-only approach, with the *output hybrid* and *full hybrid* approaches standing out in the comparison over all dynamical systems. Generally, throughout all dynamical systems and KBMs tested, it was observed that the full-hybrid approach was always performing as good or better compared to the input-hybrid approach. This is because the FH approach reduces to the IH approach for highly inaccurate models and incorporates the strengths of the output-hybrid approach for highly accurate models. Furthermore, it was demonstrated that the IH approach requires higher reservoir dimensions, r_{dim}, in order to efficiently incorporate the KBM. In contrast, using the FH and OH approaches, a small reservoir is often sufficient in order to generate good predictions, compared to the reservoir-only approach. The only downside of the FH approach becomes apparent, when confronted with a highly inaccurate KBM, as demonstrated with the ϵ-model with $\epsilon = 100$, and with the artificially wrong sinus-model. In this case, as in the IH approach, the inaccurate KBM is incorporated into the reservoir states, $\boldsymbol{r}(t)$, which may diminish its predictive abilities. By using the OH approach this problem can be surpassed, which reduces to the reservoir-only prediction for highly inaccurate KBMs. In addition, the flow-model was used as a knowledge-based model that incorporates some knowledge about the dynamical system, without being able to approximate the next state directly. For most dynamical systems tested, the flow-model lead only to a marginal improvement of valid times when incorporated into the Hybrid approaches. Using the dimension-selection-model as the KBM, it was interesting to note that the hybrid approaches are most effective when the chosen dimension l covers most of the nonlinearity of the dynamical system.

Chapter 7

Summary

To summarize, this thesis considered the software-based approach to reservoir computing for the task of predicting three-dimensional chaotic dynamical systems. In most cases, the well-known Lorenz system was used as the testing data. In chapter 4, the RC approach that includes useful extensions such as node bias and input standardization, and an ensemble method to statistically assess its prediction quality was introduced. The influence of individual hyperparameters on the prediction quality was demonstrated in Fig. 4.5.

Chapter 5 analyzed the high-dimensional reservoir states, $\boldsymbol{r}(t)$, and corresponding ridge regression output-learning by using the reservoir state's principal components, according to the theory outlined in chapter 3. The reservoir state's explained variances, ϕ_i, were shown to provide a useful tool to assess the richness of the reservoir, as they measure the amount of linearly independent dynamics in the reservoir's nodes. The observed exponential decrease in ϕ_i confirmed the expected high correlations between the reservoir's nodes. The influence of the RC hyperparameters controlling the input-reservoir coupling on the explained variances, ϕ_i, demonstrated how some RC setups lead to more and others to less independent dynamics in the reservoir. The novel interpretation of ridge regression through principal components in the case of reservoir computing resulted in the definition of the *number of active components*, i_{co}, that approximately measures the number of principal components considered in the output of the reservoir. Interestingly, the lower end of the number of active components, i_{co}, was shown to correlate strongly with the corresponding prediction quality, regardless of the hyperparameter that caused the change in i_{co}. This is a remarkable result, as the number of active components appears to explain the increase in prediction quality previously observed in Fig. 4.5 with an increase in reservoir dimension, r_{dim}, input strength, σ, spectral radius, ρ_0, average node degree, d, and with a decrease in the regularization parameter, β. Additionally, the meaning of the individual principal components was investigated in terms of their memory capacities. Here, it was shown that the first few principal components are primarily influenced by the reservoir's immediate input, $\boldsymbol{x}(t)$, while the later components have a stronger association with the reservoir's memory. For possible further studies utilizing the combinations of PCA and RC, view section 5.4.

For the second topic of this thesis, presented in chapter 6, the physics-informed hybrid RC approach introduced by Pathak et al. [10] was further explored. In this hybrid approach a knowledge-based model is connected either to the input (*input hybrid*) or output (*output hybrid*) of the reservoir, or to both in the *full hybrid* approach. It has been demonstrated that in situations where the knowledge-based model is relatively accurate, the use of hybrid approaches can significantly improve the reservoir predictions. Among the hybrid approaches, the *output hybrid* and *full hybrid* approaches showed particularly exceptional results across all three-dimensional chaotic systems tested. Providing the flow of the underlying dynamical system as the knowledge-based model also resulted in improved prediction qualities compared to the reservoir-only approach, although the improvements were smaller. Additionally, the knowledge-based model, when represented by only one dimension of the true system's iterator, was found to be most effective in assisting the reservoir when handling the difficult nonlinear dimensions. It has been demonstrated that for highly inaccurate knowledge-based models, the *output hybrid* approach defaults to the reservoir-only approach, while the prediction quality in the *input hybrid* and *full hybrid* approaches may be

undermined.

For both topics in this thesis, PCA-explainability and hybrid RC, further research could be conducted with a broader range of test data, such as higher-dimensional, noisy, and especially real-world data.

Appendix A

More three-dimensional chaotic dynamical systems

All system equations, default parameters, initial points, and maximal Lyapunov exponents are taken from [12].

Chen's system

$$\begin{aligned}\dot{x}(t) &= a(y-x)\\ \dot{y}(t) &= (c-a)x - xz + cy\\ \dot{z}(t) &= xy - bz\end{aligned} \tag{A.1}$$

Parameter values	Initial point $\boldsymbol{u}(0)$	Time step Δt	MLE $\lambda_{\max}$
$a = 35, b = 3, c = 28$	$[-10, 0, 37]$	0.03	2.0272

Chua's circuit

$$\begin{aligned}\dot{x}(t) &= \alpha[y - x + bx + 0.5(a-b)(|x+1| - |x-1|)]\\ \dot{y}(t) &= x - y + z\\ \dot{z}(t) &= -\beta y\end{aligned} \tag{A.2}$$

Parameter values	Initial point $\boldsymbol{u}(0)$	Time step Δt	MLE $\lambda_{\max}$
$\alpha = 9, \beta = 100/7, a = 8/7, b = 5/7$	$[0, 0, 0.6]$	0.2	0.3271

Double scroll

$$\begin{aligned}\dot{x}(t) &= y\\ \dot{y}(t) &= z\\ \dot{z}(t) &= -a[z + y + x - \operatorname{sgn}(x)]\end{aligned} \tag{A.3}$$

Parameter values	Initial point $\boldsymbol{u}(0)$	Time step Δt	MLE $\lambda_{\max}$
$a = 0.8$	$[0.01, 0.01, 0]$	0.03	0.0497

Halvorsen's cyclically symmetric attractor

$$\begin{aligned}\dot{x}(t) &= -ax - 4y - 4z - y^2 \\ \dot{y}(t) &= -ay - 4z - 4x - z^2 \\ \dot{z}(t) &= -az - 4x - 4y - x^2\end{aligned} \tag{A.4}$$

Parameter values	Initial point $\boldsymbol{u}(0)$	Time step Δt	MLE $\lambda_{\max}$
$a = 1.27$	$[-5, 0, 0]$	0.05	0.7899

Rössler attractor

$$\begin{aligned}\dot{x}(t) &= -y - z \\ \dot{y}(t) &= x + ay \\ \dot{z}(t) &= b + z(x - c)\end{aligned} \tag{A.5}$$

Parameter values	Initial point $\boldsymbol{u}(0)$	Time step Δt	MLE $\lambda_{\max}$
$a = b = 0.2, c = 5.7$	$[-9, 0, 0]$	0.2	0.0714

Rucklidge attractor

$$\begin{aligned}\dot{x}(t) &= -\kappa x + \lambda y - yz \\ \dot{y}(t) &= x \\ \dot{z}(t) &= -z + y^2\end{aligned} \tag{A.6}$$

Parameter values	Initial point $\boldsymbol{u}(0)$	Time step Δt	MLE $\lambda_{\max}$
$\kappa = 2.0, \lambda = 6.7$	$[1, 0, 4.5]$	0.1	0.0643

Thomas' cyclically symmetric attractor

$$\begin{aligned}\dot{x}(t) &= -bx + \sin(y) \\ \dot{y}(t) &= -by + \sin(z) \\ \dot{z}(t) &= -bz + \sin(x)\end{aligned} \tag{A.7}$$

Parameter values	Initial point $\boldsymbol{u}(0)$	Time step Δt	MLE $\lambda_{\max}$
$b = 0.18$	$[0.1, 0, 0]$	0.4	0.0359

WINDMI attractor

$$\begin{aligned} \dot{x}(t) &= y \\ \dot{y}(t) &= z \\ \dot{z}(t) &= -az - y + b - \exp(x) \end{aligned} \tag{A.8}$$

Parameter values	Initial point $\boldsymbol{u}(0)$	Time step Δt	MLE $\lambda_{\max}$
$a = 0.7, b = 2.5$	$[0, 0.8, 0]$	0.2	0.0755

Appendix B

Effect of ridge regression type in RC

In the theory presented in section 3.2, it was demonstrated that type-b ridge regression, which incorporates an unregularized intercept $\boldsymbol{w}$, is in principle superior compared to type-a (no intercept) and type-c (regularized intercept) ridge regression. However, in section 4.6, it was observed that the choice of ridge regression type (a, b, or c) had a negligible effect on the prediction of the Lorenz system. In this appendix, it is demonstrated, that the equivalence of the three RR types previously shown is due to the relatively small mean of the Lorenz data. When the Lorenz data is modified by adding a large offset, $\boldsymbol{u}(t) \to \boldsymbol{u}(t) + \delta$, the superiority of type-b RR becomes apparent.

As mentioned in section 4.4, the calculation of the valid time using the previously defined over-time error, $e(t)$ from Eq. (4.20), fails if the output data, $\boldsymbol{y}(t)$, has a very large mean. Consequently, in this experiment where a large offset is added to the Lorenz data, the over-time error must be modified to:

$$e(t) = \frac{\|\boldsymbol{y}(t) - \boldsymbol{y}_{\mathrm{r}}(t)\|}{\langle \|\boldsymbol{y}(t) - \langle \boldsymbol{y}(t) \rangle\|^2 \rangle^{1/2}} \tag{B.1}$$

The over-time error in Eq. (B.1) is now invariant under offsets in the data. The resulting valid times for the three RR types, as a function of the data offset in the range $\delta = 0$ to $\delta = 600$, are shown in Fig. B.1. It can be seen that only in the case of type-b ridge regression does the prediction quality remain invariant under the data offset, δ.

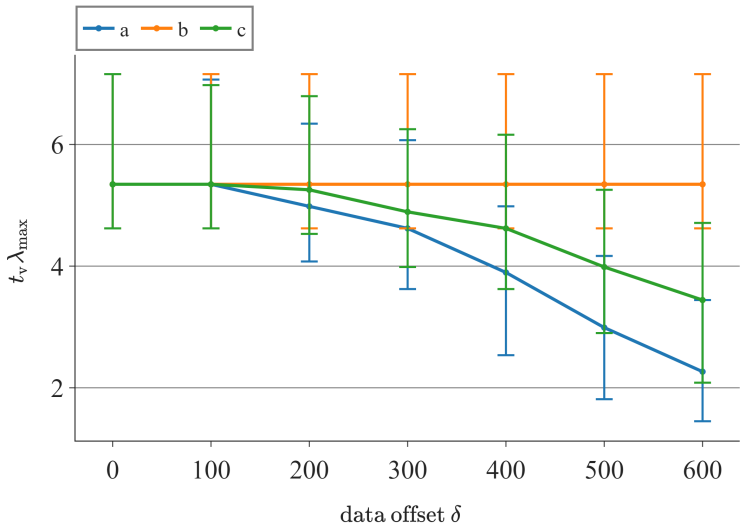


Figure B.1: The median and lower/upper quartiles of the valid times for the default RC setup in predicting the Lorenz system where each dimension includes an offset of δ, for the three RR types.

Bibliography

[1] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10):2318–2331, 2017.

[2] Zhixin Lu, Brian R Hunt, and Edward Ott. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):061104, 2018.

[3] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[4] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[5] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[6] Guy Van der Sande, Daniel Brunner, and Miguel C Soriano. Advances in photonic reservoir computing. *Nanophotonics*, 6(3):561–576, 2017.

[7] Chrisantha Fernando and Sampsa Sojakka. Pattern recognition in a bucket. In *European conference on artificial life*, pages 588–597. Springer, 2003.

[8] Dennis Valbjørn Christensen, Regina Dittmann, Bernabe Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, et al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2022.

[9] Claudio Gallicchio and Alessio Micheli. A markovian characterization of redundancy in echo state networks by pca. In *ESANN*, 2010.

[10] Jaideep Pathak, Alexander Wikner, Rebeckah Fussell, Sarthak Chandra, Brian R Hunt, Michelle Girvan, and Edward Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):041101, 2018.

[11] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[12] Julien Clinton Sprott and Julien C Sprott. *Chaos and time-series analysis*, volume 69. Oxford university press Oxford, 2003.

[13] Carl Runge. Über die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, 1895.

[14] Wilhelm Kutta. Beitrag zur naherungsweisen integration totaler differentialgleichungen. *Z. Math. Phys.*, 46:435–453, 1901.

[15] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning: data mining, inference, and prediction. springer series in statistics. *Springer New York*, 2009.

[16] Andreĭ Nikolaevich Tikhonov, Vasilij Ja Arsenin, Vasiliĭ IAkovlevich Arsenin, Vasiliy Y Arsenin, et al. *Solutions of ill-posed problems*. Vh Winston, 1977.

[17] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[18] Jaideep Pathak, Zhixin Lu, Brian R Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[19] Zhixin Lu, Jaideep Pathak, Brian Hunt, Michelle Girvan, Roger Brockett, and Edward Ott. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(4):041102, 2017.

[20] Thomas L Carroll and Louis M Pecora. Network structure effects in reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(8):083130, 2019.

[21] Alexander Haluszczynski and Christoph 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*, 29(10):103143, 2019.

[22] Qianli Ma, Lifeng Shen, and Garrison W Cottrell. Deep-esn: A multiple projection-encoding hierarchical reservoir computing framework. *arXiv preprint arXiv:1711.05255*, 2017.

[23] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120(2):024102, 2018.

[24] Sebastian Baur and Christoph Räth. Predicting high-dimensional heterogeneous time series employing generalized local states. *Physical Review Research*, 3(2):023215, 2021.

[25] Joschka Herteux and Christoph Räth. Breaking symmetries of the reservoir equations in echo state networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(12):123142, 2020.

[26] Alberto Racca and Luca Magri. Robust optimization and validation of echo state networks for learning chaotic dynamics. *Neural Networks*, 142:252–268, 2021.

[27] Alexander Haluszczynski, Jonas Aumeier, Joschka Herteux, and Christoph Räth. Reducing network size and improving prediction stability of reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(6):063136, 2020.

[28] Jason A Platt, Adrian Wong, Randall Clark, Stephen G Penny, and Henry DI Abarbanel. Robust forecasting using predictive generalized synchronization in reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(12):123118, 2021.

[29] Jason A Platt, Stephen G Penny, Timothy A Smith, Tse-Chun Chen, and Henry DI Abarbanel. A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics. *Neural Networks*, 153:530–552, 2022.

[30] Peter Steiner, Azarakhsh Jalalvand, and Peter Birkholz. Cluster-based input weight initialization for echo state networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[31] Wei Miao, Vignesh Narayanan, and Jr-Shin Li. Interpretable design of reservoir computing networks using realization theory. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[32] Sigurd Løkse, Filippo Maria Bianchi, and Robert Jenssen. Training echo state networks with regularization through dimensionality reduction. *Cognitive Computation*, 9(3):364–378, 2017.

[33] Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, and Themistoklis Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5):e0197704, 2018.

[34] Andrew J Majda. Challenges in climate science and contemporary applied mathematics. *Communications on Pure and Applied Mathematics*, 65(7):920–948, 2012.

[35] Nguyen Anh Khoa Doan, Wolfgang Polifke, and Luca Magri. Physics-informed echo state networks for chaotic systems forecasting. In *International Conference on Computational Science*, pages 192–198. Springer, 2019.

[36] NAK Doan, W Polifke, and L Magri. Short-and long-term predictions of chaotic flows and extreme events: a physics-constrained reservoir computing approach. *Proceedings of the Royal Society A*, 477(2253):20210135, 2021.

[37] Alberto Racca and Luca Magri. Automatic-differentiated physics-informed echo state network (api-esn). In *International Conference on Computational Science*, pages 323–329. Springer, 2021.

[38] Alexander Wikner, Jaideep Pathak, Brian Hunt, Michelle Girvan, Troy Arcomano, Istvan Szunyogh, Andrew Pomerance, and Edward Ott. Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(5):053111, 2020.

[39] Alexander Wikner, Jaideep Pathak, Brian R Hunt, Istvan Szunyogh, Michelle Girvan, and Edward Ott. Using data assimilation to train a hybrid forecast system that combines machine-learning and knowledge-based components. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(5):053114, 2021.

[40] Yslam D Mammedov, Ezutah Udoncy Olugu, and Guleid A Farah. Weather forecasting based on data-driven and physics-informed reservoir computing models. *Environmental Science and Pollution Research*, 29(16):24131–24144, 2022.

Acknowledgement

First of all, I would like to express my profound gratitude to my supervisor Christoph for giving me the opportunity to work on this interesting topic, and being supportive and encouraging of all my ideas. I am also very grateful to Sebastian for the interesting conversations and for his unwavering support in both scientific and IT matters. Lastly, I am thankful for my friends and family, especially to my flatmate Lukas, who provided me with a constant source of motivation and support.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet zu haben.

Ort, Datum Dennis Duncan