Aalto University
School of Science

Master’s Programme in Life Science Technologies

Reservoir Computing in Predicting Networks of Coupled Dynamical Systems

Juho Tuomaala

Master’s Thesis
2024

Aalto University
School of Science

Author Juho Tuomaala

Title Reservoir Computing in Predicting Networks of Coupled Dynamical Systems

Degree programme Master's Programme in Life Science Technologies

Major Complex Systems

Supervisor Riku Linna

Advisor Riku Linna

Date 23.04.2024 **Number of pages** 56 **Language** English

Abstract

Networks of coupled dynamical systems are high dimensional dynamical systems that consist of interlinked dynamical subsystems. These systems can exhibit highly complicated dynamics and emergent, system-level behaviour, such as synchronisation. Coupled dynamical systems have been used to model diverse natural phenomena from flashing fireflies to Josephson junctions.

Reservoir computing and specifically echo state network (ESN) models have proven to be effective machine learning models in predicting nonlinear and chaotic dynamical systems. A network scheme of parallel ESN's presented by Keshav et al. [48], specifically tailored to predict coupled dynamical systems, has been implied to show far superior prediction performance compared to a single ESN model in predicting these systems.

The primary goal of this thesis is to replicate and extend these findings by comparing the parallel and the single ESN models. The specific dynamical systems considered in this work are the linked Kuramoto system and the linked Lorenz system. The parallel and single models are tested on both of these systems, and compared in terms of prediction performance, model optimisation and computational aspects.

An additional topic of interest in this work is the effect of numerical integration method on the prediction task. In the field of reservoir computing, the training and testing data is typically generated via numerical integration, yet the choice of integration algorithms and parameters are seldom discussed in detail.

In this thesis, the parallel ESN model is shown to outperform a single ESN in predicting networks of dynamical systems, in addition to bearing computational advantages. However, the difference in performance is not shown to be as large as suggested by previous publications. In addition, varying the settings of numerical integration is shown to have a meaningful effect on the properties of the simulated data and consequently on the machine learning task results.

Keywords machine learning , reservoir computing , nonlinear dynamics , chaos , time series prediction , echo state network , numerical integration

Aalto-yliopisto
Perustieteiden
korkeakoulu

Tekijä	Juho Tuomaala
Työn nimi	Varantolaskenta kytkettyjen dynaamisten systeemien verkkojen ennustamisessa
Koulutusohjelma	Master's Programme in Life Science Technologies
Pääaine	Complex Systems
Työn valvoja	Riku Linna
Työn ohjaaja	Riku Linna
Päivämäärä 23.04.2024	**Sivumäärä** 56 **Kieli** englanti

Tiivistelmä

Kytkettyjen dynaamisten systeemien verkot ovat korkeaulotteisia dynaamisia systeemeitä, jotka koostuvat toisiinsa kytketyistä dynaamisista alisysteemeistä. Nämä järjestelmät voivat ilmentää erittäin monimutkaista dynamiikkaa ja emergenttiä, järjestelmätason käyttäytymistä, kuten synkronisaatiota. Kytkettyjä dynaamisia systeemeitä käytetään mallintamaan monenlaisia luonnonilmiöitä välkkyvistä tulikärpäsistä Josephsonin liitoksiin.

Varantolaskenta- (reservoir computing) ja erityisesti echo state network (ESN) -mallit ovat osoittautuneet tehokkaiksi koneoppimistyökaluiksi ennustettaessa epälineaarisia ja kaaottisia dynaamisia systeemeitä. Aiemmassa julkaisussa Keshav et al. [48] esittämä rinnakkaisten ESN-mallien verkko on erityisesti kytkettyjen dynaamisten systeemien ennustamiseen suunniteltu malli, jonka on näytetty suoriutuvan huomattavasti yksittäistä ESN-mallia paremmin näiden systeemien ennustamisesta.

Tämän diplomityön ensisijainen tavoite on toistaa ja laajentaa aiempia havaintoja vertaamalla rinnakkaista ja yksittäistä ESN-mallia. Tässä työssä tarkasteltavat dynaamiset järjestelmät ovat linkitetty Kuramoto-systeemi ja linkitetty Lorenz-systeemi. Sekä rinnakkaista että yksittäistä ESN-mallia testataan kummankin systeemin ennustamisessa ja niitä verrataan niin ennustustehon, optimoinnin kuin laskennallistenkin ominaisuuksien näkökulmista.

Toinen keskeinen aihe tässä työssä on numeeristen integrointimenetelmien vaikutus itse koneoppimisongelmaan. Varantolaskentaa käsittelevässä tutkimuksessa koulutus- ja testidata tuotetaan usein numeerisella integroinnilla, mutta käytettyjen algoritmien ja niiden parametrien valintaa käsitellään harvoin yksityiskohtaisesti.

Tässä työssä rinnakkaisen ESN-mallin osoitetaan suoriutuvan paremmin kuin yksittäinen ESN-malli ennustettaessa dynaamisten systeemien verkostoja, samalla tarjoten laskennallisia etuja. Havaittu suorituskyvyn ero ei kuitenkaan näyttäydy yhtä suurena kuin aiemmat julkaisut antavat ymmärtää. Lisäksi numeerisen integroinnin algoritmien ja asetusten valinnan osoitetaan vaikuttavan merkityksellisesti simuloidun koulutus- ja testidatan ominaisuuksiin ja siten itse koneoppimisongelman tuloksiin.

Avainsanat koneoppiminen , varantolaskenta, epälineaarinen dynamiikka, kaaos, aikasarjaennuste, numeerinen integrointi

Contents

1 Symbols and Abbreviations

Symbols

Δt	numerical integration timestep
$\mathbf{J}$	Jacobian matrix
D_H	Hausdorff dimension
D_{KY}	Kaplan-Yorke dimension
λ_{max}	largest Lyapunov exponent
$\mathbf{A}$	adjacency matrix
$\langle d \rangle$	network average degree
N	number of nodes in the underlying system network
$\mathbf{W}^{in}$	reservoir input adjacency matrix
$\mathbf{W}$	reservoir internal adjacency matrix
$\mathbf{W}^{out}$	reservoir output adjacency matrix
N_{in}	input vector length
N_r	size of reservoir internal network
N_{out}	output vector length
N_{res}	number of reservoirs in the parallel ESN model
$\mathbf{u}(t)$	reservoir input vector
$\mathbf{r}(t)$	reservoir internal state vector
$\mathbf{y}(t)$	reservoir output vector
ρ	reservoir network spectral radius
σ	reservoir input scaling parameter
β	leak rate parameter
λ	base 10 logarithm of the ridge parameter
k	average degree of $\mathbf{W}$
$E(t)$	normalised prediction error
f	prediction error threshold
$\lambda_{max}t$	Lyapunov time
O	computation time complexity

Abbreviations

RNN	recurrent neural network
LSTM	long short-term memory
GRU	gated recurrent unit
ESN	echo state network
RK	Runge-Kutta method
SD	standard deviation

2 Introduction

The field of dynamics, established by Newton in the 1600's, studies systems that evolve through time, space, or with respect to other pertinent variables. When modeling some natural phenomenon in terms of mathematics, arguably the most crucial benefit of the effort is the ability to predict the system's future state.

In the real world, we obtain information of dynamical systems first through measurement and data. Without the exact equations governing the underlying system, predicting the system's future involves choosing and fitting a model based on the data only. Often the choice is a linear model - assuming linear dependency for different quantities of the system greatly simplifies both the model definition and the fitting process. However, many dynamical systems in nature are in fact nonlinear, that is, the system's output is not linearly proportional to its input, and the state of the system at any given moment is not merely a linear combination of its parts.

To address nonlinearity of the underlying system the predicting model itself should contain nonlinear elements. A common approach for predicting time series with nonlinear origin is choosing a recurrent neural network (RNN) based model. For the general sequence learning task, including time series prediction problems, the most popular model choices include Long short-term memory (LSTM) [20] and gated recurrent unit (GRU) models [6]. Both are flexible deep learning models capable of learning a wide variety of sequence learning problems, trained via backpropagation. While in the recent times training machine learning models using backpropagation has become computationally feasible even on regular desktop computers, RNN models continue to suffer from vanishing or exploding gradients, granted that the previously mentioned models (LSTM, GRU) are designed to mitigate these problems and succeed in it to a notable extent.

Another, arguably much simpler way to avoid the problems intrinsic to RNN's and backpropagation is to leave the recurrent connection weights of the RNN untouched after initialisation and only modifying the vector of output weights during training. This change in design converts the training process into a linear optimisation problem, allowing for great computational simplicity and stability. This is the key idea behind Echo state network (ESN) models, originally introduced by Herbert Jaeger in 2001 [22]. Despite the model's simplicity, ESN's have been shown to perform well in time series prediction tasks, especially in predicting chaotic dynamical systems.

A defining characteristic of chaos is that the system exhibits sensitive dependency on initial conditions [49]. This sets a stringent limit on how accurately the system dynamics can be reconstructed based on experimental measurement data. Consequently, accurate long-term prediction is rendered impossible. This does not, however, make attempts at short-term predicting pointless. The field of reservoir computing and chaotic dynamics prediction is focused on two key aspects: accurate short-term prediction and long-term replication of the systems "climate". The latter means that the predicted time series is qualitatively similar to the original system and has similar ergodic properties, such as maximum Lyapunov exponent and fractal dimension. Especially when considering the previously mentioned computational advantages, reservoir computing models have demonstrated notable efficacy when addressing these tasks [3, 14, 41].

In the majority of publications, the reservoir computing models are tested on widely studied chaotic systems, such as the Lorenz system [35]. These systems are typically relatively low-dimensional, and may have analytically verified values for fractal dimension and other relevant properties. In this thesis, networks of linked dynamical systems were studied. The network of such consists of multiple subsystems, the dynamical equations of which are coupled with their neighbouring subsystems. In addition to the high dimensionality, the emergent collective behaviour of the subsystems brings adds another layer of complexity. Coupled dynamical systems are often used to model complex natural phenomena, such as the synchrony of flashing fireflies [5] or neural networks in the brain [4, 24, 52].

In addition to a traditional ESN model, a network of parallel reservoirs is studied in this thesis. The parallel reservoir scheme, introduced in a paper by Keshav et al. [48] is designed specifically for predicting networks of dynamical systems: in the parallel reservoir scheme, a network of small ESN's is set up. Each ESN is assigned with a task of learning and predicting the corresponding subsystem in the underlying system network, utilising the outputs of adjacent ESN's. The performance, advantages and shortcomings of both models are analysed and compared in this thesis.

A secondary topic studied in this thesis is the effect of numerical integration methods on the simulated dynamical system data. In the field of reservoir computing, there is no standard procedure for the data generation and which integration methods to use - in fact, the settings used for simulating the data vary from publication to publication, and the justification for the method selection is often left shallow. In this thesis we illustrate how different integration algorithms and integration time steps affect the observed dynamics and ergodic properties of the simulated data, and how this translates into differences in the prediction performance.

3 Nonlinear Dynamical Systems

A dynamical system is one whose state evolves according to some given function. Depending on the system, this state can depend on one or more variables, such as time or position. Dynamical systems are often represented by either differential equations or iterated maps, corresponding to whether time is treated as a continuous or discrete variable, respectively.

The focus of this Thesis is on dynamical systems described by ordinary differential equations, where equations only depend on a single variable, namely time in our case. In the study of dynamical systems, a typical definition for such a system is written as

$$\begin{aligned}\dot{x}_1 &= f_1(x_1, \ldots x_n),\\ &\vdots\\ \dot{x}_i &= f_i(x_1, \ldots x_n),\\ &\vdots\\ \dot{x}_n &= f_n(x_1, \ldots x_n),\end{aligned} \tag{3.1}$$

where $\dot{x}_i \equiv \frac{dx_i}{dt}$ and n is the dimension of the system [49]. A dynamical system is nonlinear if it does not satisfy the principles of homogeneity and superposition, that is, the output of the system is not directly proportional to its input[15, 49]. Analytical solution of nonlinear differential equations is often extremely hard or completely impossible. Hence, their analysis is in practise limited to graphical methods and numerical approximations.

3.1 Numerical Simulation of Dynamical Systems

In practice, the solutions for nonlinear dynamical systems are approximated via numerical methods on a computer. The simplest method for numerically integrating ordinary differential equations is the Euler method. For a one dimensional system defined as in Eq. (3.1), the formula for the method is

$$x_{n+1} = x_n + f(x_n)\Delta t, \tag{3.2}$$

where Δt is some relatively short distance in time, that is, the time step of the algorithm. Given some initial point x_0, the Euler method iteratively advances to the next estimated point Δt away from the previous by evaluating the derivative at the starting point of the current iteration.[44]

Evaluating the derivative only in the previous point, of course, causes error in the approximated next point, since the derivative may change during Δt. The error can, however, be reduced by first taking a trial step only to evaluate the derivative, and using the average derivative of the start and the trial point when taking the actual integration step:

$$\begin{aligned} \tilde{x}_{n+1} &= x_n + f(x_n)\Delta t \\ x_{n+1} &= x_n + \frac{1}{2}(f(x_n) + f(\tilde{x}_{n+1})\Delta t \end{aligned} \tag{3.3}$$

In the above equations, $\tilde{x}_{n+1}$ stands for the trial step and x_{n+1} for the actual integration step. This improved Euler method by its symmetrisation cancels the first-order error term produced by the previous method, and would hence be classified as second-order Runge-Kutta method[44, 49].

The idea of multiple intermediate calculations for each iteration step can be extended much further while still maintaining a favorable balance between computational cost and accuracy. Nowadays perhaps the most widely used numerical integration algorithm is the Dormand-Prince method, originally presented in [7], which executes the integration steps using fifth-order Runge-Kutta methods, but manages the error assuming the accuracy of a fourth-order Runge-Kutta method. The Dormand-Prince method is the current standard algorithm in various ordinary differential equation solver tools, such as MATLAB's ODE solver and Python's Scipy library[53].

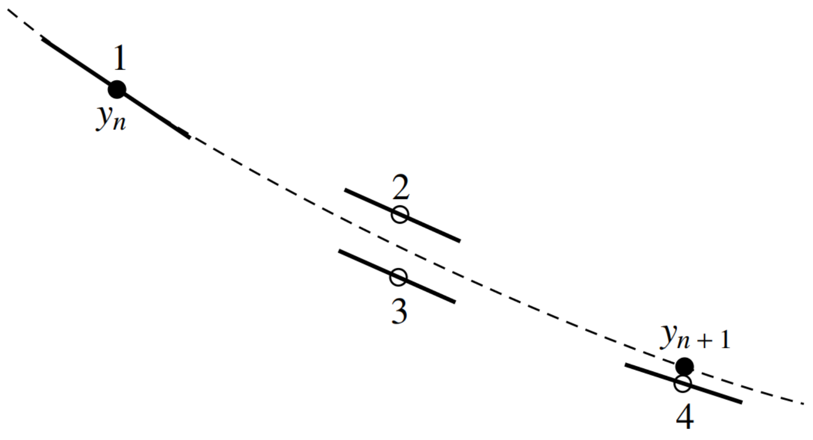


Figure 3.1: Fourth-order Runge-Kutta method, where for each iteration, the derivative is evaluated at four different points. Figure 17.1.3. adapted from [44], p.909.

While the Runge-Kutta methods are suitable for solving a great number of dynamical systems, they may fail when encountering so called stiff systems. Solving stiff systems with explicit numerical integration methods (such as the ones previously presented) can yield unstable solutions unless a very short step size is used. A dynamical system may show stiffness if, for example, the solution involves components decaying at highly varying rates, or if neighbouring solutions approach the desired solution at a rate higher in comparison to the rate at which the solution value itself varies (in some interval)[34]. Solving stiff dynamical systems with reasonable integration step size requires implicit numerical integration methods. In contrast to the explicit methods, where the next iteration x_{n+1} is given explicitly in terms of x_n, implicit methods estimate the next point, as the name suggests, in implicit terms. The simplest implicit integration method is the backward Euler method:

$$x_{n+1} = x_n + f(x_{n+1})\Delta t. \tag{3.4}$$

In practise, evaluating Eq. (3.4) often requires approximating the term $f(x_{n+1})$ numerically with, for example, Newton's method. As with the explicit Runge-Kutta methods, higher order implicit methods have been developed as well. The extra computational cost resulting from the implicit nature of the methods is worth the effort when solving stiff systems.

Depending on the dynamical system to be solved, the choice of a numerical integration algorithm and its parameters can have a significant impact on the solution and its properties. This will be further discussed in Section 5.

3.2 Chaos

Although more rigorously defined than the word "chaos" in its everyday use, a chaotic system is yet to have a precise, universally accepted mathematical definition. There are, however, certain characteristics that chaotic systems share. As defined by Strogatz in [49], a chaotic system has the following attributes:

1. The system exhibits aperiodic long-term behaviour: The term 'aperiodic long-term behavior' refers to trajectories that don't gravitate towards fixed points, periodic orbits, or quasiperiodic orbits as $t \to \infty$.

2. The system is deterministic, meaning that the system is devoid of random or noisy inputs and parameters. Instead of arising from chaotic or noisy driving forces, the irregular behavior in the system emerges from its nonlinear characteristics.

3. The system's evolution is sensitive to its initial conditions: 'Sensitive dependence on initial conditions' refers to the phenomenon where closely aligned trajectories diverge at an exponential rate. In other words, the system exhibits a positive Lyapunov exponent (defined in Section 3.3).

3.3 Lyapunov Exponent

A system's sensitivity to initial conditions is quantified by the notion of Lyapunov exponent. More precisely: given a point in the phase space of the system $\mathbf{x}(t)$ at time t, and another point $\mathbf{x}(t) + \delta(t)$, where δ is a vector separating the two points, starting at some small initial length $||\delta_{\mathbf{0}}||$. As the trajectories starting from the two points evolve in time, the length of the separating vector δ grows according to

$$||\sigma(t)|| \sim ||\delta_0||e^{\lambda t},$$

where δ is called the Lyapunov exponent.

In higher dimensions, rates of divergence corresponding to the projections of the separation vector along independent directions results in a spectrum of Lyapunov exponents

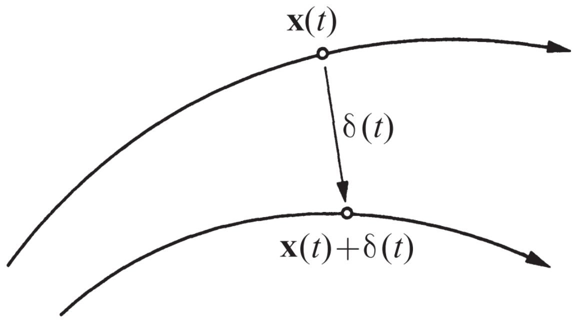


Figure 3.2: Illustrating the diverging trajectories and the separating vector. Adapted from [49], p.328.

$$\{\lambda_1, \lambda_2, ..., \lambda_n\},$$

where n is equal to the number of degrees of freedom. Mathematically the Lyapunov spectrum is defined as follows:

Let $\mathbf{J}$ denote The Jacobian matrix of the dynamical system, defined in Eq. (3.1), is

$$\mathbf{J}_{ij}(t) = \left.\frac{\partial f_i}{\partial x_j}\right|_t .$$

The matrix $\mathbf{J}$ defines the evolution of the tangent vectors starting from $\mathbf{x}(t)$ at some time t. By conditions given by the Oseledec theorem [40], the following limit exists:

$$\lim_{t\to\infty} \frac{1}{2t} \log(J^T J) = \Lambda, \tag{3.5}$$

where the eigenvalues of Λ are the Lyapunov exponents of the dynamical system [8].

Despite the concept's simplicity, determining the Lyapunov exponents for complex dynamical systems analytically often proves to be futile. Instead, numerical methods are employed to estimate the Lyapunov spectra of them. A widely used numerical method orginally proposed by Benettin et al. [1] and Shimada and Nagashima [47], was adapted and utilised for the purposes of this thesis.

In each iteration step, the algorithm operates by first numerically integrating the system starting from some initial state, and similarly calculating m trajectories with orthogonal perturbations to the initial state, where m is the number of Lyapunov exponents to be calculated. To retain the orthogonality of the perturbed trajectories for the next iteration, QR decomposition is applied to them. The amount of stretching or shrinking resulting from the QR decomposition for each orthogonal direction provides information about the respective Lyapunov exponent. The average over all iterations of the logarithm of the shrinking or stretching for each m directions is the corresponding estimated Lyapunov exponent.

3.4 Fractal Dimension

Fractal dimension is a metric used to characterise the spatial complexity of patterns, curves or shapes. The term is most notably applied in the study of fractals, characteristics of which are not adequately covered by traditional geometric or topological measures. A simple, often used example of such a pattern is the Koch snowflake, originally presented in [29]. The curve outlining this snowflake can be constructed by starting from an equilateral triangle, drawing new equilateral triangles spiking from each of the first triangle's sides, and repeating the process ad infinitum for each straight line segment in the newly formed shape. The first four iterations are shown in figure 4.1.

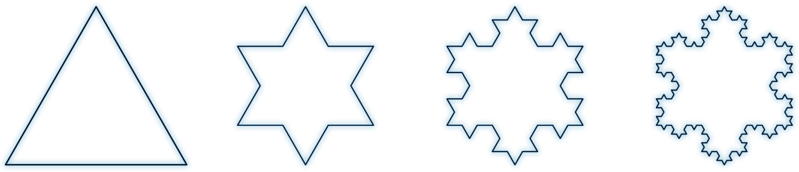


Figure 3.3: The first four iterations of the Koch snowflake.

The topological dimension of the Koch curve is 1, it has no area, yet the distance between any two points on the curve is infinite. Hence the curve can be thought as being somewhere between a line and a surface, and its fractal dimension would therefore lie between 1 and 2.

There are various mathematical definitions for fractal dimension, most of which agree on the fractal dimension of smooth curves or surfaces (that is, when the fractal dimension is equal to its topological dimension), but are not necessarily equal for rough, non-differentiable curves and shapes. The Hausdorff dimension, introduced by Felix Hausdorff in [17], is one of the classic definitions of fractal dimension.

Here, we follow the derivation of the Hausdorff dimension as given by Falconer [9]. We start by defining the Hausdorff measure. Given an non-empty subset U of n-dimensional Euclidean space $\mathbb{R}^n$, the diameter, or the greatest distance between any two points inside it is $|U| = \sup\{|x-y| : x, y \in U\}$. For a finite set of at most δ-diameter sets $\{U_i\}$ that each cover another set F, that is, $F \subset \bigcup_{i=1}^{\infty} U_i$, it can be said that the set $\{U_i\}$ is a δ-cover of F. We define, for some non-negative number s,

$$\mathcal{H}^s_\delta(F) = \inf\left\{\sum_{i=1}^{\infty} |U_i|^s : \{U_i\} \text{ is a } \delta\text{-cover of } F\right\}. \tag{3.6}$$

As δ decreases, the number of possible sets covering F decreases and thus the above increases towards a limit. The limit

$$\mathcal{H}^s(F) = \lim_{\delta\to 0} \mathcal{H}^s_\delta(F) \tag{3.7}$$

is the s-dimensional Hausdorff measure of the set F. The Hausdorff dimension of the set F can then be defined as

$$D_H = \inf\{s \geq 0 : \mathcal{H}^s(F) = 0\} = \sup\{s : \mathcal{H}^s(F) = \infty\}. \tag{3.8}$$

In essence, the Hausdorff dimension D_H is the critical value of s for which $\mathcal{H}^s(F)$ drops from ∞ to 0 as s increases.

The Hausdorff dimension of dynamical systems can be estimated with the Kaplan-Yorke dimension. The Kaplan-Yorke dimension, also known as the Lyapunov dimension, approximates the Hausdorff dimension by utilising the system's Lyapunov exponents. The dimension measure was first conjectured by Kaplan and Yorke [11, 26], and later developed and justified in various publications, see e.g. [33].

The conjecture by Kaplan and Yorke can be stated as follows: Given an ordered set containing the Lyapunov exponents of a dynamical system $\{\lambda_1, \lambda_2, ..., \lambda_n\}$, λ_1 being the biggest of the exponents, the Kaplan-Yorke dimension of the system is given by

$$D_{KY} = j + \frac{\sum_{i=1}^{j} \lambda_i}{|\lambda j + 1|}, \tag{3.9}$$

where j is the index for which $\sum_{i=1}^{j+1} \lambda_i < 0 \leq \sum_{i=1}^{j} \lambda_i$.

The Kaplan-Yorke dimension simplifies the estimation of the system's fractal dimension, provided that the Lyapunov spectrum of the system is accurately estimated. It is worth noting that there is no general proof of D_{KY} being exactly equal to D_H for any dynamical system. However, empirical results on different dynamical systems have shown the Kaplan-Yorke dimension D_{KY} to be either exactly or approximately equal to D_H [25].

4 Descriptions of the Systems of Interest

The dynamical systems considered in this Thesis are slightly modified versions of widely known, extensively studied models: the Kuramoto model and a network of coupled Lorenz systems.

4.1 Linked Kuramoto System

The Kuramoto model, originally formulated by Yoshiki Kuramoto [31, 32], is a dynamical system describing the dynamics of coupled oscillators. The most prominent version of the model found in the literature is defined as

$$\dot{\theta}_i = \omega_i + \frac{K}{N} \sum_{j=1}^{N} \sin(\theta_j - \theta_i), \tag{4.1}$$

where i and j index the N oscillators and θ_i is the phase of oscillator i. Each oscillator is assigned some natural frequency ω and coupled to others with strength proportional to the coupling constant K.

The coupling of the oscillators via phase differences causes the oscillators to synchronise in time. The model's motivation and its connection to nature lies in the synchrony: to name a few examples, the Kuramoto model has been used to model superconducting Josephson junctions[54], self-synchronising cardiac pacemaker cell activation[38, 42], and the behaviour of fireflies flashing in synchrony[5].

The Kuramoto model exhibits rich dynamics, including chaos and periodicity, as K and N are varied. Since its origin, the model has been studied extensively. When exploring the system's dynamics analytically, the ensemble of oscillators is typically assumed either very small ($N \in \{1, 2, 3, 4\}$) or asymptotically large ($N \to \infty$). Under these constraints the analysis in the early literature has focused on explaining the emergence of synchronisation (for example Strogatz [50]) or the onset of chaos (see e.g. Maistrenko, Popovych and Tass [37]). When N is large but finite, the system and its dynamics become much more challenging to explain analytically. For weak coupling and with N increasing past 3, the dynamics become increasingly more chaotic. This thesis focuses on this regime. Hence, the work largely resorts to analysis of the pertinent systems based on computer simulations and numerical methods.

In this thesis the Kuramoto model is defined as in [45, 48]:

$$\dot{\theta}_i = \omega_i + K \sum_{j=1}^{N} \mathbf{A}_{ij} \sin(\theta_j - \theta_i). \tag{4.2}$$

Here, $\mathbf{A}$ is the adjacency matrix representing the underlying network of oscillators. The oscillator network is a regular graph with average degree $\langle d \rangle$, where the edges are selected so that the network shows assortativity in terms of the natural frequencies of the oscillators: that is, oscillators with similar natural frequency are more likely to be connected. The detailed network generation procedure is the following [48]: After selecting a node i that still requires links another node j also requiring links and not yet connected to node i is chosen randomly. Subsequently, nodes i and j are linked with probability

$$p_{ij} = \frac{\delta^\gamma}{\delta^\gamma + |\omega_i - \omega_j|^\gamma}, \tag{4.3}$$

where δ and γ are some pre-selected parameters. The natural frequencies ω_i and ω_j are drawn from a uniform probability distribution $U(\frac{-\pi}{2}, \frac{\pi}{2})$.

The level of synchronisation of the whole system at time t can be neatly expressed by the global order parameter

$$R(t) = \frac{1}{N\langle d\rangle} \sum_{k,l=1}^{N} \mathbf{A_{kl}} e^{(i\theta_l)} \tag{4.4}$$

(Here, indexing was changed to k and l to distinguish from the imaginary unit i.) On the complex plane, the order parameter can be expressed as

$$R(t) = re^{i\psi},$$

where the radius r corresponds to the level of global synchronisation and the angle ψ to the mean phase of the oscillators. The order parameter is a key summarising macroscopic quantity of the Kuramoto model, and is an important metric in this thesis when assessing the prediction performance.

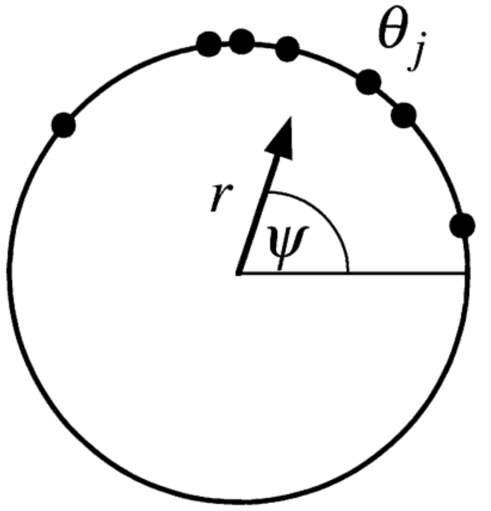


Figure 4.1: The Kuramoto model visualised as points oscillating around the complex unit circle. Individual points represent the oscillators θ_j, while the order parameter is shown as the arrow with length r. Adapted from [50], figure 1.

The Effect of the Underlying Network Structure on Dynamics

The previously defined frequency-assortative network generation itself affects the dynamics of the oscillators. This can be studied by examining networks with varying frequency assortativity, which can be computed as follows [39]:

Let e_{xy} denote the fraction of links in a network that connect a node with attribute value x to a node with attribute value y (the attribute in our case being the natural frequency of the oscillator node). To measure the assortativity of the network one can compute the standard Pearson correlation coefficient

$$r = \frac{\sum_{xy} xy(e_{xy} - a_x b_y)}{\sigma_a \sigma_b}, \tag{4.5}$$

where

$$a_x = \sum_y e_{xy}, \qquad b_y = \sum_x e_{xy},$$

and σ_a, σ_b are the standard deviations of a_x and b_y.

The higher the assortativity, the easier the oscillators tend to synchronise with one another. In extreme cases, an example of such presented in Figure 4.2, the fast synchronisation leads to simpler dynamics which are typically easier to predict.

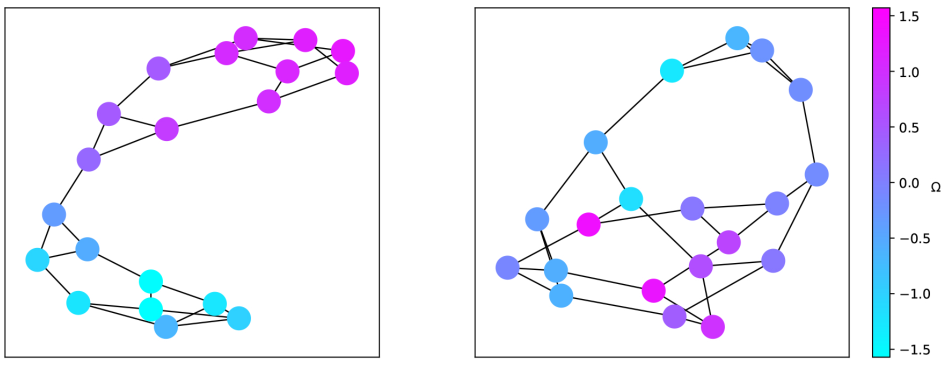


Figure 4.2: Examples of networks with high (left, $r \approx 0.92$) and low (right, $r \approx 0.23$) assortativity coefficients based on natural frequencies Ω of the oscillator nodes.

Interestingly, systems with varying assortativity of the underlying network show no clear differences in their ergodic properties, such as in the estimated Lyapunov spectrum or in the Kaplan-Yorke dimension (Figure 4.3).

As one would expect, the variation in assortativity diminishes as the number of nodes in the network grows. With the choice of parameters of Keshav et al. [48] $\delta = 0.8$ and $\gamma = 5$, the mean assortativity of randomly generated networks tends towards 0.8 with decreasing variance as the size of the network increases (Figure 4.4).

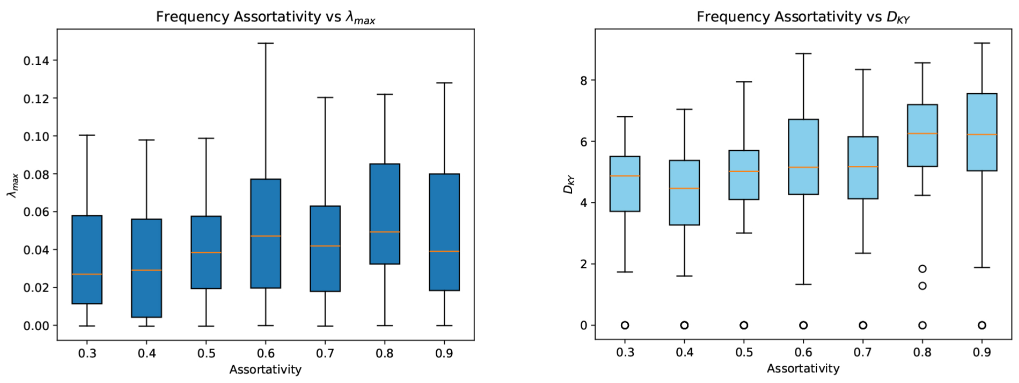


Figure 4.3: Maximum Lyapunov exponent and Kaplan-Yorke dimension as a function of assortativity of the underlying network. Each boxplot was obtained by first randomly generating a network with pre-determined assortativity value and estimating the Lyapunov spectrum of the corresponding Kuramoto system ($N = 30$) using the Benettin - Shimada & Nagashima algorithm, and repeating this process 50 times.

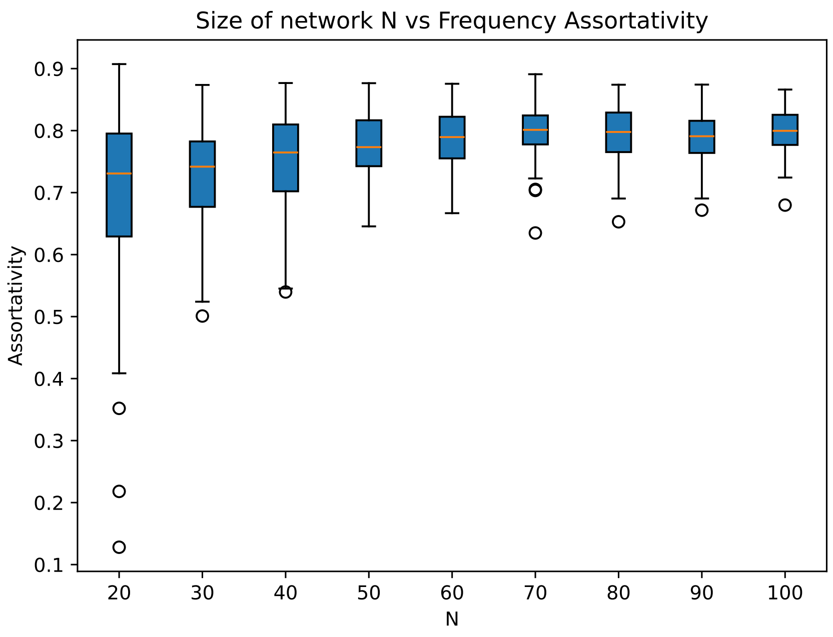


Figure 4.4: Frequency assortativity of the network as a function of number of nodes in the network, $\delta = 0.8$ and $\gamma = 5$. Each boxplot corresponds to a sample of 100 individual randomly generated networks.

4.2 Linked Lorenz System

The Lorenz system is perhaps the most widely studied and well known system in the field of nonlinear dynamics and chaos. The original formulation of the system

$$\begin{aligned}\dot{x} &= \sigma(y - x)\\ \dot{y} &= rx - y - xz\\ \dot{z} &= xy - bz\end{aligned} \tag{4.6}$$

presented by Edward Lorenz in 1963 [35] was intended as a model for atmospheric convection. The system's dynamics depend on the values of the parameters σ, r, and b: the origin is a saddle point (for all parameter values), and for $r > 1$ there are two fixed points often denoted C^+ and C^-. At $r = r_H \approx 24.74$ a subcritical Hopf bifurcation occurs: the fixed points C^+ and C^- lose their stability marking the beginning of the chaotic regime $r > r_H$.

Lorenz's choice of parameters $\sigma = 10$, $r = 28$, $b = \frac{8}{3}$ results in a dissipative system with no stable fixed points or limit cycles. Trajectories starting from any initial point will land on a so-called strange attractor and remain there indefinitely, never intersecting. All solutions of the system eventually contract in space into a set of zero volume, or more precisely, a fractal with dimension approximately equal to 2.06.

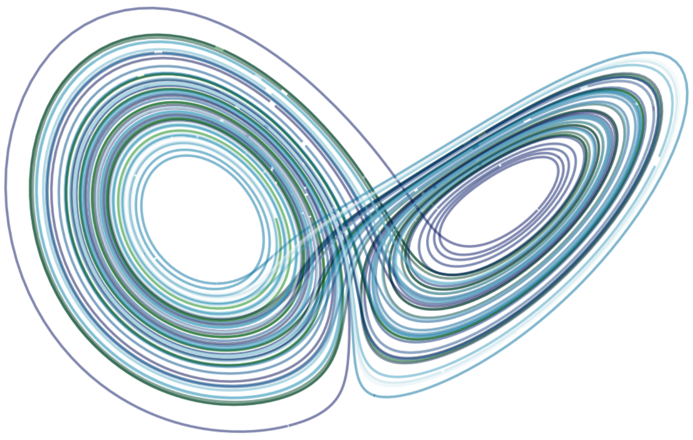


Figure 4.5: The famous butterfly-shaped strange attractor of the original Lorenz system.

In this thesis, a system of N coupled Lorenz systems is defined as

$$\begin{aligned}\dot{x}_i &= \sigma\left[y_i - x_i + c\sum_{j=1}^{N}\mathbf{A}_{ij}(y_j - y_i)\right]\\ \dot{y}_i &= rx_i - y_i - x_i z_i\\ \dot{z}_i &= x_i y_i - bz_i,\end{aligned} \tag{4.7}$$

where the individual subsystems are coupled via $\dot{x}$. The coupling strength is controlled by the parameter c, and $\mathbf{A}$ is the adjacency matrix of the system network. Similarly to the Kuramoto system, the network is a random, regular graph with average degree $\langle d \rangle$.

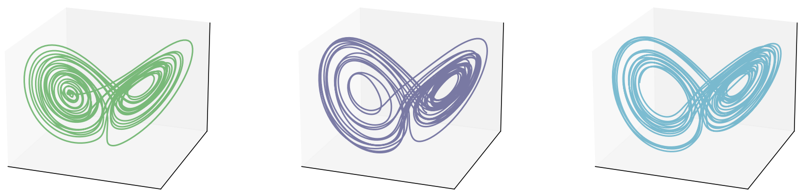


Figure 4.6: Three example subsystems in a network of coupled Lorenz systems. The differences in the subsystem dynamics are due to the underlying network structure and the coupling strength c.

The effect of coupling strength c on the system's maximum Lyapunov exponent and Kalpan-Yorke dimension are shown in Figures 4.7 and 4.8. For $c = 0$, the system is a collection of independent classic Lorenz systems, for which the maximum Lyapunov exponent is estimated to be slightly above 1 (the maximum Lyapunov exponent of a single, classic Lorenz system ≈ 0.9). λ_{max} rises approximately logarithmically as c increases. On the other hand, the entire spectrum of Lyapunov exponents and accordingly the Kaplan-Yorke dimension are only slightly increased with growing c.

In this thesis, the standard Lorenz parameter values $\sigma = 10$, $r = 28$, $b = \frac{8}{3}$ are used. The coupling constant was set to $c = 0.09$ and the number of nodes N in the underlying network of average degree $\langle d \rangle = 3$ to 10.

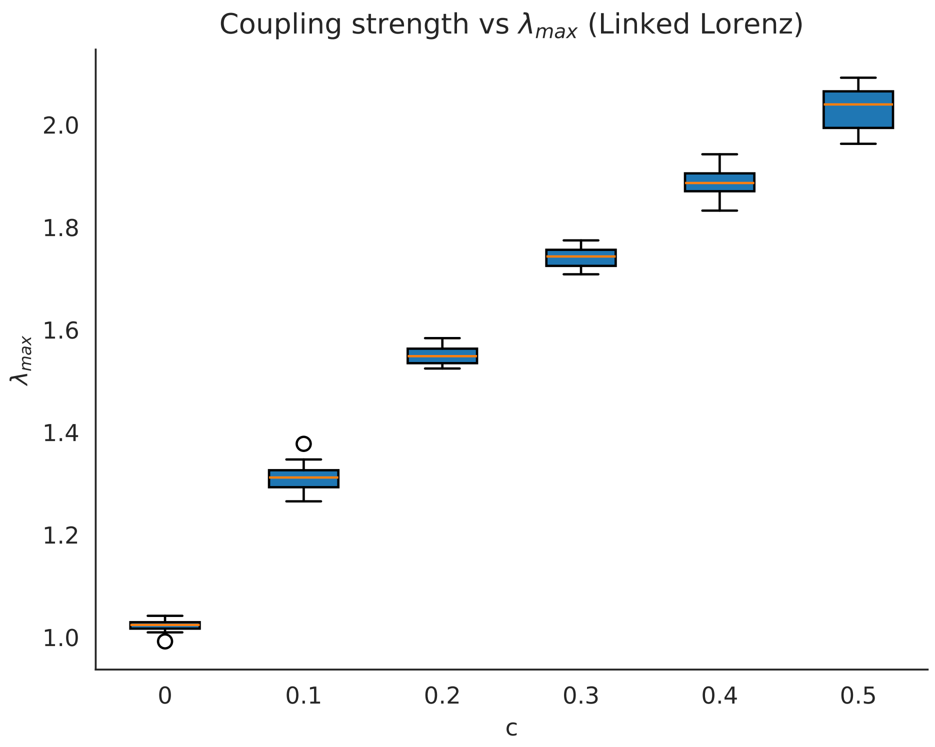


Figure 4.7: The maximum Lyapunov exponent of the linked Lorenz system as a function of the coupling strenght c.

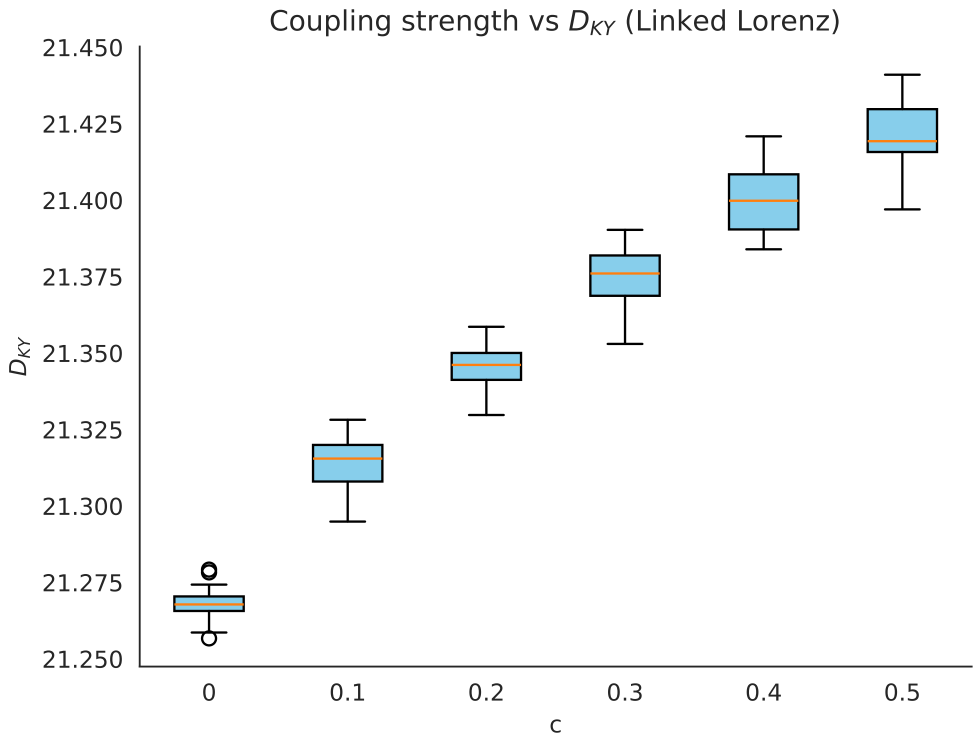


Figure 4.8: The Kalpan-Yorke dimension of the linked Lorenz system as a function of the coupling strenght c. Note the narrow scale on the y-axis.

5 The Effect of Numerical Integration Method and Accuracy on the System Dynamics

In the field of reservoir computing and nonlinear dynamics prediction, the choice of integration method and integration time step used for simulating the ground truth data is seldom discussed in detail. The methods used for generating the training and testing data vary between different authors and the dynamical systems studied (Euler's method for the linked Kuramoto system in [48], RK23 for the systems studied in [12], for instance), and are often not explicitly mentioned in the publications. This can result in difficulties when attempting to replicate the obtained results.

To investigate the significance of the ground truth data generation methods in this thesis, the properties of the simulated data were compared for different integration time steps and methods. Most importantly the maximum Lyapunov exponents and the Kaplan-Yorke dimensions are computed, since these properties are often used as indications of the existence of chaos, and to some extent as a metric of the "level of chaoticity" in the system.

Figures 5.1, 5.2, 5.3, 5.4 show the ergodic properties computed for the systems considered in this thesis for different integration methods and time steps. These results were again computed using the Benettin-Shimada & Nagashima algorithm. For each boxplot presented, the experiment was repeated 50 times with different initial conditions for the simulated dynamical system. The underlying network and the natural frequencies for the linked Kuramoto system were kept fixed throughout the trials.

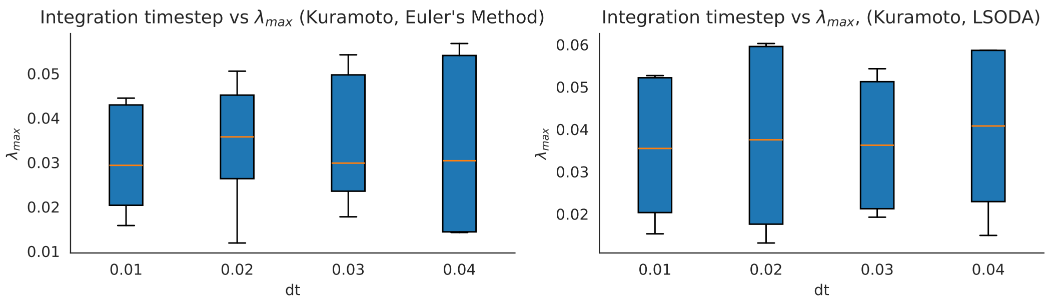


Figure 5.1: Maximum Lyapunov exponent of the linked Kuramoto system for different integration methods and time steps.

The results for the linked Kuramoto system show little variation between the two integration methods or between integration time steps of different length. However, it is worth noting the wide range of obtained values for each setting: the obtained Lyapunov spectrum can vary significantly between different initial states of integration.

The linked Lorenz system shows clear differences in the estimated Lyapunov spectra for different time steps. This is mainly true for Euler's method: the solution obtained by the higher order Runge-Kutta method remains robust to the changing integration time step.

Another curious detail is in the sizes of the attractors produced by the different integration algorithms. With the same integration time steps and initial conditions, the

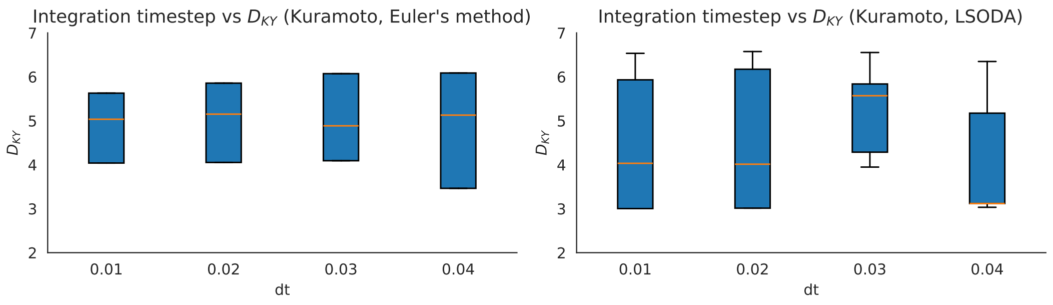


Figure 5.2: Kaplan-Yorke dimension of the linked Kuramoto system for different integration methods and time steps.

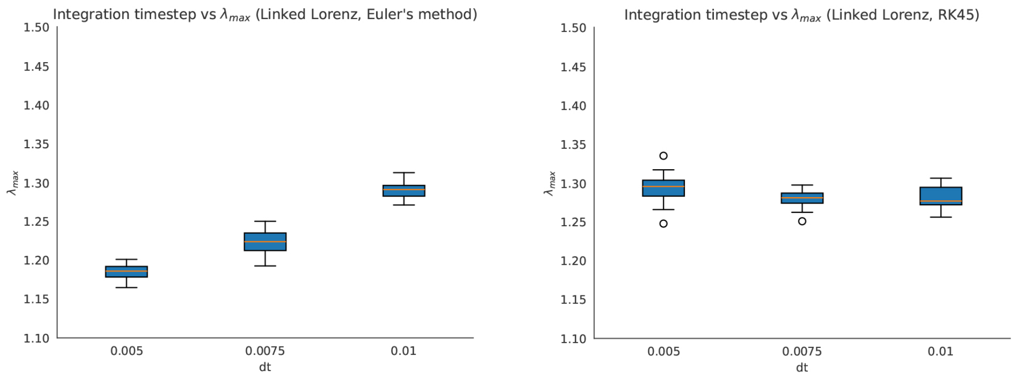


Figure 5.3: Maximum Lyapunov exponent of the linked Lorenz system for different integration methods and time steps.

solution of a single Lorenz system obtained by RK45 is confined to a clearly smaller space than that of Euler's method (Figure 5.5). In addition, Euler's method does not "respect" the unstable fixed points C^+ and C^- as RK45 does. As a conclusion, the choice of integration method and time step can have significant effects on the properties of the simulated data. These effects can translate into varying performance when predicting these systems using machine learning.

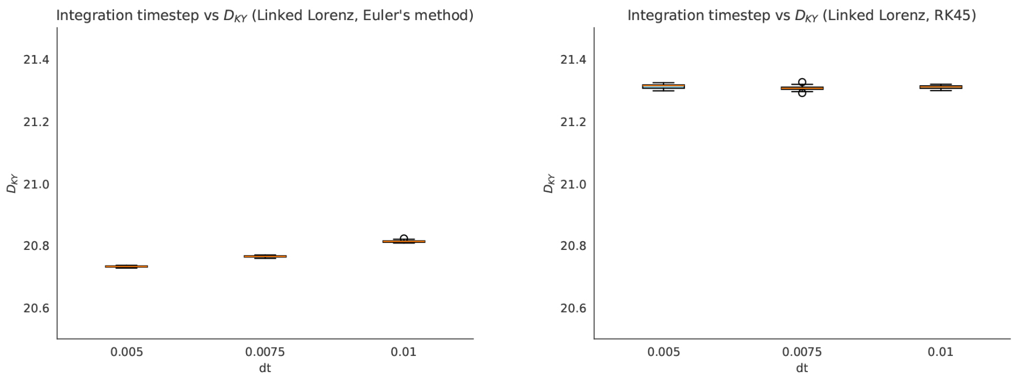


Figure 5.4: Kaplan-Yorke dimension of the linked Lorenz system for different integration methods and time steps.

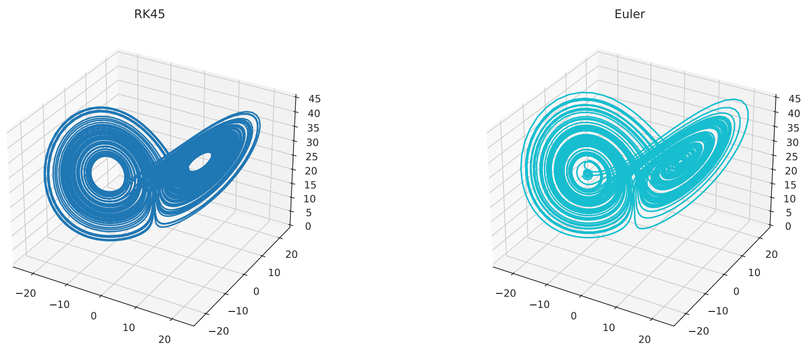


Figure 5.5: The chaotic attractor of a single Lorenz systems for different integration methods.

6 Reservoir Computing

A reservoir computer is a generalized term referring to machine learning algorithms where the input is mapped into an internal state of a high dimensional, nonlinear dynamical system called a reservoir [51]. The reservoir can in principle be any artificial or physical dynamical system, such as an artificial neural network or, literally, a bucket of water [10], as long as its different states can be invoked by some input and measured. The reservoir state is further mapped onto an output layer of the model to form a prediction for the next state of the system of interest.

In this thesis, a reservoir computing method called echo state network (ESN) was utilised. This type of reservoir computing model was first presented by Herbert Jaeger in 2001[22]. ESN is a type of recurrent neural network model where instead of attempting to learn all the link weights of the internal network, only the weights from the internal network (or reservoir) to the output layer are optimised, resulting in a linear optimisation problem.

An ESN consists of nodes in three distinct layers: the input, the reservoir, and the output. An input vector $\mathbf{u}(t)$ of length N_{in} is connected to the reservoir network by links defined by the $N_r \times N_{in}$ adjacency matrix $\mathbf{W}^{in}$, where N_r is the number of nodes in the reservoir. The internal links in the reservoir network are defined by the $N_r \times N_r$ adjacency matrix $\mathbf{W}$, and the reservoir is connected to the output layer according to another adjacency matrix $\mathbf{W}^{out}$ of size $N_{out} \times N_r$. The structure of an ESN is illustrated in Figure 6.2.

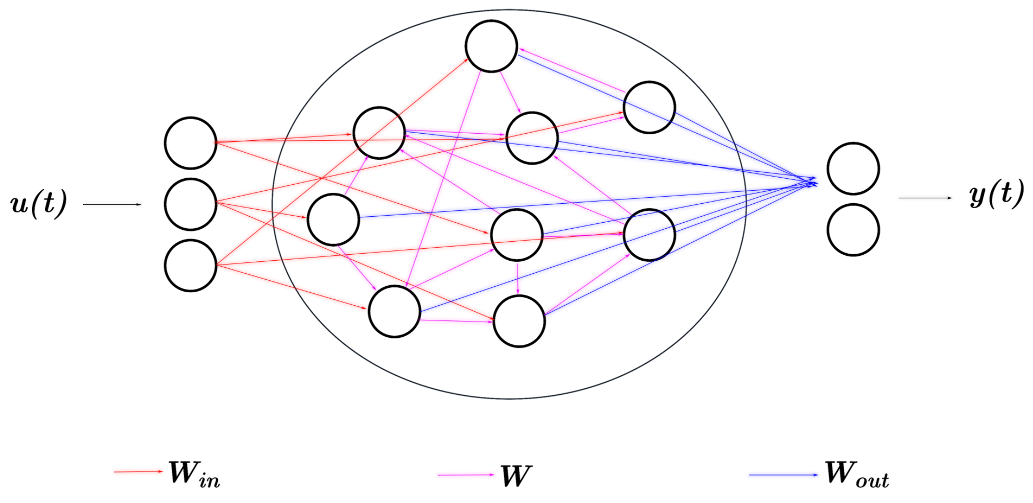


Figure 6.1: Schematic illustration of an echo state network (ESN).

At each step, the internal state of the ESN $\mathbf{r}(t)$ evolves according to

$$\mathbf{r}(t+1) = (1-\beta)\mathbf{r}(t) + \beta \tanh\left[\mathbf{W}\mathbf{r}(t) + \mathbf{W}^{in}\mathbf{u}(t)\right], \tag{6.1}$$

where $\mathbf{u}(t)$ is the input vector and β is the leak rate parameter which controls the inverse timescale of the reservoir dynamics. The reservoir output vector $\mathbf{y}(t)$ is then given by

$$\mathbf{y}(t) = \mathbf{W}^{out}\mathbf{r}(t). \tag{6.2}$$

After learning the output weights $\mathbf{W}^{out}$ through training, the ESN can be run autonomously by iteratively treating the the previous output $\mathbf{y}(t)$ as the next input vector $\mathbf{u}(t+1)$. This autonomously produced series of outputs is the model prediction of the system the ESN has been trained on.

6.1 Echo State Property

The operating principle of an ESN model requires the reservoir state $\mathbf{r}(t)$ to be uniquely defined for any left-infinite input sequence $..., \mathbf{u}(n-1), \mathbf{u}(n) = \mathbf{u}^{-\infty}$. This condition is satisfied if the ESN has the so called *echo state property*, as defined by Yildiz et al.[56]

Definition. A general ESN $F(\mathbf{r}_k, \mathbf{u}_k) : X \times U \rightarrow X$, (where $X \subset \mathbb{R}^{N_r}$, $U \subset \mathbb{R}^{N_{in}}$ are compact sets and $F(\mathbf{r}_k, \mathbf{u}_k) \in X$ and $\mathbf{u}_k \in U, \forall k \in \mathbb{Z}$) has the echo state property with respect to U if for any left infinite input sequence $\mathbf{u}^{-\infty}$ and any two state state vector sequences $\mathbf{x}^{-\infty}, \mathbf{y}^{-\infty} \in X^{-\infty}$ compatible with $\mathbf{u}^{-\infty}$, it holds that $\mathbf{x}_0 = \mathbf{y}_0$.

For an ESN with the echo state property, information about the initial state of the reservoir will asymptotically vanish, that is, the initial reservoir state is arbitrary and has no effect on the prediction in long term.

Echo state property is ensured for any input sequence and for a reservoir connection matrix $\mathbf{W}$ that is diagonally Schur stable, the criterion of which is that there exist a diagonal matrix $\mathbf{P} > 0$ such that $\mathbf{W}^T\mathbf{P}\mathbf{W} - \mathbf{P}$ is negative definite [56]. Diagonal Schur stability for a matrix $\mathbf{W}$ can be ensured via numerous different criteria, many of which depend on the *spectral radius* $\rho(\mathbf{W})$, the maximum absolute value of the eigenvalues of $\mathbf{W}$. Detailed explanations of the criteria are given by Yildiz et al. [56] and by Bhaya and Kaszkurewicz [2].

6.2 Training

Training the reservoir is to solve the system

$$\mathbf{Y}_{target} = \mathbf{W}^{out}\mathbf{R}, \tag{6.3}$$

where $\mathbf{Y}_{target}$ is the concatenated matrix for all target or "ground truth" output vectors $\mathbf{y}(t)$, and similarly $\mathbf{R}$ is the matrix formed by concatenating all reservoir states $\mathbf{r}(t)$, $t = 0, ..., T$ given by Eq. (6.1) that correspond to some input sequence $\mathbf{u}_0, ...\mathbf{u}_T$. The system is most commonly solved using ridge regression, also known as Tikhonov regularisation, first introduced by Hoerl and Kennard in 1970 as a method for solving nonorthogonal problems [21]. To find optimal reservoir output weights $\mathbf{W}^{out}$, ridge regression can be applied by solving [36]

$$\mathbf{W}^{out} = \underset{\mathbf{W}^{out}}{\arg\min} \frac{1}{N_{out}} \sum_{i=0}^{N_{out}} \left(\sum_{t=0}^{T} \left(y_i(t) - y_i^{target}(t) \right)^2 + \lambda ||\mathbf{w}_i^{out}||^2 \right), \tag{6.4}$$

where $y_i(t)$ is the i'th row of $\mathbf{y}(t) = \mathbf{W}^{out}\mathbf{r}(t)$, $y_i^{target}(t)$ is the corresponding row of $\mathbf{Y}_{target}$, $\mathbf{w}_i^{out}$ is the i'th row of $\mathbf{W}^{out}$ and $|| \cdot ||$ is the Euclidean norm. λ is the ridge parameter which penalizes higher weights in $\mathbf{W}^{out}$. The choice of λ effectively controls the relative significance of pure training error and the magnitude of output weights; this is a valuable feature for ESN models since higher output weights in $\mathbf{W}^{out}$ can make the model highly sensitive and prone to overfitting to the training data.

Before training, the reservoir internal state $\mathbf{r}(t)$ has to be initialised, which is typically done by setting all reservoir internal elements to zero. The initialization can however be chosen arbitrarily as long as the ESN satistifes the echo state property, and a sufficiently long initial period of reservoir states computed in the training process is discarded. This can be seen as a drawback of the ESN model when data is scarce.

6.3 Parallel Reservoir Computing Scheme

As the dimensionality of the underlying dynamical system increases, the cost of computation during the ESN's training and prediction rises abruptly [41]: More degrees of freedom in the underlying system requires bigger reservoir networks. In order to improve both the computational efficiency as well as the prediction accuracy of the model, a parallel reservoir computing scheme, as presented by Keshav et al. [48], is introduced.

Instead of using a single reservoir for the entire system's prediction, the parallel reservoir architecture assigns a relatively small reservoir for some subset of the underlying system. In the case of network systems, such as the Kuramoto model or the linked Lorenz system, an individual ESN is assigned to each node in the underlying system's network. For each reservoir in the parallel scheme, the input consists of that of the assigned node as well as the input from the neighbours of that node.

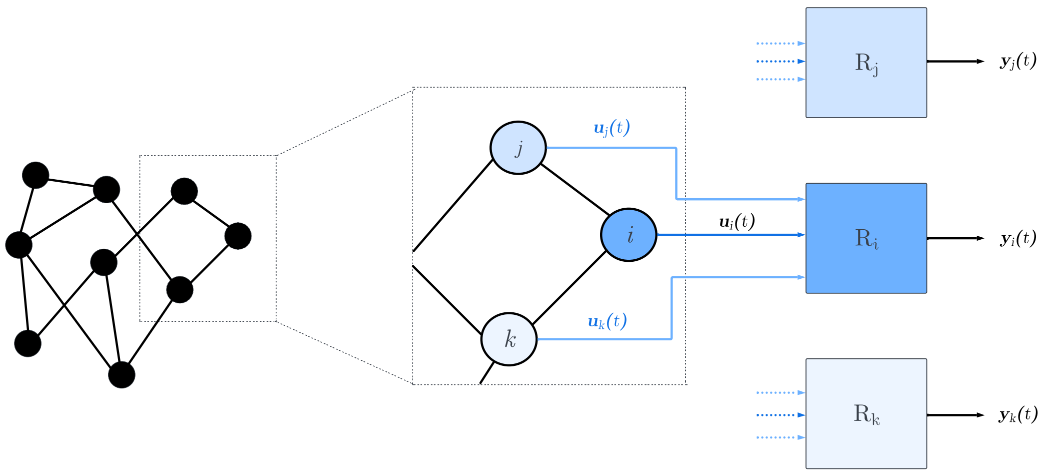


Figure 6.2: Schematic illustration of the parallel ESN architecture. The network of ESNs reflect the underlying network structure: each node of the underlying system is predicted by an ESN using the inputs from the node to the predicted as well as the input from the neighbouring nodes.

7 Model Optimisation

In machine learning, model optimisation refers to selecting a set of hyperparameters which results in optimal prediction performance. Hyperparameters are the predefining model parameters that are not altered in the training process and must therefore be set prior to fitting the model to the data. For reservoir computing and echo state networks specifically, hyperparameter tuning is crucial but alas, a computationally challenging task.

7.1 Hyperparameters of the Reservoir Computing Model

The hyperparameters chosen for the ESN models considered in this thesis are the following:

1. $\rho \in [0, 1.2]$ - the spectral radius of the reservoir adjacency matrix $\mathbf{W}$ or the maximal absolute eigenvalue of $\mathbf{W}$. For ESN models, spectral radius is typically kept below 1 to maintain the echo state property: large values of ρ increase the reservoir's likelihood of exhibiting periodicity or fixed points in its internal dynamics [36]. $\rho < 1$ is however not a necessary or sufficient condition for preserving the echo state property, and thus values slightly above 1 are allowed in the optimisation process.

2. $\sigma \in [0, 1]$ - input scaling; the absolute value of which sets the bounds for the input weights $\mathbf{W}^{in}$. More precisely, the non-zero elements of $\mathbf{W}^{in}$ are drawn from an uniform distribution $U(-\sigma, \sigma)$.

3. $\beta \in [0, 0.3]$ - leak rate parameter in Eq. (6.1) controls the inverse timescale of the ESN: a higher leaking rate means that more information of the previous state is retained when computing the next state.

4. $\lambda \in \{-9, ..., 0\}$ - base 10 logarithm of the ridge parameter in Eq. (6.4). The choice of λ effectively controls the relative significance of pure training error and the magnitude of output weights: Higher λ generally results in a model with more bias but less variance.

5. $k \in \{0, ..., 5\}$ - average degree of the reservoir network with adjacency matrix $\mathbf{W}$.

While the size of the reservoir network N_r is technically a hyperparameter similar to the aforementioned, it is often manually chosen rather than optimised along with the other parameters. This can be justified given the drastically different effect on the ESN behaviour caused by N_r - to optimise N_r is to balance the model bias and variance to the desired level [23]. Generally, the longer the training time series and the higher the dimension of it, the greater the selected N_r should be.

7.2 Bayesian Hyperparameter Optimisation

While training an ESN is a rather simple task both mathematically and computationally, optimising the hyperparameters of one is not. This is mainly due to the following: Firstly, all possible selections of hyperparameters form a sizeable space to choose from. Secondly, the parameters are not independent of each other but are rather connected in convoluted ways through the model definitions. Moreover, optimal hyperparameter values are highly task dependent for ESN's.

Learning and predicting networks of dynamical systems requires large reservoirs, and evaluating an ESN model's performance in a time series prediction task requires iterative, autonomous running of the trained reservoir. Thus, traversing the vast space of hyperparameters and evaluating model performance in each point via grid search approach is computationally extremely expensive. This imposes further limitations for the hyperparameter optimisation process, especially since the low computational cost of training is the main benefit of utilising ESN models in the first place. The problem is typically addressed by limiting the number of optimised hyperparameters [46], manual tuning of the search range of the parameters [36], and coarse grid search or stochastic search methods. For this thesis, Bayesian hyperparameter optimisation method was selected.

Expressed as a generic optimisation problem, the hyperparameter optimisation can be written as

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbf{A}} f(\mathbf{x}), \tag{7.1}$$

where $\mathbf{x}$ is some set of hyperparameters, $\mathbf{A}$ is the hyperparameter space, and f is some function yielding the model evaluation metric, namely validation error in our case. By modeling f as a stochastic process M, choosing a prior probability distribution $P(M)$, and given observations $\mathbf{y}$ from the process M, the unnormalised posterior probability density according to the Bayes' theorem is [13]

$$P(M|\mathbf{y}) \propto P(\mathbf{y}|M)P(M). \tag{7.2}$$

For machine learning hyperparameter optimisation, the prior of choice for $f(\mathbf{x})$ is a Gaussian process:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}_i, \mathbf{x}_j) \tag{7.3}$$

with some mean function m and covariance function k. The mean and covariance functions can be utilised to incorporate prior information into the Gaussian process: typical choices for them are [55]

$$m(\mathbf{x}) = 0,$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2}\right).$$

The posterior is inferred as follows: given a set of t observations $D = \{\mathbf{x}_n, f(\mathbf{x}_n)\}_{n=1}^{t}$, where $f(\mathbf{x}_n) \sim N(0, \mathbf{K})$,

$$\mathbf{K} = \begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \dots & k(x_t, x_t) \end{bmatrix},$$

the joint probability density of the previous observations and a new observation $\{\mathbf{x}_{t+1}, f(\mathbf{x}_{t+1})\}$ follows a multivariate Gaussian [13, 55]:

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f_{t+1} \end{bmatrix} \sim N\left(0, \begin{bmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & k(x_{t+1}, x_{t+1}) \end{bmatrix}\right).$$

Having updated the posterior probability density for f, the next point in the hyperparameter space can be determined by optimising some cheap-to-evaluate acquisition function u:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in \mathbf{A}} u(\mathbf{x}|D). \tag{7.4}$$

The acquisition function used for the hyperparameters optimisation in this thesis is the so-called negative expected improvement function

$$u(\mathbf{x}) - EI(\mathbf{x}) = \mathbb{E}\left[f(\mathbf{x}) - f(\mathbf{x}_t^*)\right]. \tag{7.5}$$

With the posterior density for f determined, evaluating the expectation in Eq. (7.5) is straightforward.

The optimisation process starts by an initial random guess at the hyperparametes. For each iteration of the Bayesian optimisation, a reservoir computing model is trained and evaluated by computing the valid prediction time, defined as the time t when the prediction error $E(t)$ first exceeds some predetermined threshold $f \in [0, 1]$. The error is defined as

$$E(t) = \frac{||\mathbf{u}(t) - \mathbf{y}(t)||}{\sqrt{\langle ||\mathbf{u}(t)||^2 \rangle}}, \tag{7.6}$$

where $||\cdot||$ is the Frobenius norm and $\langle\cdot\rangle$ is the mean operator. This process is continued until the optimisation converges, or in a vastly more likely scenario, a set number of iterations have been performed. Afterwards, the best performing set of hyperparameters is selected.

In this thesis, the Skopt Python package [18] implementation of Bayesian hyperparameter optimisation was utilized.

7.3 Reservoir Connection Network Generation

While the ESN connection matrices $\mathbf{W}$ and $\mathbf{W}^{in}$ are random graphs, their properties, such as average degree or size, are not trivial choices when constructing the model. It is largely agreed that the reservoir network, both between the input layer and the reservoir internal connections should be sparse; this was noted in the original publication by Jaeger [22] and later supported by numerous papers, such as [14, 16]. Sparse reservoir networks have been heuristically shown to yield better prediction results, and most popular programming languages provide efficient sparse matrix operations for faster computation.

The strategy for generating the links from the input layer to the reservoir, defined by the matrix $\mathbf{W}^{in}$, can be selected in various ways. Previously used methods found in the literature often involve simply connecting each input node to each reservoir node with certain probability. In this thesis $\mathbf{W}^{in}$ was constructed according to the strategy defined by Keshav et al. in [48]: connect each of the N_r nodes in the reservoir to exactly one input node at random. The method creates sparse connections between the input and the reservoir, which was found to increase the prediction performance compared to more dense $\mathbf{W}^{in}$ matrices [48].

The internal reservoir network defined by matrix $\mathbf{W}'$ is initialized as follows: for each node i, k other nodes from the network are drawn without replacement, and links from these nodes are connected to node i. A weight for each of these links is then drawn from $\mathcal{N}(0, 1)$. $\mathbf{W}'$ thus defines a directed network with average in-degree k. Finally, the provisional matrix $\mathbf{W}'$ is scaled to obtain the final reservoir matrix according to

$$\mathbf{W} = \frac{\rho}{\rho_{\mathbf{W}'}}\mathbf{W}', \tag{7.7}$$

where $\rho_{\mathbf{W}'}$ is the spectral radius of the provisional matrix and ρ is the desired spectral radius specified in the model hyperparameters.

In addition to regular graphs, other types of reservoir topologies have been studied in previous publications [14, 27, 28]. These include, for example, tree-like, cyclic, and line-like graphs. For relatively small reservoir networks ($N_r < 1000$) predicting the typical chaotic systems, such as the Lorenz system, the different network topologies have shown little to no variation in terms of prediction performance. Another possible way to define the model is to have no internal links at all, resulting in a reservoir of unconnected nodes. It has been suggested that when predicting dynamical systems with low enough fractal dimension, the unconnected reservoir outperforms the traditional ESN models with recurrent network structures [27, 28]. However, the traditional ESN approach was chosen in this thesis due to the complexity of the coupled dynamical systems analysed.

8 Parallel Reservoir Scheme Implementation Details

A key benefit of the parallel reservoir scheme is the option to utilise parallel computing when training the individual ESN's. Since training of a single ESN requires iterative computation of the internal state vectors, parallelisation of such computation without modifying the training process principles is not feasible. In the parallel reservoir scheme, the individual ESN's can be trained independently of each other, enabling straightforward utilisation of parallel computing. Parallel computing is especially beneficial when predicting high-dimensional systems, as these typically require the usage of larger reservoirs. In general, given that resources for parallel computing are available, training multiple small, parallel ESN's is considerably more efficient than training a single, large ESN.

The benefits of parallelisation are prominently realised in the hyperparameter optimisation process, during which the model is trained up to several hundred times. However, in order to evaluate the model and for running the parallel reservoirs autonomously in general, each ESN in the network requires the previous outputs from the neighbouring ESN's as input to compute the next output. This dependency complicates parallelisation of the model computation: during autonomous prediction, the parallel reservoirs have to be both in synchrony for each iteration, and communicate the outputs in some way. This essentially rules out the usage of separate processing and memory approaches, such as message passing interface - while technically feasible to implement, the need for constant, blocking communication between processes nullifies the benefits gained by sharing the workload. Furthermore, for large networks of reservoirs it may not be possible to assign each reservoir with its own process, further complicating the implementation. Thus, a shared memory computing environment is the most practical option for the parallel reservoir scheme.

After training the reservoirs separately and in parallel, all the reservoir matrices are concatenated in preparation for the autonomous run computation. The concatenated matrix dimensions are $\mathbf{W}_{N_{res}\times N_r\times N_r}$, $\mathbf{W}^{in}_{N_{res}\times N_r\times N_{in}}$, and $\mathbf{W}^{out}_{N_{res}\times N_{out}\times N_r}$. Now for the concatenated matrices and the whole parallel reservoir scheme, the reservoir's next state computation Eq. (6.1) can be written as

$$\mathbf{R}(t+1) = (1-\beta)\mathbf{R}(t) + \beta\tanh\left[\sum_{i=0}^{N_{res}}\sum_{j=0}^{N_r}\sum_{k=0}^{N_r}\mathbf{W}_{ijk}\mathbf{R}_{ik}(t) + \sum_{i=0}^{N_{res}}\sum_{j=0}^{N_r}\sum_{k=0}^{N_{in}}\mathbf{W}^{in}_{ijk}\mathbf{U}_{ki}(t)\right], \quad (8.1)$$

where $\mathbf{R}$ is the internal state matrix and $\mathbf{U}$ is the input matrix for the whole parallel scheme. Correspondingly, the output computation Eq. (6.2) becomes

$$\mathbf{Y}(t) = \sum_{i=0}^{N_{res}}\sum_{j=0}^{N_{out}}\sum_{k=0}^{N_r}\mathbf{W}^{out}_{ijk}\mathbf{R}_{ik}(t), \quad (8.2)$$

where $\mathbf{Y}(t)_{N_{res}\times N_{out}}$ is the matrix containing all outputs from each of the reservoirs.

Note that in the parallel scheme, $\mathbf{U}$ and $\mathbf{Y}$ do not share the same dimensions; thus, for each iteration in the autonomous run of the model, the output has to be arranged so that each of row of the next input contains both the output of the corresponding reservoir node and the outputs of the neighbours of the same reservoir.

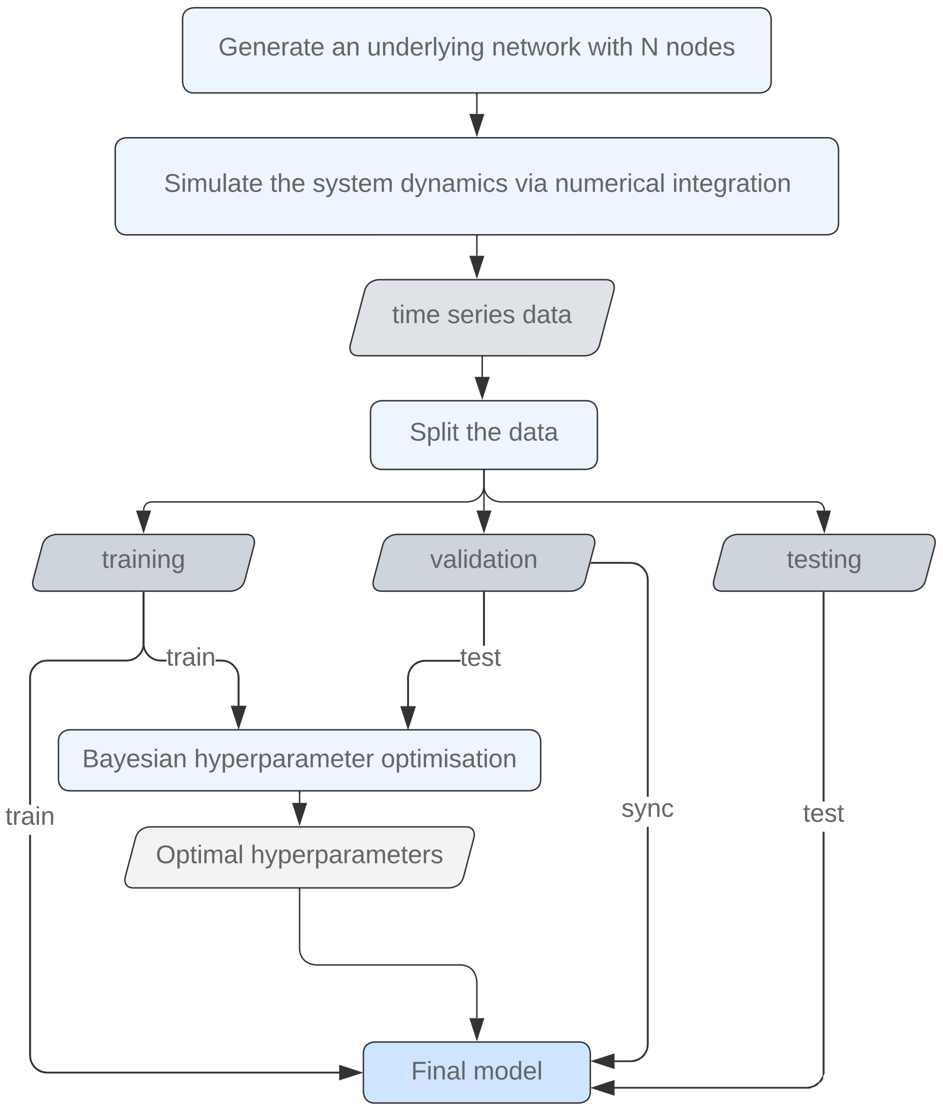
8.1 Hyperparameter Optimisation in the Parallel Scheme

In principle, each reservoir in the parallel model could be assigned its own set of hyperparameters independent of the other reservoirs in the network. This could potentially be highly beneficial for the model performance, as suggested in [48]. However, finding an optimal, possibly unique set of hyperparameters for each reservoir is not feasible in reasonable computation time. This is due to the reservoir predictions being dependent on each other through the neighbouring connections: tweaking the parameters of one reservoir affects the behaviour of the entire reservoir network (assuming the network is connected). This immediately leads to the Bayesian optimisation approach to not work at all, and the vast hyperparameter space would require massive number of iterations for methods such as random search or grid search.

In this thesis, an attempt to overcome this problem was to find optimal hyperparameters for each reservoir by employing a "pseudo-parallel" optimisation scheme by running the Bayesian optimisation separately for each of the reservoirs. When evaluating the model performance for each iteration of the optimisation, the outputs from the neighbouring reservoirs were replaced by the ground truth data, essentially assuming perfect predictions from the neighbouring nodes and thus enabling independent optimisation of hyperparameters for each reservoir. However, using a common set of hyperparameters for all of the reservoirs and treating the entire parallel scheme as the model in the Bayesian optimisation consistently outperforms the "pseudo-parallel" approach. In conclusion of this, a common set of hyperparameters for each reservoir in the parallel scheme was optimised and used in this thesis.

9 Methodology and Results

The general flow of analysis applied to different dynamical systems in this thesis is presented in Figure 9.1.



Generate an underlying network with N nodes
Simulate the system dynamics via numerical integration
time series data
Split the data
training
validation
testing
train
test
Bayesian hyperparameter optimisation
train
sync
test
Optimal hyperparameters
Final model

Figure 9.1: Analysis flow diagram.

The stochastic elements in both data simulation (due to random initial conditions) and the Bayesian hyperparameters optimisation cause variance in the prediction performance. In an attempt to find a model that performs well when predicting the system dynamics in general, the hyperparameter optimisation process is done as follows:

1. Having set up an underlying system network, do for a sufficient number of times:
   - simulate ground truth data with a random initial state
   - run Bayesian hyperparameter optimisation, record the best performing parameter set
2. Construct the final set of hyperparameters from the median values of all of the optimisation runs.

Training, validation and testing datasets are sections of the same time series and the model is trained with the same data during both hyperparameter optimisation and final evaluation. Hence, after training the model in the final evaluation phase, the reservoir is kept up-to-date by iteratively computing the internal states through the validation dataset. This, too, is a parallelisable process in the parallel reservoir scheme.

The main performance metric used for model evaluation in all stages of the analysis is the valid prediction time defined in Section 7.2 with maximum threshold $f = 0.4$ for the error in Eq. (7.6). The valid prediction times between predictions are compared in terms of Lyapunov time, that is, a characteristic timescale defined by the largest Lyapunov exponent of the system. For numerically simulated time series of the dynamical systems, the unit Lyapunov time is defined as

$$\lambda_{max} t = \frac{1}{\lambda_{max} dt}, \tag{9.1}$$

where dt is the integration time step used when simulating the system dynamics.

9.1 Linked Kuramoto System

The linked Kuramoto system analysed consists of 30 Kuramoto oscillators. The natural frequencies are drawn independently according to $\omega_i \sim U(\frac{-\pi}{2}, \frac{\pi}{2})$, and the underlying network is generated according to the process described in section 4 with parameters $\delta = 0.8, \gamma = 5$. The frequency assortativity value for the analysed network is approximately 0.8, and the system has estimated maximum Lyapunov exponent $\lambda_{max} \approx 0.074$, and Kaplan-Yorke dimension $D_{KY} \approx 6.15$. The ground truth data for the results in this section was simulated using the Scipy [53] implementation of the LSODA numerical integration algorithm [43, 19].

Single Reservoir Versus Parallel Reservoir Scheme

The prediction performances of a single, large reservoir ($N_r = 3000$) and a network of parallel reservoirs ($N_r = 300$) are compared. The performance of both the single reservoir model and the parallel reservoir scheme are measured by

1) computing valid prediction times for individual subsystems in the underlying network,

2) computing the valid prediction time for the global order parameter Eq. (4.4).

Typical example predictions for 1) and 2) are illustrated in Figures 9.3, 9.3 and 9.4. The parallel reservoir scheme consistently outperforms a single ESN in terms of valid prediction time, although the difference is hardly as clear as obtained by Keshav et al. [48].

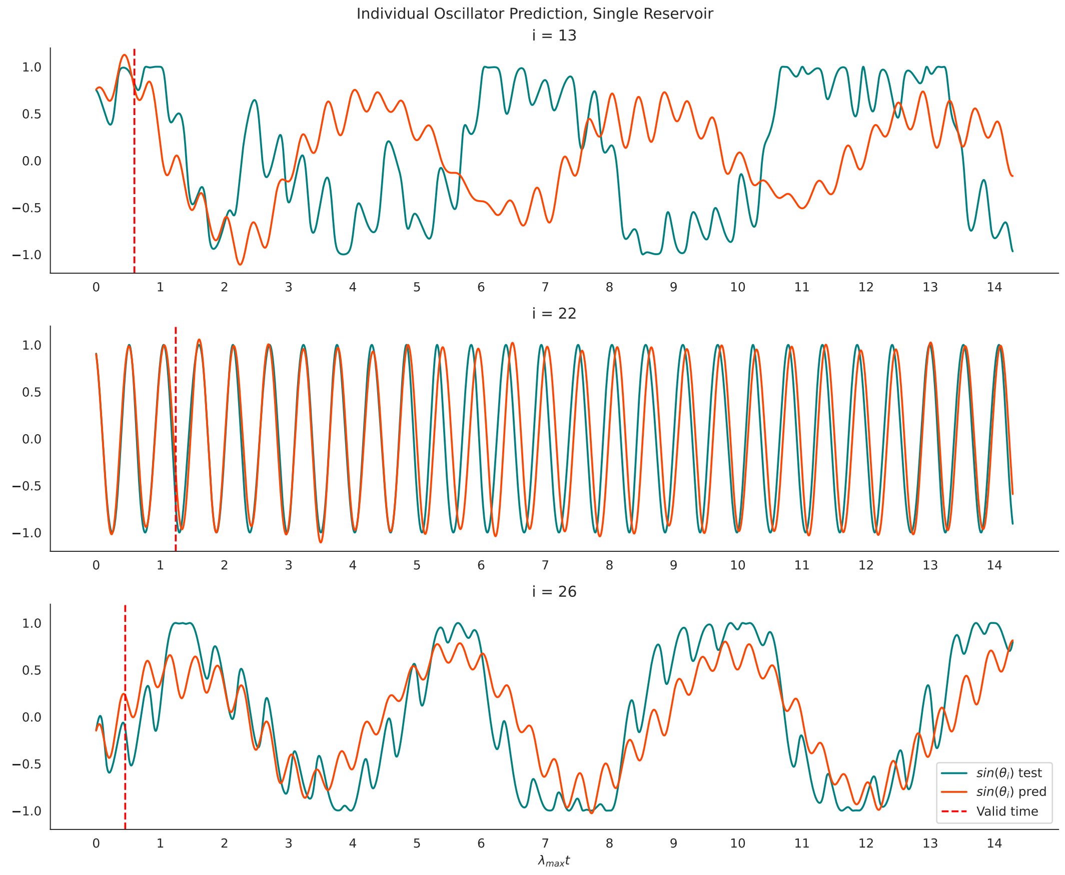


Figure 9.2: Example predictions for individual oscillators of the coupled Kuramoto system (Single reservoir).

To account for the intrinsic randomness in the prediction performance of both models, sampling distributions over multiple runs were computed. Figure 9.5 shows distributions of valid times for individual oscillators and for the global order parameter for the linked Kuramoto system with $N = 30$. The results consist of recorded valid times over 50 prediction runs. The initial conditions for the ground truth data simulation were varied between the runs, since some random initial conditions may lead to easier prediction if, for example, the system starts near synchrony or fixed state.

As clearly seen in Figure 9.5, the parallel reservoir scheme significantly outperforms a single ESN on average, in terms of both global order parameter and individual oscillator predictions. The related sample means and standard deviations are presented in Table 9.1.

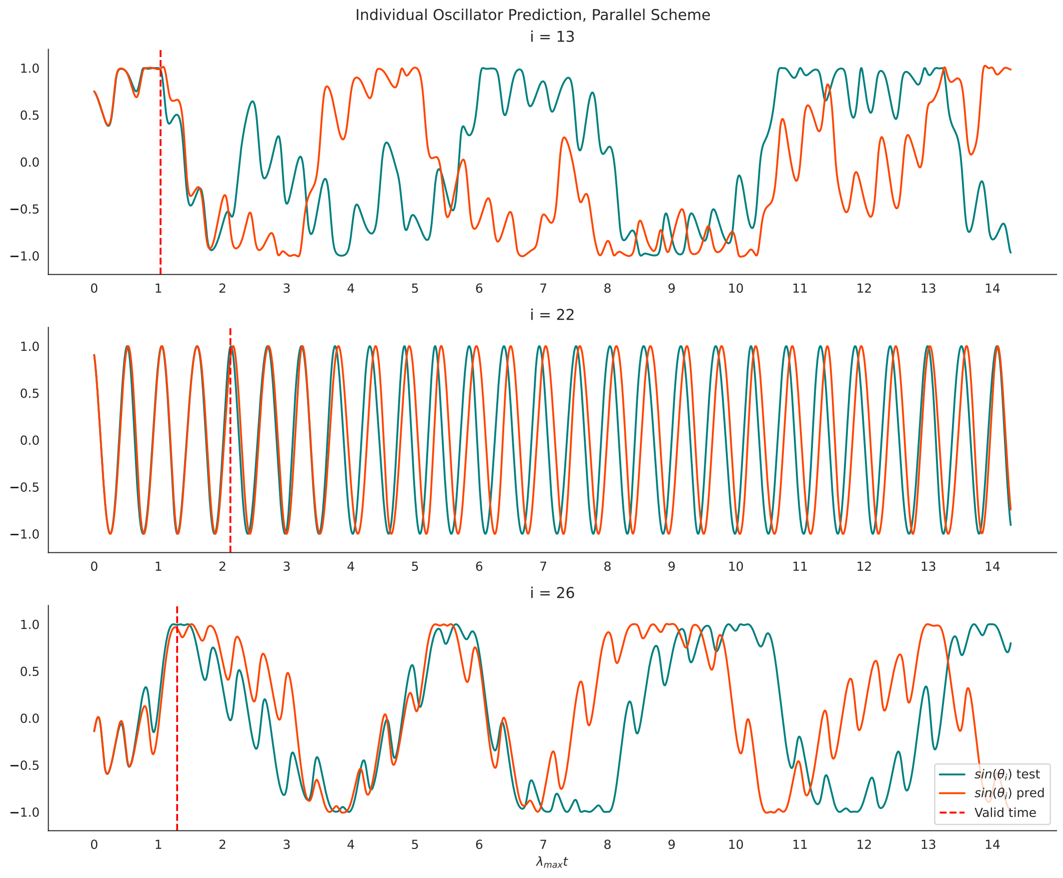


Figure 9.3: Example predictions for individual oscillators of the coupled Kuramoto system (Parallel reservoirs).

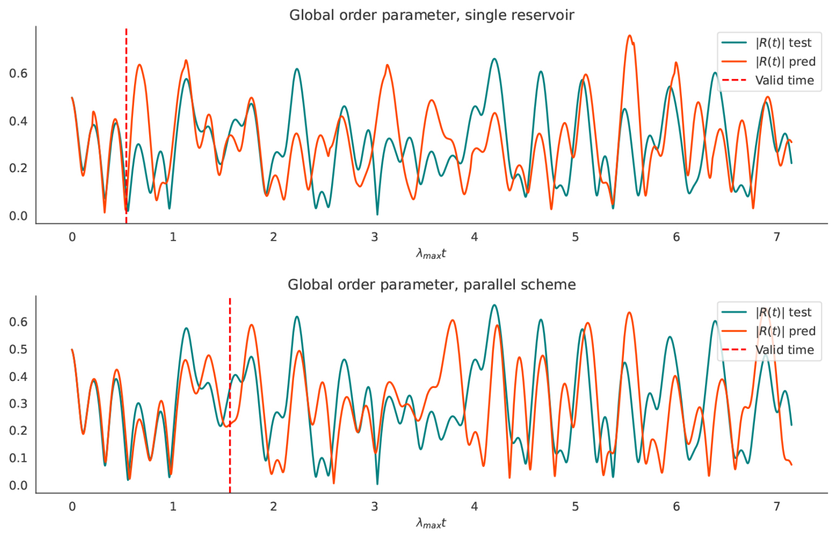


Figure 9.4: Example predictions for the global order parameter of the coupled Kuramoto system.

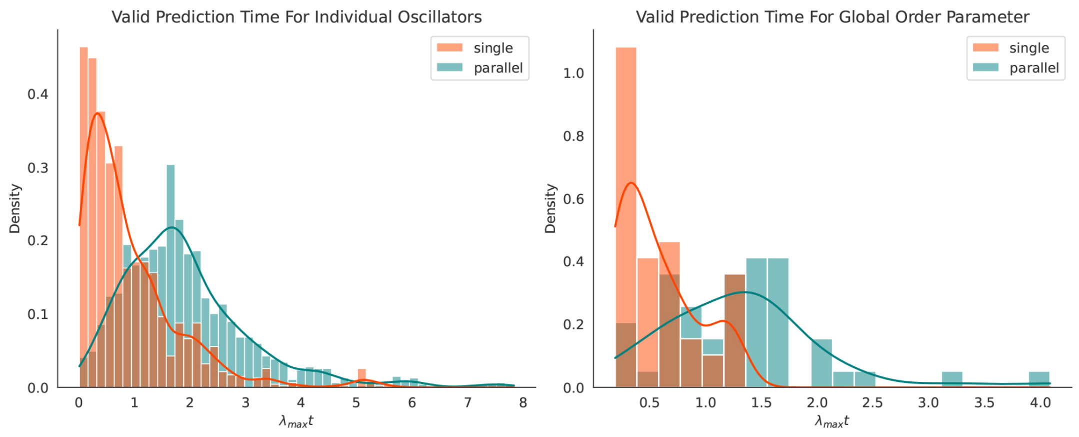


Figure 9.5: Valid time distributions for individual oscillators and for the global order parameter for the linked Kuramoto system with $N = 30$.

model/metric	Mean	SD
Single Reservoir/subsystem valid time	0.91	0.89
Parallel Reservoirs/subsystem valid time	1.92	1.19
Singe Reservoir/order parameter valid time	0.56	0.33
Parallel Reservoirs/order parameter valid time	1.31	0.71

Table 9.1: Means and standard deviations of valid prediction times for both models.

Effect of Time Step Length and Integration Method

To analyse the effect of varying settings in simulating the ground truth data, the parallel reservoir model was evaluated 50 times for different combinations of integration time step length and integration algorithm. For each time step-method pair, the hyperparameters were selected in a similar fashion with the previous section. For each time step length dt, the reservoirs were trained and tested on fixed system time, that is, the total number of datapoints in the initially simulated ground truth time series was $\frac{1600}{dt}$ for all dt tested. Valid time distributions for subsystem and the global order parameter predictions are presented in Figures 9.6 and 9.7, respectively.

Smaller integration time step tends to produce slightly better performing models in terms of valid prediction time. This is expected since the training data essentially contains more information per time unit. One would expect the more sophisticated (and typically more precise) integration method to be less sensitive to increasing the time step length; this however cannot be observed from the results with the linked Kuramoto system. This may be due to the relatively narrow range of tested time step lengths: the LSODA method may be more robust with longer time steps, but on the other hand Euler's method quickly becomes numerically unstable with larger time steps, in which case, the two time series for different integrators being too dissimilar, the comparison of reservoir prediction times becomes meaningless.

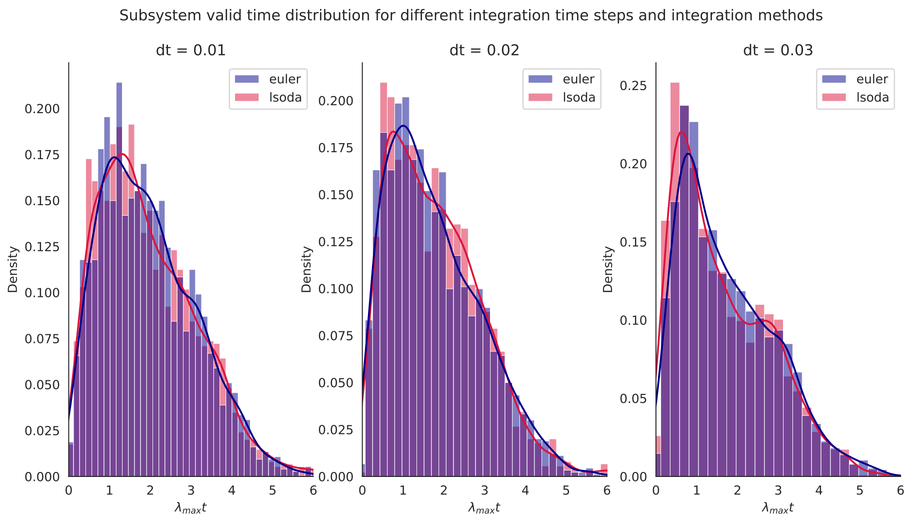


Figure 9.6: Valid time distributions for individual oscillators with different integration methods and time steps for the linked Kuramoto system with $N = 30$.

Integration method	time step	Mean	SD
Euler	0.01	1.96	1.17
LSODA	0.01	1.93	1.19
Euler	0.02	1.78	1.17
LSODA	0.02	1.81	1.16
Euler	0.03	1.76	1.20
LSODA	0.03	1.63	0.14

Table 9.2: Means and standard deviations of subsystem valid prediction times for different integration methods and time steps.

Integration method	time step	Mean	SD
Euler	0.01	1.27	0.91
LSODA	0.01	1.23	0.84
Euler	0.02	1.32	0.81
LSODA	0.02	1.27	0.93
Euler	0.03	1.13	0.90
LSODA	0.03	1.15	0.95

Table 9.3: Means and standard deviations of the global order parameter valid prediction times for different integration methods and time steps.

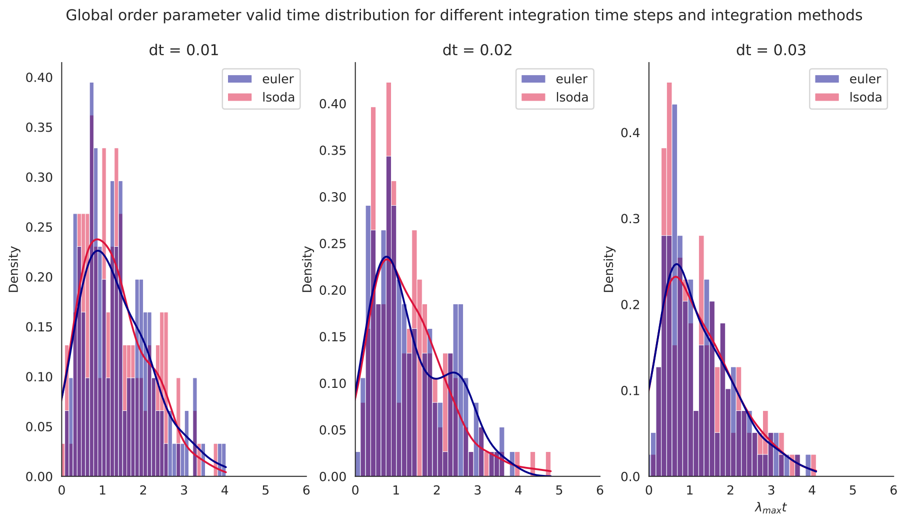


Figure 9.7: Valid time distributions for the global order parameter with different integration methods and time steps for the linked Kuramoto system with $N = 30$.

9.2 Linked Lorenz System

A linked Lorenz system presented in Section 4.2 with 10 subsystems was analysed. Similar to the linked Kuramoto system, the underlying network for the linked Lorenz system is a regular undirected graph where the degree of each node $\langle k \rangle = 3$. For convenience, the network was generated using the same process as with the frequency assortative network of the Kuramoto system, albeit there are no natural frequencies or other attributes associated with the Lorenz systems that would affect the system dynamics.

For the linked Lorenz system, similar analysis flow as in previous sections was followed, see Figure 9.1. The main performance metric for the predictions was the valid prediction time computed for the 10 subsystems in the system network.

Single Reservoir Versus Parallel Reservoir Scheme

The prediction performances of a single reservoir of size $N_r = 3000$ and a network of parallel reservoirs each with $N_r = 600$ were compared. Examples of prediction performances are shown in Figures 9.8 - 9.11. In a majority of cases, the parallel reservoir scheme outperforms the single reservoir in terms of valid prediction time. Aside from the strict valid prediction times, both methods tend to replicate the qualitative behaviour, or the "climate" of the system quite well for around 20 to 40 multiples of the system Lyapunov time, although the parallel scheme typically stays more true to the original, as seen in Figure 9.11.

model/metric	Mean	SD
Single Reservoir/subsystem valid time	2.24	1.32
Parallel Reservoirs/subsystem valid time	2.58	1.38

Table 9.4: Means and standard deviations of valid prediction times for both models (Linked Lorenz system).

Effect of Time Step Length and Integration Method

As with the linked Kuramoto system, similar experiment with varying ground truth data integration methods was conducted. The largest tested time step was selected to be at the edge of numerical stability for the Euler's method. Instead of the LSODA method, a fifth order Runge-Kutta with fourth order error estimation (RK45) was used as the higher order integrator for comparison.

For the linked Lorenz system, smaller time steps again yield significantly better valid prediction times. Unlike in the case of the linked Kuramoto system, noticeable differences between the two integration methods were observed, RK45 being the method with stronger prediction performance. Reflecting upon Section 5 some differences were expected; however, since RK45 is the method producing data with larger maximum Lyapunov exponents and Kaplan-Yorke dimension, the result may seem unintuitive.

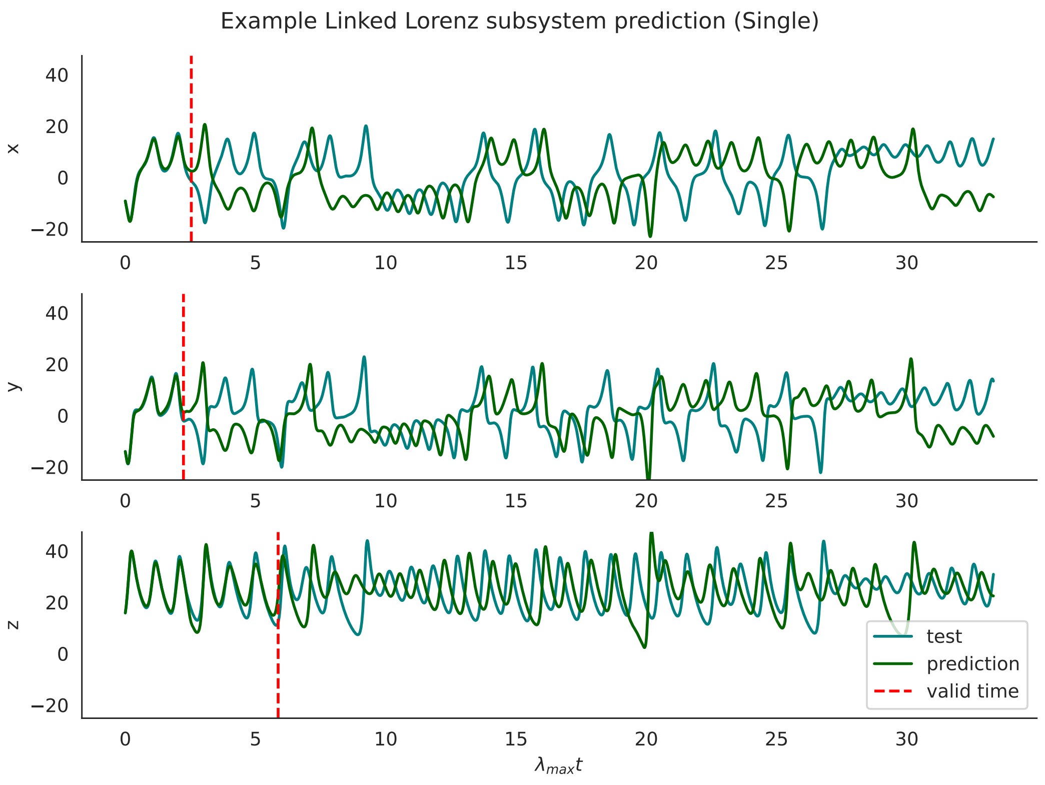


Figure 9.8: Example predictions for individual coordinates of a single subsystem in the linked Lorenz system with $N = 10$ (Single Reservoir).

Integration method	time step	Mean	SD
Euler	0.005	2.51	1.32
RK45	0.005	3.16	1.34
Euler	0.0075	1.39	0.81
RK45	0.0075	2.57	1.38
Euler	0.01	1.43	0.73
RK45	0.01	2.12	1.24

Table 9.5: Means and standard deviations of linked Lorenz subsystem valid prediction times for different integration methods and time steps.

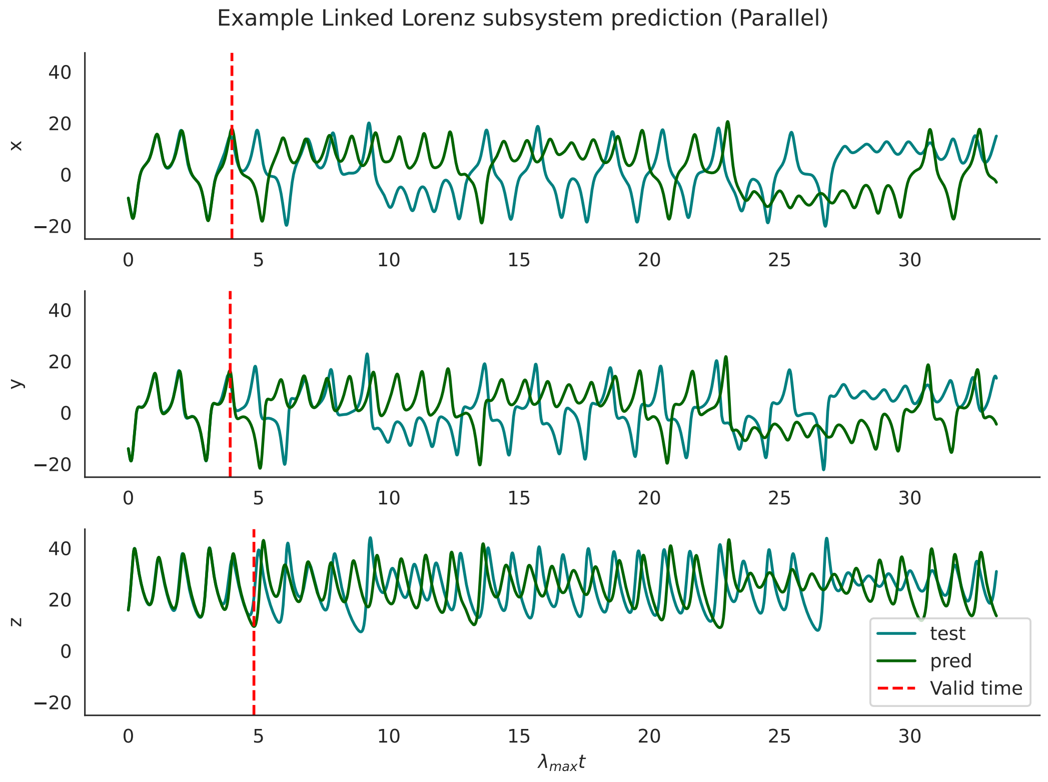


Figure 9.9: Example predictions for individual coordinates of a single subsystem in the linked Lorenz system with $N = 10$ (Parallel reservoirs).

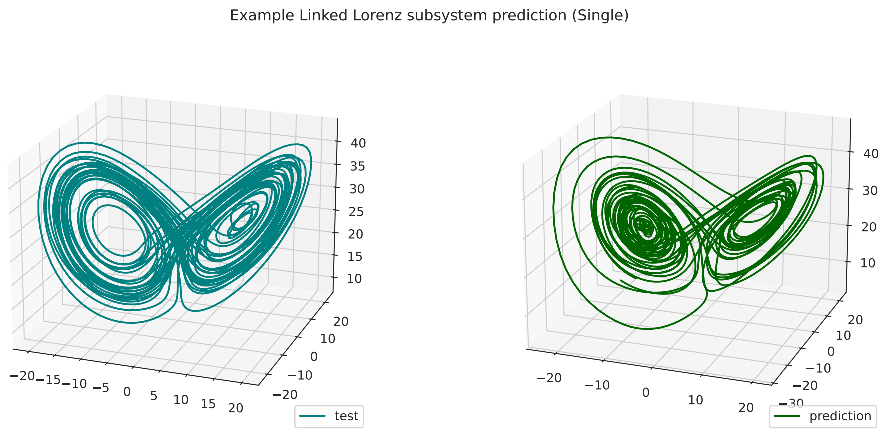


Figure 9.10: Examples of predicted attractors of a single subsystem in the linked Lorenz system with $N = 10$ (Single reservoir).

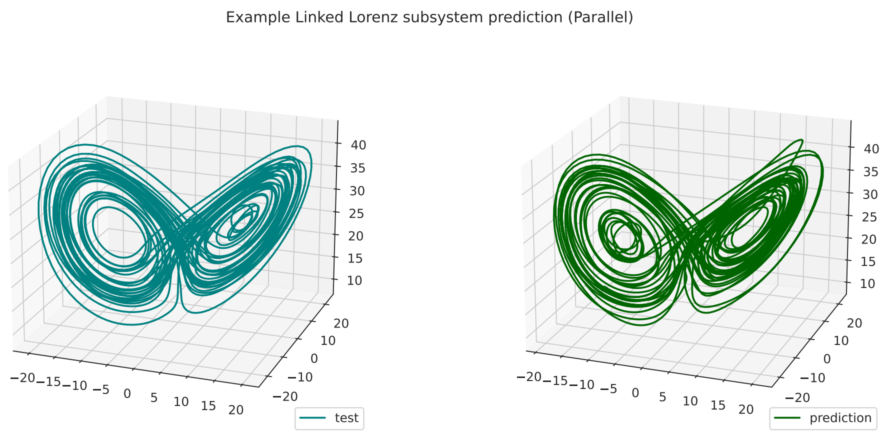


Figure 9.11: Examples of predicted attractors of a single subsystem in the linked Lorenz system with $N = 10$ (Parallel reservoirs).

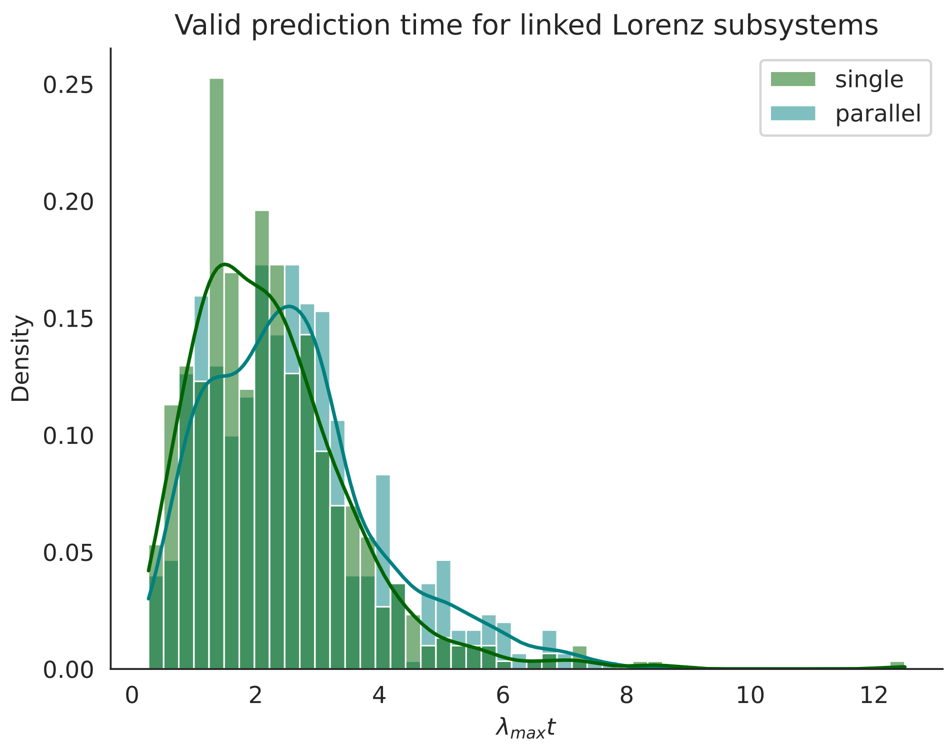


Figure 9.12: Valid time distributions for subsystem predictions for the linked Lorenz system with $N = 10$.

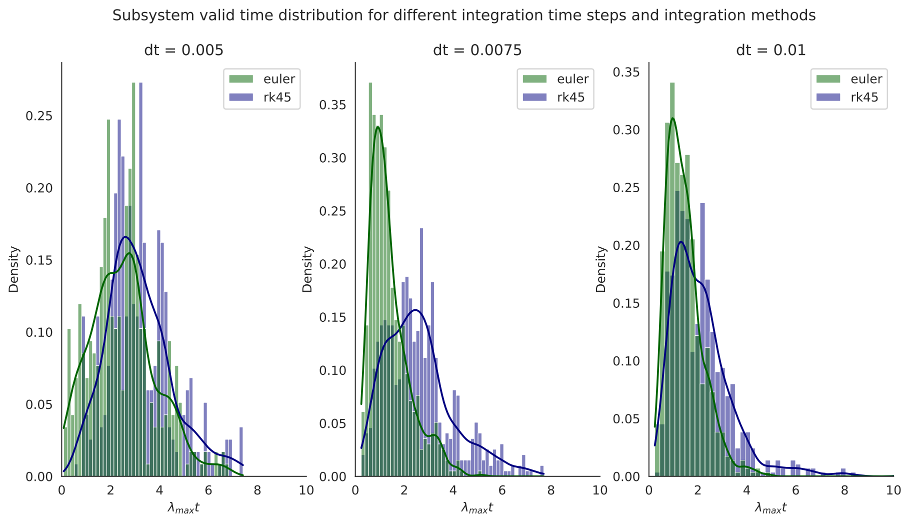


Figure 9.13: Valid time distributions for subsystem predictions for different integration methods and time steps (linked Lorenz system, $N = 10$).

10 Discussion

Comparison between a single ESN and the parallel reservoir scheme indicate a clear difference - the network of small, parallel ESN's outperform a single, large ESN by a significant margin in terms of both subsystem valid prediction time and qualitative replication of the ground truth system's climate. The results agree partly with the analysis of Keshav et al. [48]: though the parallel reservoir scheme shows stronger prediction performance, the valid prediction times obtained in the analysis of this thesis did not consistently match those presented in [48].

It can be argued that the comparison between the models is not fair: for example in the case of the linked Kuramoto system, the total possible number of states a network of 30 reservoirs each with $N_r = 300$ can represent significantly exceeds that of a single reservoir with $N_r = 3000$. While in principle and in terms of prediction performance there is no downside for increasing the size of the reservoir [23, 36], the benefits of increasing the reservoir size diminish after a certain, case specific point. This was observed during the analysis of this thesis. Moreover, the average single reservoir prediction performance obtained in this analysis exceeds the performance reported in [48], where a significantly larger single reservoir ($N_r = 10000$) was used. This is especially true for the linked Lorenz system, where the results of [48] indicate a misspecified or poorly optimised single reservoir model.

An important advantage of the parallel reservoir scheme is the possibility of parallel training and synchronisation of the reservoirs. For problems with a single training sequence, the training process of an ESN is inherently non-parallelisable. Futhermore, since the reservoir state computation is dominated by matrix multiplication operations with computational complexity (depending on the algorithm and matrix dimensions) between $O(n^2)$ and $O(n^3)$, limiting the matrix sizes is necessary for computational feasibility. With modern computers parallel processing is almost always an available option, and being able to leverage this with the parallel reservoir scheme is a major benefit especially with high-dimensional data.

As discussed in Section 7, the only actually viable way to optimise the parallel reservoirs is to use a common set of hyperparameters for each ESN in the network. This is due to the reservoirs being dependent on each others' outputs, which in turn precludes Bayesian hyperparameter search. If each reservoir were to have its own set of hyperparameters, random search and grid search would be the only options; however, for a hyperparameter space this large these methods are not computationally feasible.

In order to utilise tailored hyperparameters for each reservoir, the hyperparameter space would have to be drastically limited. One possible path towards that and improving the model optimisation process could be to study the model sensitivity to each parameter in detail, find robust parameters and optimise only the volatile parameters. Distributions of optimal hyperparameters obtained by a large number of Bayesian hyperparameter optimisation for the linked Lorenz system are presented in Figure 10.1. An obvious example of a parameter which likely should not be optimised in further runs is the input scaling σ, as overwhelming majority of the obtained values are in the low end of the scale. Unfortunately, the same cannot be said of the other parameters, as the Bayesian optimisation settles on a wide range of different parameter values between different runs.

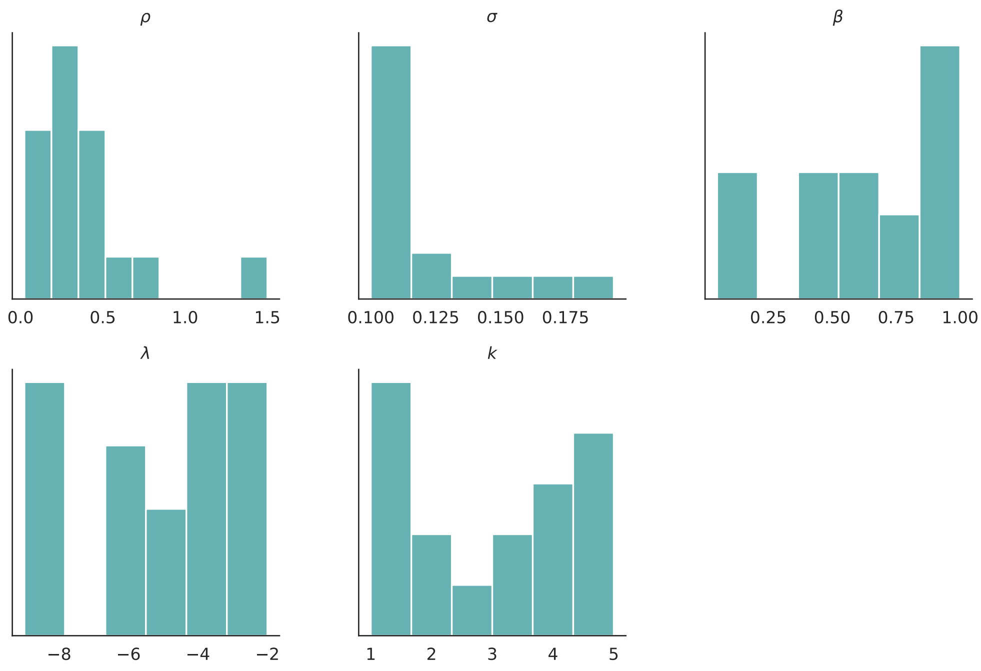


Figure 10.1: Distributions of optimal hyperparameters obtained during Bayesian optimisation over a ca. 50 runs for the linked Lorenz system, parallel reservoir scheme.

Nevertheless, hyperparameter optimisation remains highly important for both single and parallel ESN models. Alas, this adds another layer of inherent randomness to the model with already high statistical variation between different initialisations of the reservoirs. One way to rid the model of these stochastic elements would require explicit definition of the reservoirs' input and internal connections. This approach quickly leads back to learning the optimal connection weights instead of random initialisation, and our model design has come into a full cycle back to deep learning.

Another solution to the varying model performance is to reduce the number of random connections both in the input layer and in the recurrent internal layer, or remove them altogether. This exact approach has been studied in previous publications, some example models of which include Next-generation reservoir computing [12] and reservoir of unconnected nodes [28]. Both of these models lack any recurrent neural network structure. Contrary to previous expectations, these models can show equal or superior performance compared to traditional ESN's when predicting sufficiently low-dimensional dynamical systems. Conversely, the opposite is implied to be true: the higher the system complexity, the more important recurrent connections and careful model tuning seem to become [27, 28, 30].

The previous remark encapsulates one of the base principles of machine learning and statistical modeling in general: more complex phenomena require more complex models to accurately describe them. The linked dynamical systems studied in this thesis are indeed quite complex compared to the systems typically analysed in the field and which reservoir computing models are known to perform well with. Therefore it is no surprise that by incorporating information of the underlying network structure directly into the model and thus increasing the model complexity leads to improved prediction performance.

The secondary topic of interest in this work was the effect of numerical integration methods on the properties of the simulated time series and on the machine learning task. As described in the results of Section 5, changes in either the integration algorithm or the integration time step cause significant changes in the properties of the data. Similar analysis on lower dimensional chaotic systems was done in [30]. The differences between integration methods and time steps seem to vary from system to system in a non-systematic fashion, and general guidelines for what numerical integration method should be used for which dynamical systems cannot be drawn.

In the field of reservoir computing and chaotic dynamics prediction, different authors often test their models on the same, widely known dynamical systems, such as the Lorenz system. While the dynamical equations for these systems are largely standard, the method for numerically solving them is not. Generating simulated data of the dynamical systems often seems to be treated as a trivial preliminary step before the actual analysis. This may lead to careless documenting of the data generation process, which in turn complicates possible replication of the results and comparison of different models. Thus, taking the effects of numerical integration presented in [30] and in this thesis into consideration should be an integral part of chaotic dynamics machine learning analysis flow.

References

[1] G. Benettin et al. “Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems - A method for computing all of them. I - Theory. II - Numerical application”. In: *Meccanica* 15 (Mar. 1980), pp. 21–30. DOI: 10.1007/BF02128236.

[2] Amit Bhaya and Eugenius Kaszkurewicz. “On discrete-time diagonal and D-stability”. In: *Linear Algebra and its Applications* 187 (1993), pp. 87–104. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(93)90129-C.

[3] Erik Bollt. “On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.1 (Jan. 2021), p. 013108. ISSN: 1054-1500. DOI: 10.1063/5.0024890.

[4] Michael Breakspear, Stewart Heitmann, and Andreas Daffertshofer. “Generative Models of Cortical Oscillations: Neurobiological Implications of the Kuramoto Model”. In: *Frontiers in Human Neuroscience* 4 (2010). ISSN: 1662-5161. DOI: 10.3389/fnhum.2010.00190.

[5] John Buck. “Synchronous Rhythmic Flashing of Fireflies. II.” In: *The Quarterly Review of Biology* 63.3 (1988), pp. 265–289. ISSN: 00335770, 15397718. DOI: 10.1086/415929.

[6] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[7] J.R. Dormand and P.J Prince. “A family of embedded Runge-Kutta formulae”. In: *Journal of Computational and Applied Mathematics* 6.1 (1980), pp. 19–26. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0771-050X(80)90013-3.

[8] J. -P. Eckmann and D. Ruelle. “Ergodic theory of chaos and strange attractors”. In: *Rev. Mod. Phys.* 57 (3 July 1985), pp. 617–656. DOI: 10.1103/RevModPhys.57.617.

[9] K. Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. 2nd ed. John Wiley Sons, Ltd, 2003. ISBN: 9780470013854. DOI: https://doi.org/10.1002/0470013850.

[10] Chrisantha Fernando and Sampsa Sojakka. “Pattern Recognition in a Bucket”. In: *Advances in Artificial Life*. Ed. by Wolfgang Banzhaf et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 588–597. ISBN: 978-3-540-39432-7.

[11] Paul Frederickson et al. “The liapunov dimension of strange attractors”. In: *Journal of Differential Equations* 49.2 (1983), pp. 185–207. ISSN: 0022-0396. DOI: https://doi.org/10.1016/0022-0396(83)90011-6.

[12] Daniel J. Gauthier et al. “Next generation reservoir computing”. In: *Nature Communications* 12.1 (Sept. 2021). ISSN: 2041-1723. DOI: 10.1038/s41467-021-25801-2.

[13] Andrew Gelman et al. *Bayesian Data Analysis*. 3rd ed. Chapman and Hall/CRC, 2013. ISBN: 9780429113079. DOI: https://doi.org/10.1201/b16018.

[14] Aaron Griffith, Andrew Pomerance, and Daniel J. Gauthier. “Forecasting chaotic systems with very low connectivity reservoir computers”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.12 (Dec. 2019), p. 123108. ISSN: 1054-1500. DOI: 10.1063/1.5120710.

[15] R. Haberman. *Elementary Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems*. Prentice-Hall, 1987. ISBN: 9780132528757.

[16] Alexander Haluszczynski and Christoph Räth. “Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.10 (Oct. 2019), p. 103143. ISSN: 1054-1500. DOI: 10.1063/1.5118725.

[17] F. Hausdorff. “Dimension und äußeres Maß”. In: *Mathematische Annalen* 79 (1919), pp. 157–179.

[18] Tim Head et al. *scikit-optimize/scikit-optimize*. Version v0.9.0. Oct. 2021. DOI: 10.5281/zenodo.5565057.

[19] A. C. Hindmarsh. “ODEPACK: A systematized collection of ODE solvers”. In: *IMACS Transactions on Scientific Computation* 1 (1983), pp. 55–64.

[20] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[21] Arthur E. Hoerl and Robert W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems”. In: *Technometrics* 12.1 (1970), pp. 55–67. ISSN: 00401706.

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

[23] Herbert Jaeger et al. “Optimization and applications of echo state networks with leaky- integrator neurons”. In: *Neural Networks* 20.3 (2007). Echo State Networks and Liquid State Machines, pp. 335–352. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2007.04.016.

[24] Yohan J. John et al. “It’s about time: Linking dynamical systems with human neuroimaging to understand the brain”. In: *Network Neuroscience* 6.4 (Oct. 2022), pp. 960–979. ISSN: 2472-1751. DOI: 10.1162/netn_a_00230.

[25] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*. 2nd ed. Cambridge University Press, 2003. ISBN: 9780511755798. DOI: https://doi.org/10.1017/CBO9780511755798.

[26] James L. Kaplan and James A. Yorke. “Chaotic behavior of multidimensional difference equations”. In: *Functional Differential Equations and Approximation of Fixed Points*. Ed. by Heinz-Otto Peitgen and Hans-Otto Walther. Berlin, Heidelberg: Springer Berlin Heidelberg, 1979, pp. 204–227. ISBN: 978-3-540-35129-0.

[27] Pauliina Kärkkäinen. “Optimal reservoir computers for nonlinear systems of varying complexity”. Available at https://urn.fi/URN:NBN:fi:aalto-202312067102. Licentiate thesis. Espoo, Finland: Aalto University, Aug. 2023.

[28] Pauliina Kärkkäinen and Riku Linna. *Dimensional criterion for forecasting nonlinear systems by reservoir computing*. 2022. arXiv: 2202.05159 [cs.LG].

[29] H. von Koch. “On a continuous curve without a tangent, obtained by an elementary geometrical construction”. In: *Ark. Mat. Astron. Fys.* 1 (1904), pp. 681–702. ISSN: 0365-4133.

[30] Anna Kosklin. *Reservoir Computing in the Prediction of Nonlinear Systems*. Master’s thesis. Espoo, Finland, Apr. 2024.

[31] Y. Kuramoto. *Chemical Oscillations, Waves, and Turbulence*. Springer Series in Synergetics. Springer Berlin Heidelberg, 1984. ISBN: 978-3-642-69691-6. DOI: https://doi.org/10.1007/978-3-642-69689-3.

[32] Yoshiki Kuramoto. “Self-entrainment of a population of coupled non-linear oscillators”. In: *International Symposium on Mathematical Problems in Theoretical Physics*. Ed. by Huzihiro Araki. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 420–422. ISBN: 978-3-540-37509-8.

[33] Nikolay Kuznetsov and Volker Reitmann. *Attractor Dimension Estimates for Dynamical Systems: Theory and Computation: Dedicated to Gennady Leonov*. Jan. 2021, p. 545. ISBN: 978-3-030-50986-6. DOI: 10.1007/978-3-030-50987-3.

[34] J.D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley and Sons, 1991.

[35] Edward N. Lorenz. “Deterministic Nonperiodic Flow”. In: *Journal of Atmospheric Sciences* 20.2 (1963), pp. 130–141. DOI: https://doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

[36] Mantas Lukoševičius. “A Practical Guide to Applying Echo State Networks”. In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 659–686. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_36.

[37] Yuri Maistrenko, Oleksandr Popovych, and Peter Tass. “Chaotic Attractor in the Kuramoto Model.” In: *I. J. Bifurcation and Chaos* 15 (Nov. 2005), pp. 3457–3466. DOI: 10.1142/S0218127405014155.

[38] Renato E. Mirollo and Steven H. Strogatz. “Synchronization of Pulse-Coupled Biological Oscillators”. In: *SIAM Journal on Applied Mathematics* 50.6 (1990), pp. 1645–1662. DOI: 10.1137/0150098.

[39] M. E. J. Newman. “Mixing patterns in networks”. In: *Phys. Rev. E* 67 (2 Feb. 2003), p. 026126. DOI: 10.1103/PhysRevE.67.026126.

[40] V.I. Oseledets. “A multiplicative ergodic theorem. Characteristic Ljapunov, exponents of dynamical systems”. In: *Tr. Mosk. Mat. Obs.* 19 (1968), pp. 179–210.

[41] Jaideep Pathak et al. “Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach”. In: *Phys. Rev. Lett.* 120 (2 Jan. 2018), p. 024102. DOI: 10.1103/PhysRevLett.120.024102.

[42] Charles S. Peskin. “Mathematical Aspects of Heart Physiology”. In: *Courant Institute of Mathematical Science Publication* (1975), pp. 268–278.

[43] Linda Petzold. “Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations”. In: *SIAM Journal on Scientific and Statistical Computing* 4 (Mar. 1983). DOI: 10.1137/0904010.

[44] William H. Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. Cambridge University Press, 2007. ISBN: 0521880688.

[45] Juan G. Restrepo and Edward Ott. “Mean-field theory of assortative networks of phase oscillators”. In: *Europhysics Letters* 107.6 (Sept. 2014), p. 60006. DOI: 10.1209/0295-5075/107/60006.

[46] Ali Rodan and Peter Tino. “Minimum complexity echo state network. IEEE Trans Neural Netw”. In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 22 (Nov. 2010), pp. 131–44. DOI: 10.1109/TNN.2010.2089641.

[47] I. Shimada and T. Nagashima. “A Numerical Approach to Ergodic Problem of Dissipative Dynamical Systems”. In: *Progress of Theoretical Physics* 61.6 (June 1979), pp. 1605–1616. ISSN: 0033-068X. DOI: 10.1143/PTP.61.1605.

[48] Keshav Srinivasan et al. “Parallel Machine Learning for Forecasting the Dynamics of Complex Networks”. In: *Phys. Rev. Lett.* 128 (16 Apr. 2022), p. 164101. DOI: 10.1103/PhysRevLett.128.164101.

[49] S.H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. CRC Press, 2018. ISBN: 9780429961113.

[50] Steven H. Strogatz. "From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators". In: *Physica D: Nonlinear Phenomena* 143.1 (2000), pp. 1–20. ISSN: 0167-2789. DOI: https://doi.org/10.1016/S0167-2789(00)00094-4.

[51] Gouhei Tanaka et al. "Recent advances in physical reservoir computing: A review". In: *Neural Networks* 115 (2019), pp. 100–123. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2019.03.005.

[52] "The influence of inter-regional delays in generating large-scale brain networks of phase synchronization". In: *NeuroImage* 279 (2023), p. 120318. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2023.120318.

[53] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[54] Kurt Wiesenfeld, Pere Colet, and Steven H. Strogatz. "Synchronization Transitions in a Disordered Josephson Series Array". In: *Phys. Rev. Lett.* 76 (3 Jan. 1996), pp. 404–407. DOI: 10.1103/PhysRevLett.76.404.

[55] Jia Wu et al. "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization b". In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40. ISSN: 1674-862X. DOI: https://doi.org/10.11989/JEST.1674-862X.80904120.

[56] Izzet B. Yildiz, Herbert Jaeger, and Stefan J. Kiebel. "Re-visiting the echo state property". In: *Neural Networks* 35 (2012), pp. 1–9. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2012.07.005.