Estimating Parameters of Governing Equations of Non-Linear Systems from Data Using Synchronisation and Machine Learning

Schätzung der Parameter von zugrunde liegenden Gleichungen nicht linearer Systeme aus Daten mittels Synchronisation und maschinellen Lernens

A Master's thesis at the
Faculty of Physics of the
Ludwig Maximilian University Munich

Eine Masterarbeit an der
Fakultät für Physik der
Ludwig-Maximilians-Universität München

submitted by vorgelegt von

Davide Prosperino

on the am

15th June 2022 15. Juni 2022

supervised by betreut von

Haochun Ma
PD Dr. Christoph Räth

Dedicato a
Papà, Mamma, Jassem
e anche Johnny

Grazie per tutto quello che mi avete reso possibile!

Abstract Deriving governing equations from a measured time series is an ongoing topic of research across different disciplines in science. One method studied can derive the form of governing equations, however it cannot infer the coefficients in front of each term. This is where our work comes in: given the form of governing equations and a measured time series, we propose an algorithm for finding the correct coefficients of the governing equations describing the observed data best. We achieve this by treating the data as primary system and coupling a secondary system to it. Then by inducing synchronisation, we can change the parameters of the secondary system in the direction minimising a loss function. After the loss has reached its minimum, the found parameters are a good estimation of the real parameters producing the data. We applied our algorithm successfully on a number on synthetic systems and even found a method for reconstructing signals masked by chaotic data.

Zusammenfassung Die Herleitung von Gleichungen aus einer gemessenen Zeitreihe ist ein aktives Forschungsthema in verschiedenen wissenschaftlichen Disziplinen. Eine der untersuchten Methoden kann die Form der zugrunde liegenden Gleichungen herleiten, jedoch nicht die Koeffizienten vor jedem Term. Hier setzt unsere Arbeit an: Wenn die Form der zugrunde liegenden Gleichungen und eine gemessenen Zeitreihe gegeben sind, stellen wir einen Algorithmus vor, um die richtigen Koeffizienten der zugrunde liegenden Gleichungen zu finden. Wir erreichen dies, indem wir die Daten als primäres System betrachten und ein sekundäres System an dieses koppeln. Durch Herbeiführen von Synchronisation, können wir dann die Parameter des sekundären Systems in die Richtung ändern, die eine Verlustfunktion minimiert. Nachdem die Verlustfunktion ihr Minimum erreicht hat, sind die gefundenen Parameter eine gute Schätzung der echten Parameter, die die Daten erzeugt haben. Wir haben unseren Algorithmus erfolgreich auf eine Reihe synthetischer Systeme angewandt und sogar eine Methode gefunden um Signale zu rekonstruieren, die durch chaotische Daten maskiert worden sind.

Contents

1 Introduction 1

2 Overview of Complex Systems 3
- 2.1 Lyapunov Exponent . . . 4
  - 2.1.1 Calculation of Lyapunov Exponents for Different Systems . . . 5
- 2.2 The Lorenz System . . . 7

3 Synchronisation of Complex Systems 9
- 3.1 Synchronisation Through a Driving Variable . . . 9
  - 3.1.1 Synchronisation Using Active-Passive-Decomposition . . . 11
- 3.2 Synchronisation of General Systems by External Coupling . . . 14
  - 3.2.1 Bidirectional Synchronisation . . . 14
  - 3.2.2 Unidirectional Synchronisation . . . 18

4 Optimisation Algorithm 23
- 4.1 Integration of Ordinary Differential Equations . . . 23
  - 4.1.1 Runge–Kutta–Fehlberg Integration . . . 23
  - 4.1.2 Integration of ODEs Coupled to Data . . . 25
- 4.2 Loss Function and Gradient . . . 27
  - 4.2.1 The Loss Function . . . 27
  - 4.2.2 Calculation of the Gradient . . . 28
- 4.3 Update Step . . . 31
- 4.4 Summary . . . 33

5 Results and Applications to Synthetic Systems 35
- 5.1 Effect of Hyperparameters . . . 39
- 5.2 Robustness Against Noise & Incorrect Knowledge . . . 40
  - 5.2.1 Robustness Against Noisy Data . . . 40
  - 5.2.2 Robustness Against Imprecise Knowledge of the Governing Equations . . . 45
- 5.3 Comparison to Similar Work . . . 49

6 Applications to Real Systems 55
- 6.1 Lotka–Volterra System . . . 55
- 6.2 Financial Markets . . . 58
- 6.3 Reconstruction of Signals Masked by Complex Systems . . . 60

7 Summary & Outlook 63

A Complex Systems Studied in this Work 65

Bibliography 69

Chapter 1

Introduction

We live in a world in which more and more data is produced and processed than ever before. Different techniques exist for utilising this data and building models and making forecasts with it. However, most of modern approaches hide their functionality in an unexplainable, high-dimensional space and can therefore be seen as "black box" approaches. In contrast, one approach consists of deducing the governing equations from a given time series. This has the advantage of rendering the obtained model understandable and explainable. Such models could be used in the modelling of the climate and turbulent air flows, epidemics modelling, and even in financial markets; all fields, where explainability can be as important as accuracy.

This work is a direct continuation of [1]. In their work, the authors present a framework with which the form of governing equations can be reconstructed by measuring the causality in the resulting time series. Subsequently, given the form of governing equations and the time series, we want to calibrate the equations by finding the coefficients which describe the data best. Current work on this topic exists, however it is limited to the Lorenz system and lacks general applicability [2]. In [3] the authors present an overview of different algorithms for this task. They are based on different approaches such as genetic algorithms [4] and particle swarm optimisations [5]. For our work however, we focus on gradient descent based algorithm, since the recent advent of deep learning technologies [6] lead to a family of improved first-order optimisers [7], [8].

In this work we propose an optimisation algorithm for finding the set of coefficients on a given structure of equations which describe a given time series best. For this we use synchronisation and a touch of machine learning. We treat the discrete data points as a primary system and couple a secondary system to it inducing synchronisation. We then apply gradient descent-algorithms to find the set of parameters minimising the loss between the two systems. The set of parameters minimising the loss are a good approximation of the parameters describing the original dataset.

This work is structured as follows: in the next chapter, 2, we introduce the concept of a complex system with the Lorenz system as the prime example. In chapter 3 we describe the synchronisation of complex systems, where we present different approaches for achieving synchronisation. In chapter 4 we propose our optimisation algorithm and explain its workings in detail. In chapter 5 we apply our algorithm on synthetic systems and investigate its performance on different noise levels and difference levels of confidence in the assumptions of the underlying equations. Additionally, we compare it to a similar algorithm presented in [2]. In chapter 6 we try to apply our algorithm on data stemming from real world applications and additionally break an 'encryption' scheme based on complex systems. We conclude this work with a summary and outlook in chapter 7.

Chapter 2

Overview of Complex Systems

In the context of this work, we define complex systems as non-linear differential equations resulting in chaotic behaviour. While we acknowledge that some authors may differentiate between a "complex system" and a "chaotic system", in this work both terms are used interchangeably. Even though chaotic behaviour has no rigorous, universally accepted, mathematical definition, it manifests itself in various properties. One approach for defining chaotic systems in an intuitive way can be found in [9], where the author presents three properties in order to classify a system as chaotic: aperiodic long-term behaviour, determinism, and sensitive dependence on initial conditions.

In the following we will sketch these properties and illustrate their meaning with ideas borrowed from [9]. Starting with the latter, being the most intuitive one, this property states that small differences in the initial conditions will amplify and will cause the system to be in vastly different states after few iterations. An example for this can be seen in figure 2.1b, where a difference of the order of 10^{-12} in one coordinate will result in a difference of order 1 between the two trajectories. Even though this sensitivity is one of the most prominent features of chaotic systems, it is not sufficient as a definition, since other trivial systems, such as iterating the doubling map $x \mapsto 2x$ for positive values, will also exhibit a similar sensitivity on initial conditions.

Another important property of complex systems is each trajectory not settling down and not converging towards a periodic orbit or a fixed point. Taking the aforementioned example of the doubling map, each trajectory diverges towards positive infinity and never returns from there. So, positive infinity acts as an attractive fixed point for that system, meaning it does not qualify as a complex system. Trajectories of complex systems do not converge, instead they present an aperiodic behaviour in the long run.

Lastly, we want to emphasize that complex systems are deterministic. The irregular behaviour is not based on stochasticity or random forces, instead it is caused due to the non-linear properties of the dynamical system itself. A complex system with the exact same initial conditions should produce the exact same trajectories. In practice this is hard to obtain, since it is impossible to have numerically identical parameters due to the limited precision of computers. By definition, this limited precision will eventually result in diverging trajectories.

An interesting property of complex systems is them having a so-called "*strange attractor*". While the presence of a strange attractor does not imply the existence of a complex system [10], complex systems have strange attractors. Again, it is hard to define an attractor rigorously, however intuitively an attractor can be seen as a set of points to which all neighbouring trajectories converge [9]. So, fixed points and limit cycles are typical examples of attractors. Strange attractors are even harder to define mathematically. Usually their definition includes the presence of a fractal structure. In the context of this work, the exact definition of a strange attractor is not important, and an attractor can be labelled as "strange", if its trajectories diverge locally, but remain confined globally. An example for a strange attractor can be found in figure 2.1a.

2.1 Lyapunov Exponent

In this section we want to formalise the idea of the sensitive dependence on initial conditions. We use the argumentation described in [9]. Experimentally, e.g. in figure 2.1b, we can see that nearby trajectories diverge exponentially fast. Formally this means that we take a point $\underline{x} \in \mathcal{A}$ on an attractor $\mathcal{A}$ and a second point $\underline{x}'$ in the close vicinity of $\underline{x}$. The second point $\underline{x}'$ can be modelled in time t as $\underline{x}'(t) = \underline{x} + \underline{\delta}(t)$, where $\underline{\delta}$ describes the coordinate-wise difference between the two points. Given an initial distance $\underline{\delta}(0)$, we see that the distance between the two points $\underline{x}$ and $\underline{x}'$ is the norm of $\underline{\delta}$. Experimentally, this distance increases exponentially and can therefore be written as:

$$\|\underline{\delta}(t)\| = \|\underline{\delta}(0)\| \, \exp(\lambda\, t) \tag{2.1}$$

Colloquially, the parameter λ is called the "Lyapunov exponent". However, its correct term is "largest Lyapunov exponent", since an n-dimensional system has n Lyapunov exponents. An illustrative definition of Lyapunov exponents can be found in [11]: given an n-dimensional dynamical system, we monitor an n-sphere of initial conditions. Due to its dynamic, this sphere will evolve into an n-ellipsoid. By definition, at time t this n-ellipsoid has n principal axes labelled $p_i(t)$. The i-th Lyapunov exponent can then be defined as:

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \, \ln \frac{p_i(t)}{p_i(0)} \tag{2.2}$$

For analyses however, usually only the largest Lyapunov exponent is relevant, since for large times the diameter of the n-ellipsoid is controlled by the most positive Lyapunov exponent [9]. For this reason, in this work we will use the colloquial term and refer to the "largest Lyapunov exponent" simply as the "Lyapunov exponent".

Given the governing equations of a system, the largest Lyapunov exponent can be calculated in an illustrative way: according to equation 2.1, a logarithmic plot of the distance $\underline{\delta}$ over time t should result in a linear relationship, as can be seen exemplarily in figure 2.1b. The slope of the linear fit, marked in red in figure 2.1b, of the linear part is then the Lyapunov exponent λ.

We need to make three remarks about this calculation [9]: firstly, we can see in figure 2.1b that the line is not perfectly straight, instead it shows some oscillations, since the exact distance depends on the location of the trajectories on the attractor. Secondly, the distance saturates after some time. This happens when the distance between the two starting points reaches the order of the size of the attractor. Since both trajectories will not leave the attractor, their maximum distance is bounded by the size of said attractor. For the calculation only the linear regime of the plot, underlaid in grey in figure 2.1b, shall be considered. Lastly, the exact value of the slope depends slightly on the initial conditions, which is why the average of multiple runs should be considered. In subsection 2.1.1 we will perform a calculation for different synthetic systems. However, we will not be using the illustrative approach, instead we will use a more sophisticated method, which will be sketched in the following section 2.1.1.

The largest Lyapunov exponent is an important measure of a dynamical system. If the exponent is positive, it means that nearby trajectories diverge exponentially fast and that the dynamical system behaves chaotically [12]. Additionally, the largest Lyapunov exponent can be used for defining a time horizon beyond which predictions breaks down [9]. This is done as follows, where we follow the ideas of [9]: consider an initial uncertainty of $\|\underline{\delta}(0)\|$ and a tolerance a, which we use to define a prediction as incorrect if $\|\underline{\delta}(0)\| > a$. Using equation 2.1, we estimate the order of the time t_{horizon}, after which the prediction exceeds our tolerance, as follows:

$$t_{\text{horizon}} \sim \mathcal{O}\left(\frac{1}{\lambda} \, \ln \frac{a}{\|\underline{\delta}(0)\|}\right) \tag{2.3}$$

Motivated by the upper equation, we can define a Lyapunov time as follows, where λ describes the largest Lyapunov exponent [13]:

$$\tau_\lambda = \lambda^{-1} \tag{2.4}$$

The Lyapunov time can be used as a measure for defining the quality of a prediction for chaotic systems. We will use this measure throughout this work.

2.1.1 Calculation of Lyapunov Exponents for Different Systems

In this section we will discuss and perform the calculations of the largest Lyapunov exponent for a variety of synthetic chaotic systems. In the past section, we defined the largest Lyapunov exponent as the slope of the linear part of the logarithmic distance of two nearby trajectories over time. Therefore, the calculation would require finding the linear regime of the logarithmic distance. However, this turns out to be a non-trivial task, and even though the upper definition is a great visual definition of the Lyapunov exponent, for its calculation more sophisticated methods have been developed. In this work we use the method described in [14]. The algorithm is sketched in the following section.

The main focus of this work is not the calculation of the largest Lyapunov exponents for different chaotic systems. In fact, the Lyapunov exponents are only introduced for defining the quality of a prediction for chaotic systems by utilising the Lyapunov time. Therefore, we cannot build the complete mathematical framework described in [14] needed to fully understand the algorithm. Instead, we sketch the main ideas of the algorithm and refer to [14] for a proper mathematical treatment. For an easier understanding, we use the notation of the aforementioned article in this section and in the algorithm 2.1.

Given a time t dependent deviation $\underline{w}(t)$ from a point on an attractor, the largest Lyapunov exponent λ can be calculated in the limit of the following quantity X. This idea is analogous to the definition used in equation 2.2.

$$X(t) = \frac{1}{t} \ln \frac{\|\underline{w}(t)\|}{\|\underline{w}(0)\|} \tag{2.5}$$

$$\lambda = \lim_{t \to \infty} X(t) \tag{2.6}$$

However, numerically we run into problems, since the quantity $\underline{w}(t)$ grows exponentially in time t, and we cannot store arbitrarily large numbers. To overcome this problem, we use the deterministic behaviour of our system. Therefore, we can reach a time t by integrating towards it piecewise with small time steps τ until we reach $t = k\tau$, with $k \in \mathbb{N}$. Now, we can rewrite equation 2.5 as follows, where we expand the upcoming fraction with the product $\|\underline{w}(\tau)\| \cdots \|\underline{w}((k-1)\,\tau)\|$.

$$\begin{aligned} X(k\tau) &= \frac{1}{k\tau} \ln \frac{\|\underline{w}(k\tau)\|}{\|\underline{w}(0)\|} \\ &= \frac{1}{k\tau} \ln \left(\frac{\|\underline{w}(k\tau)\|}{\|\underline{w}((k-1)\tau)\|} \, \frac{\|\underline{w}((k-1)\tau)\|}{\|\underline{w}((k-2)\tau)\|} \cdots \frac{\|\underline{w}(2\tau)\|}{\|\underline{w}(\tau)\|} \, \frac{\|\underline{w}(\tau)\|}{\|\underline{w}(0)\|} \right) \\ &= \frac{1}{k\tau} \sum_{i=1}^{k} \ln \frac{\|\underline{w}(i\tau)\|}{\|\underline{w}((i-1)\tau)\|} \end{aligned} \tag{2.7}$$

Denoting the norm of the initial deviation $\underline{w}(0)$ with D_0, and the norm of the i-th iteration of $\underline{w}(i\tau)$ with D_i, the author in [14] shows that we can rewrite equation 2.7 as follows:

$$X(k\tau) = \sum_{i=1}^{k} \ln \frac{D_i}{D_0} =: \sum_{i=1}^{k} \ln \alpha_i \tag{2.8}$$

Using equations 2.6 & 2.8, we can now rewrite the largest Lyapunov exponent, for understanding the algorithm 2.1, as:

$$\lambda = \lim_{k \to \infty} \frac{1}{k\tau} \sum_{i=1}^{k} \ln \alpha_i \tag{2.9}$$

For the actual calculation, we use a unitary vector for $\underline{w}(0)$, so its norm $D_0 = 1$, and is therefore not relevant for the calculation. Additionally, after we calculate α_i, we renormalise the vector $\underline{w}(k\tau)$ to having a norm of 1, but keeping its direction. For each successive time interval, we therefore compute α_i and in the end, we estimate the largest Lyapunov exponent λ using 2.9.

In our implementation of the algorithm, we stay close to the algorithm presented in [14], however we extend it by two small points. Firstly, we take the average over a number of runs, since the Lyapunov exponent depends slightly on the initial conditions of the system. Secondly, for each run we use a point on the attractor as initial condition for that run. This is achieved in our algorithm by utilising a random point from the so-called "master sequence" as initial condition for the subsequent run. The algorithm of [14] together with our modifications leads to following algorithm 2.1 for calculating the largest Lyapunov exponent of any dynamical system.

```
master sequence ← integrate(global initial condition, $N_m$, $\Delta t$)
master sequence ← trim from beginning(master sequence, $\frac{1}{10} N_m$)
$\Lambda$ ← initialise empty list($N_e$)
for $i \leftarrow 1 \dots N_e$ do
    local initial condition ← choose random point(master sequence)
    chaotic sequence ← integrate(local initial condition, $N_s$, $\Delta t$)
    $\underline{w}$ ← random normed vector( )
    $A$ ← initialise empty list($N_s - 1$)
    for $k \leftarrow 1 \dots N_s - 1$ do
        next deviated step ← integrate(chaotic sequence[$k$] + $\underline{w}$, 1, $\Delta t$)
        $\underline{w}$ ← next deviated step − chaotic sequence[$k + 1$]
        $\alpha \leftarrow \|\underline{w}\|$
        $A[k] \leftarrow \alpha$
        $\underline{w} \leftarrow \frac{\underline{w}}{\alpha}$
    end
    $\lambda \leftarrow {}^{1}\!/_{(N_s - 1)\,\Delta t} \sum_{s=1}^{N_s-1} \ln A[s]$
    $\Lambda[i] \leftarrow \lambda$
end
return mean($\Lambda$) ± standard deviation($\Lambda$)
```

Algorithm 2.1: This algorithm was adapted from [14] and returns the largest Lyapunov exponent for a given dynamical system. The exact value of the "global initial condition" is not important, as long as it ensures a convergence towards the attractor. We use the convention to store the lower-case values in an upper-case list: the values for α are stored in the list A, similarly for λ and Λ. The calculations have been performed using $N_m = 10^5$ and $\Delta t = 10^{-3}$. N_e describes the number of times to estimate the largest Lyapunov exponent. Here, we chose to run each chaotic system for a thousand times, meaning $N_e = 10^3$. N_s describes the number of steps to integrate each realisation of the dynamical system. In this work we used $N_s = 2 \times 10^5$.

The results of upper calculation can be found in table 2.1: as expected, for each chaotic system the largest Lyapunov exponent is greater than zero. For some systems, such as the Four-wing system or the Thomas system, the Lyapunov exponent is barely above zero meaning that their nearby trajectories do not diverge as strong as for the Lorenz63 system.

Table 2.1: This table shows the results of the calculations of the largest Lyapunov exponent. For each system, the standard parameters, which are defined at the introduction of each system, have been used. For the Lorenz63 system the parameters can be consulted in section 2.2. The definitions and parameters of the remaining systems can be found in the appendix A.

Chaotic system	**Largest Lyapunov exponent**
Dadras–Momeni system	0.60 ± 0.05
Four-wing system	0.007 ± 0.014
Halvorsen system	0.321 ± 0.007
Lorenz63 system	0.902 ± 0.013
Lorenz86 system	0.63 ± 0.04
Lotka–Volterra system	0.21 ± 0.04
Rabinovich–Fabrikant system	0.05 ± 0.03
Rössler system	0.067 ± 0.008
Thomas system	0.009 ± 0.011
Three-scroll unified system	0.161 ± 0.012

2.2 The Lorenz System

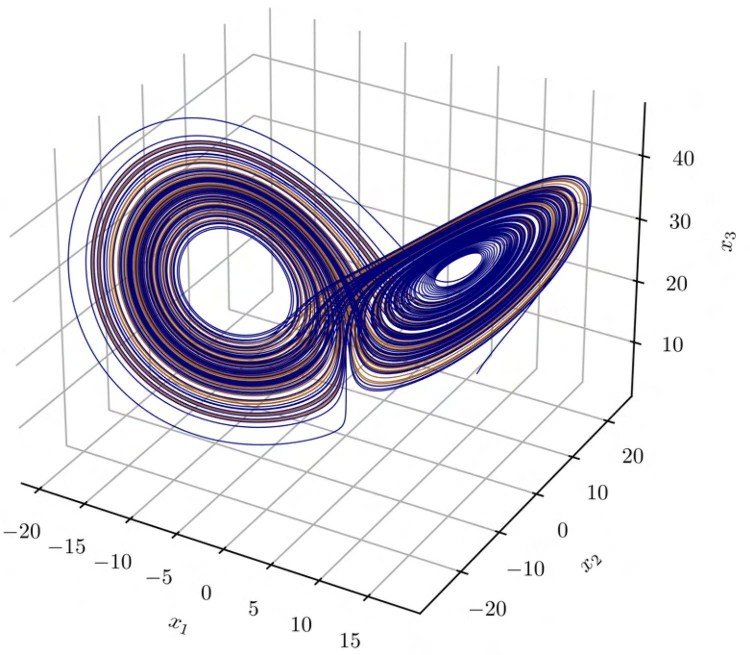
The Lorenz63 system, colloquially just referred to as "Lorenz system", is probably the most prominent example of a chaotic system. Initially used as a model for atmospheric convection [15], it has since been used as the prime object of study for chaotic, dynamical systems. We take this example to illustrate the ideas introduced in section 2.

The Lorenz63 system is a three-dimensional system of ordinary differential equations. It is defined as follows [15]:

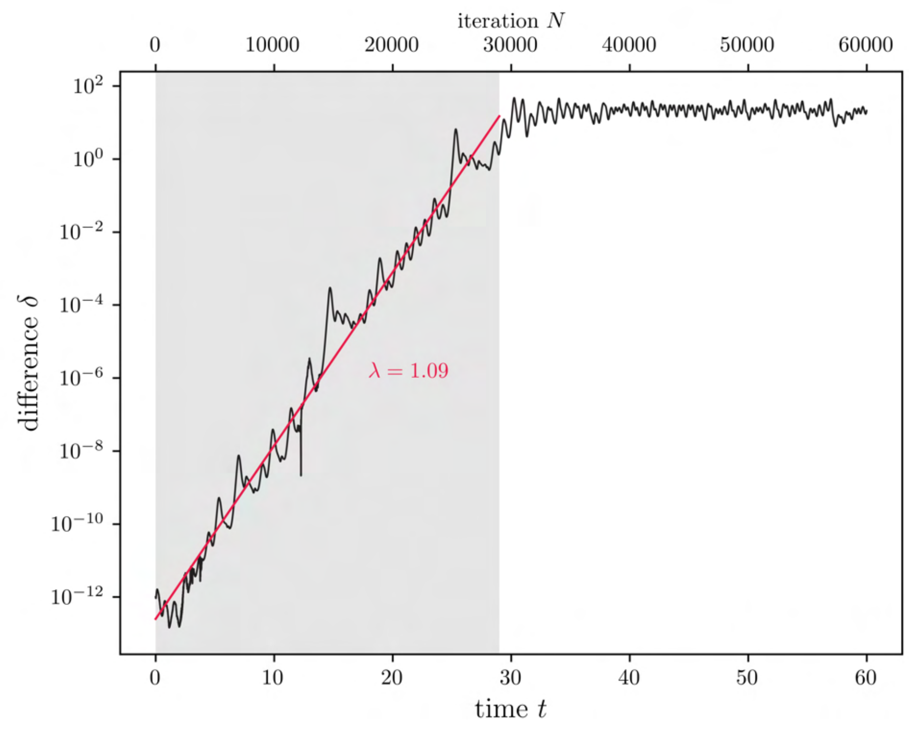
$$\begin{aligned}\dot{x}_1 &= -\sigma\, x_1 + \sigma\, x_2 \\ \dot{x}_2 &= -x_1\, x_3 + \rho\, x_1 - x_2 \\ \dot{x}_3 &= x_1\, x_2 - \beta\, x_3\end{aligned} \tag{2.10}$$

The standard parameters known to cause chaotic behaviour are: $\sigma = 10$, $\rho = 28$, and $\beta = \frac{8}{3}$. If not stated otherwise, the Lorenz63 system will be used exclusively with these parameters in this work.

Figure 2.1 visualises the characteristics of complex system presented in section 2. Figure 2.1a shows the attractor of the Lorenz system, and there the globally confined structure of the attractor becomes apparent. Concerning the Lyapunov exponent, figure 2.1b shows the exponentially fast diverging trajectories for almost identical initial conditions. However, the trajectories do not diverge forever, instead their difference settles down to the order of the size of the attractor. As the most prominent complex system, the Lorenz system depicts all typical features very prominently.



(a)



(b)

Figure 2.1: Figure (a) shows two trajectories of the Lorenz system on the attractor. Looking at the shape, we can see where the term "strange" comes from. For both trajectories the standard parameters have been used and each trajectory has been integrated for 10^6 steps with an integration step of 10^{-3}. The first trajectory in blue has the initial condition $(1,1,1)^\intercal$, and the second trajectory in brown has the initial condition $(1+10^{-12},1,1)^\intercal$. Since both colours are clearly visible in the plot, this shows that the trajectories take different paths on the attractor. Figure (b) shows the norm of the difference between the two trajectories on a logarithmic scale. We can clearly observe a linear dependence in the beginning phase marked in grey, from which the largest Lyapunov exponent λ can be estimated through the linear interpolation shown in red.

Chapter 3

Synchronisation of Complex Systems

Coupled dynamical systems can exhibit an interesting behaviour of synchronisation. Synchronisation can be understood as a strong correlation between the behaviours of two or more systems, and in its most elementary form it describes the property of two systems behaving identically. In this chapter we will look at two different coupling methods and we will analyse whether they will eventually lead to synchronisation. The first coupling method describes finding a variable, which drives a sub-system to synchronisation, whereas the second method introduces an additional term for inducing synchronisation in the systems. This chapter will mostly follow the ideas of [16] and we refer to it for a more detailed analysis.

3.1 Synchronisation Through a Driving Variable

In [17] the authors noted the possibility to synchronise certain complex systems. For that they divided the system into a driving variable and into a sub-system. By choosing a specific driving variable and inserting it into a secondary sub-system, they were able to synchronise the secondary sub-system to the primary system supplying the driving variable. The necessary criterion for synchronisation seems to be a negative sign for all Lyapunov exponents of the sub-system [17].

While in their article they presented this phenomenon for the Lorenz and Rössler system, in this work we will use the Lorenz system as the prime example. Table 3.1 indicates that for the Lorenz system the first two variables can be used to create a secondary system driven by a primary variable. Exemplary, taking x_1 as the driving variable, we can construct the governing equations of the coupled system as follows. The primary system is described by the standard Lorenz equations 2.10 and the secondary system is described by the following equations:

$$\begin{aligned} \dot{y}_1 &= \dot{x}_1 \\ \dot{y}_2 &= -x_1\, y_3 + \rho\, x_1 - y_2 \\ \dot{y}_3 &= x_1\, y_2 - \beta\, y_3 \end{aligned} \tag{3.1}$$

The equations are analogous to the equations in 2.10, with the significant difference being that the variable y_1 has been replaced with the driving variable x_1. Meaning, there is no additional term for the variable $\dot{y}_1$, since by definition the behaviour of the variable y_1 is described entirely by x_1. Figure 3.1 visualises the connection between the primary and the secondary system and it emphasises the distinction between the driving variable x_2 and the resulting sub-system (y_1, y_3).

We precede analogously for the remaining two driving variables and define the responding sub-systems accordingly. The results are shown in figure 3.2, where the observations of [17] can be reproduced: We see in figures 3.2a and 3.2b that using x_1 and x_2 as driving variables will lead to synchronisation of the systems. However with x_3 as the driving variable, the systems will not synchronise.

Table 3.1: This table, copied from [17], shows the Lyapunov exponents for the Lorenz system for each possible permutation of driving variable and resulting sub-system. The article uses the Lorenz system with the parameters $\sigma = 10$, $\rho = 60$, and $\beta = \frac{8}{3}$.

Driving variable	Responding sub-system	Lyapunov exponents of responding sub-systems
x_1	(y_2, y_3)	$(-1.81, -1.86)$
x_2	(y_1, y_3)	$(-2.67, -9.99)$
x_3	(y_1, y_2)	$(+0.0108, -11.01)$

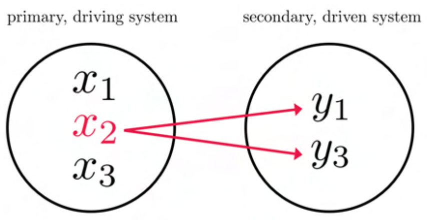
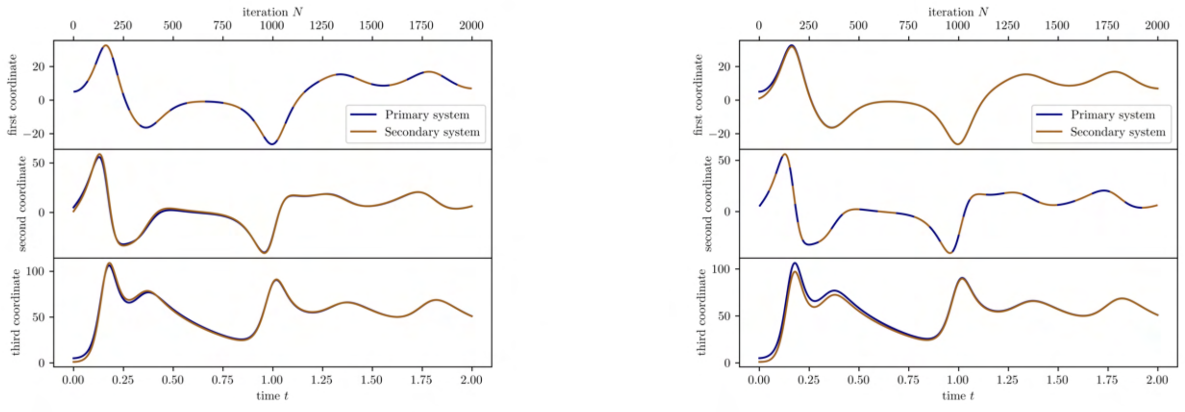
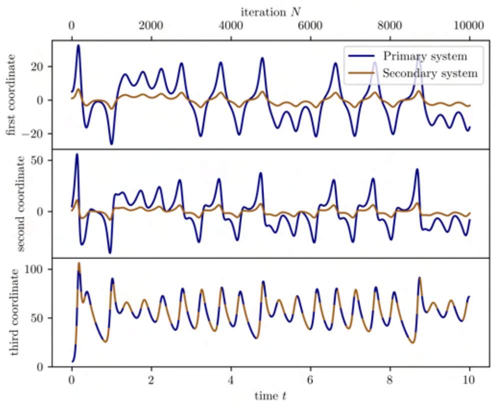


Figure 3.1: This figure shows the connection between a primary system and a secondary, driven system taking the Lorenz system, with the configuration shown in the second row of table 3.1, as an example. More specific, it illustrates how the driving variable of the primary system affects the secondary, driven system.



(a) x_1 as driving variable

(b) x_2 as driving variable



(c) x_3 as driving variable

Figure 3.2: This figure shows the synchronisation (and in case (c) the lack thereof) for two coupled Lorenz systems. Figure (a) shows x_1, figure (b) shows x_2, and figure (c) shows x_3 as the driving variable. The sub-systems are defined as stated in table 3.1. For each Lorenz system we used the parameters $\sigma = 10$, $\rho = 60$, and $\beta = \frac{8}{3}$ and an integration step of 10^{-3}. The primary system has the initial condition $(5, 5, 5)^\intercal$ and each secondary system has the initial condition $(1, 1)^\intercal$.

3.1.1 Synchronisation Using Active-Passive-Decomposition

While the authors in [17] present a rather descriptive view on this topic, the idea is generalised and formalised in [18]. In the following we want to present their ideas, since this kind of synchronisation is used in comparable work such as [2] and [16]. So, the following passage describes the active-passive decomposition as presented in [18].

An arbitrary N-dimensional dynamical system can be described the following way:

$$\underline{x}_{n+1} = F(\underline{x}_n; \underline{\theta}) \tag{3.2}$$

In this case, $\underline{x}_n \in \mathbb{R}^N$ is the state of the system at step n and $\underline{\theta} \in \mathbb{R}^m$ encapsulates the m parameters of the dynamical system F: $\mathbb{R}^N \to \mathbb{R}^N$. Here, $\underline{x}$ is an autonomous system. However, we want to rewrite this system into a non-autonomous one driven by a time-dependant signal $\underline{s}_n \in \mathbb{R}^{N'}$ with $N' \leq N$. Formally, we can describe this as:

$$\underline{x}_{n+1} = p(\underline{x}_n, \underline{s}_n; \underline{\theta}) \tag{3.3}$$

The function p: $\mathbb{R}^N \times \mathbb{R}^{N'} \to \mathbb{R}^N$ is the non-autonomous decomposition of F. The signal $\underline{s}_n$ is also a function of the state $\underline{x}_n$ and can formally be defined as:

$$\underline{s}_n = a(\underline{x}_n) \quad \text{with } a\text{: } \mathbb{R}^N \to \mathbb{R}^{N'} \tag{3.4}$$

The functions a and p are a decomposition of the original system F [18]. Concretely, this means that replacing the signal into the description of 3.3 will lead back to equation 3.2:

$$p(\underline{x}_n, a(\underline{x}_n); \underline{\theta}) = F(\underline{x}_n; \underline{\theta}) \tag{3.5}$$

Now we define a secondary system $\underline{y}_n$ analogously to equation 3.3 as follows:

$$\underline{y}_{n+1} = p\Big(\underline{y}_n, \underline{s}_n; \underline{\theta}\Big) \tag{3.6}$$

The key idea now is, that for specific choices of the function a any secondary system $\underline{y}_n$ will synchronise to the primary system $\underline{x}_n$, meaning:

$$\lim_{n\to\infty} \left\|\underline{y}_n - \underline{x}_n\right\| = 0 \tag{3.7}$$

Note that for each secondary system $\underline{y}$ the function p and the parameters $\underline{\theta}$ have to be identical, solely the coordinates in $\underline{y}$ may differ. The synchronisation in equation 3.7 can be reformulated in a different way using the error term $\underline{e} = \underline{x} - \underline{y}$: if the time evolution of the error term has a stable fixed point at the origin $\underline{e} = 0$, the systems 3.2 and 3.6 synchronise. The time evolution of the error is given by:

$$\begin{aligned} \underline{e}_{n+1} &= p\Big(\underline{y}_n, \underline{s}_n; \underline{\theta}\Big) - p(\underline{x}_n, \underline{s}_n; \underline{\theta}) \\ &= p(\underline{x}_n + \underline{e}_n, \underline{s}_n; \underline{\theta}) - p(\underline{x}_n, \underline{s}_n; \underline{\theta}) \end{aligned} \tag{3.8}$$

For some systems this can be proven analytically be analysing the stability of the linearised system, where Dp denotes the Jacobian matrix.

$$\underline{e}_{n+1} = \mathrm{D}p(\underline{x}_n, \underline{s}_n; \underline{\theta})\ \underline{e}_n \tag{3.9}$$

However, in general the stability has to be checked numerically by requiring that all Lyapunov exponents of the non-autonomous sub-system are negative as described by [17]. If this condition is met, p is the passive and a is the active decomposition of the system 3.2.

Active-Passive-Decomposition of the Lorenz System

We want to demonstrate the analytical application of equation 3.9 and for that purpose we use the Lorenz system, since in comparable work [2] the active-passive-decomposition of the Lorenz system is introduced, but not proven.

In aforementioned article the authors propose the following decomposition of the Lorenz system, where equation 3.10 describes the passive decomposition and 3.11 describes the scalar, driving signal.

$$p\left(\underline{y}_n, s_n; \underline{\theta}\right) = \begin{pmatrix} -\sigma\, y_1 + \sigma\, s_n \\ -y_1\, y_3 + \rho\, y_1 - y_2 \\ y_1\, y_2 - \beta\, y_3 \end{pmatrix} \tag{3.10}$$

$$a(\underline{x}_n) = x_2 = s_n \tag{3.11}$$

This leads to the following set of equations for the secondary, coupled Lorenz system:

$$\begin{aligned} \dot{y}_1 &= -\sigma\, y_1 + \sigma\, x_2 \qquad (3.12)\\ \dot{y}_2 &= -y_1\, y_3 + \rho\, y_1 - y_2 \\ \dot{y}_3 &= y_1\, y_2 - \beta\, y_3 \end{aligned}$$

The existence of a fixed point at $\underline{e} = 0$ is trivial using equation 3.8. The interesting question regards the stability of that fixed point, and for that we utilise equation 3.9 and calculate the Jacobian matrix of 3.10:

$$\mathrm{D}p = \begin{pmatrix} -\sigma & 0 & 0 \\ \rho - y_3 & -1 & -y_1 \\ y_2 & y_1 & -\beta \end{pmatrix} \tag{3.13}$$

This Jacobian matrix has the following eigenvalues:

$$\nu_1 = -\sigma \tag{3.14a}$$

$$\nu_{2,3} = \frac{1}{2}\left(\pm\sqrt{\beta^2 - 2\beta - 4y_1^2 + 1} - \beta - 1\right) \tag{3.14b}$$

Evaluating the eigenvalues at the standard parameters $\underline{\theta} = \left(10, 28, \frac{8}{3}\right)^\mathsf{T}$ leads to all negative real-parts of the eigenvalues, indicating that the fixed point is stable. The stability of the fixed point in turn shows that the signal in 3.11 can be used to drive any passive decomposition 3.10 as visualised in figure 3.3, where synchronisation is shown for a variety of differing initial conditions. We want to emphasize at this point that the parameters $\underline{\theta}$ for each system have to be identical — and in this specific case be the standard parameters for the Lorenz system — for synchronisation to occur.

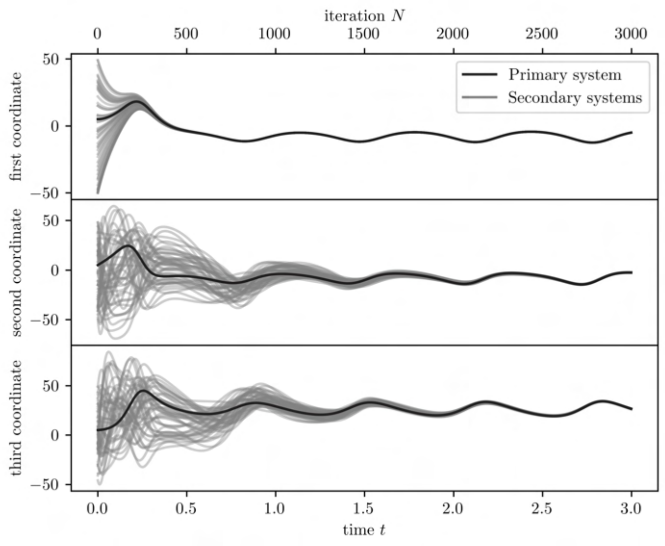


Figure 3.3: This figure visualises the synchronisation of numerous secondary systems (each defined by equation 3.10) to a single primary system providing the signal as specified in equation 3.11. We can see clearly that eventually, each secondary system behaves identically to the primary system. For each Lorenz system the standard parameters have been used with an integration step of 10^{-3}. The primary system has the initial condition $(5, 5, 5)^\intercal$, whereas for the initial condition of each secondary system three random numbers between -50 and 50 were used. This figure shows fifty secondary systems.

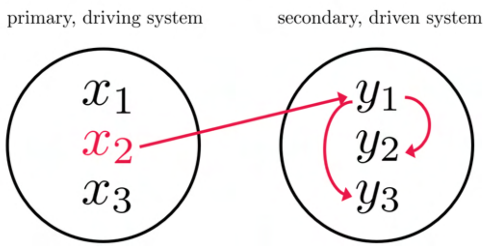


Figure 3.4: This figure shows the connection between a primary system and a secondary, driven system for the active-passive decomposition of the Lorenz system described in equations 3.10f.

Comparing the figures 3.1 and 3.4, we can see that in both cases we have a single variable causing the synchronisation. However, the active-passive decomposition is a more subtle way to synchronise two systems, since instead of replacing a whole variable and essentially propagating a system with reduced dimensionality, as it is done in figure 3.1, the active-passive-decomposition inserts a single variable, which in turn leads to synchronisation. Notice that in both cases, the secondary systems are synchronised to the primary system, meaning that in the long run, the secondary systems will behave exactly like the primary system.

Once found, the active-passive-decomposition works exceptionally well. Nevertheless, the major limitation of this method is finding a suitable decomposition. Performing the calculations in equations 3.10ff. is time-intensive. In practice, those calculations are emitted entirely in favour of a simple "trial-and-error" approach for finding a suitable decomposition leading to synchronisation. However, as of today, there is limited research on the existence of such a driving variable for general complex systems and therefore, there is no guarantee that such a driving variable must exist.

Even though this method is highly limited to a handful of systems, it is used in comparable work to induce required synchronisation [2]. This means that the algorithm presented in aforementioned article is limited to such systems, where a driving variable can be discovered. In this work we aim to generalise the results and propose an algorithm, which works for arbitrary complex systems by using an external coupling to induce synchronisation. For that, we need a general way of inducing synchronisation, which we present in the following section.

3.2 Synchronisation of General Systems by External Coupling

In the past section we introduced a way of synchronising two chaotic systems. However, that method only worked for specific systems and no generalisation could be formulated. In this section we will present a general method for synchronising two non-linear systems. At first, we will synchronise the systems to each other citing the findings by [16]. However, for our calibration we need a unidirectional coupling with a driving system and a driven system in the framework of section 3.1. We will develop the unidirectional coupling as a corollary of the bidirectional coupling in a second step in section 3.2.2.

3.2.1 Bidirectional Synchronisation

In their article the authors presented a general way of synchronising two systems to each other. Using their method, both systems will synchronise towards a common behaviour. We will present those findings and follow the argumentation of that article closely in subsequent section [16].

Two coupled, arbitrary N-dimensional systems can be described by following equations:

$$\dot{\underline{x}} = F(\underline{x}) + \gamma\,\mathbf{H}\big(\underline{y} - \underline{x}\big) \tag{3.15a}$$

$$\dot{\underline{y}} = F\big(\underline{y}\big) + \gamma\,\mathbf{H}\big(\underline{x} - \underline{y}\big) \tag{3.15b}$$

Here $\underline{x}$, $\underline{y} \in \mathbb{R}^N$ are the states of each system, and F: $\mathbb{R}^N \to \mathbb{R}^N$ describes the evolution of the original dynamical system in the absence of any coupling. We emphasize that the states $\underline{x}$ and $\underline{y}$ are naturally time dependant, but for the sake of readability we refrain from explicitly noting it. However, time dependence is always implied. Note, that the description has to be identical for both systems. The matrix $\mathbf{H}$: $\mathbb{R}^N \to \mathbb{R}^N$ describes the coupling between the two systems $\underline{x}$ and $\underline{y}$, and the parameter $\gamma \in \mathbb{R}^+$ named "coupling strength" determines the strength of the coupling. We require $\mathbf{H}(\underline{0}) = \underline{0}$, meaning that for synchronised systems the coupling vanishes and both systems evolve without the influence of the additional coupling term. We can illustrate the need for this requirement by two intuitive arguments: firstly, if for two initial conditions the systems were already synchronised, the coupling should not have any effect. Secondly, once both systems are eventually synchronised, the coupling should not affect their further evolution.

In their article [16], the authors show that if the coupling is sufficiently strong, the systems described by equations 3.15 will synchronise. Adapting the criterion in equation 3.7 to the continuous case, synchronisation between two systems is defined by: $\lim_{t\to\infty} \big\|\underline{y} - \underline{x}\big\| = 0$. In the following we will present the argumentation of aforementioned article.

For the rest of the argument we consider the identity coupling, meaning $\mathbf{H} = \mathbf{1}$, with $\mathbf{1}$ being the identity matrix. This simplifies the coupling terms in equations 3.15 to $\gamma\,\mathbf{H}\big(\underline{y} - \underline{x}\big) = \gamma\,\big(\underline{y} - \underline{x}\big)$ and $\gamma\,\mathbf{H}\big(\underline{x} - \underline{y}\big) = \gamma\,\big(\underline{x} - \underline{y}\big)$ respectively. Additionally, we define a difference variable $\underline{z} = \underline{x} - \underline{y}$. The evolution of the new variable $\underline{z}$ can be described as follows, using equations 3.15:

$$\begin{aligned}\dot{\underline{z}} &= \dot{\underline{x}} - \dot{\underline{y}}\\ &= F(\underline{x}) - F\big(\underline{y}\big) - 2\,\gamma\,\underline{z}\end{aligned} \tag{3.16}$$

Furthermore, synchronisation can be redefined with respect to new variable $\underline{z}$ as follows: $\lim_{t\to\infty} \underline{z} = \underline{0}$. Since we are interested in synchronisation, meaning $\underline{z} = \underline{x} - \underline{y} = 0$, we can Taylor-expand the system $F\big(\underline{y}\big)$ at said criterion $\underline{x} = \underline{y}$, leading to:

$$\begin{aligned}F\big(\underline{y}\big) &= F(\underline{x}) + \mathrm{D}F(\underline{x})\ \big(\underline{y} - \underline{x}\big) + \mathcal{O}\Big(\big\|\underline{y} - \underline{x}\big\|^2\Big)\\ &= F(\underline{x}) - \mathrm{D}F(\underline{x})\,\underline{z} + \mathcal{O}\Big(\|\underline{z}\|^2\Big)\end{aligned} \tag{3.17}$$

Again, $\mathrm{D}F(\underline{x})$ describes the Jacobian matrix of $F(\underline{x})$. Plugging in the Taylor-expansion in equation 3.17 into the definition of the difference variable $\underline{z}$ in equation 3.16 leads to the following first-order approximation at the synchronisation criterion for $\underline{z}$:

$$\dot{\underline{z}} = (\mathrm{D}F(\underline{x}) - 2\,\gamma\,\mathbf{1})\ \underline{z} \tag{3.18}$$

We note that equation 3.18 is a non-autonomous equation, since it depends explicitly on the solution $\underline{x}$, so analysing it is non-trivial. For an easier treatment, we introduce the variable $\underline{w} = \exp\left(2\,\gamma\,t\right)\underline{z}$, in order to get rid of the term $-2\,\gamma\,\mathbf{1}\,\underline{z}$ in equation 3.18. The derivative of the new variable $\underline{w}$ is given by following expression, where we have to apply the chain rule considering the time dependence correctly:

$$\dot{\underline{w}} = 2\,\gamma\exp\left(2\,\gamma\,t\right)\underline{z} + \exp\left(2\,\gamma\,t\right)\dot{\underline{z}} \tag{3.19}$$

Using the definition of $\underline{w}$ and its derivative in equation 3.19, equation 3.18 now simplifies to:

$$\dot{\underline{w}} = (\mathrm{D}F(\underline{x}))\ \underline{w} \tag{3.20}$$

Equation 3.20 is the first variational equation for $\underline{x}$ solving $\dot{\underline{\xi}} = F\big(\underline{\xi}\big)$.

Now, let $\mathbf{\Phi}$ be the fundamental matrix for equation 3.20. This matrix is a function of $\underline{x}$. By definition, this means that any solution of $\underline{w}$ can be written as $\underline{w} = \mathbf{\Phi}\,\underline{w}(0)$ for any initial condition $\underline{w}(0)$, and by substitution the same can be done for $\underline{z}$. In a next step, we consider the matrix $\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi}$. In [19] the author states, that the largest Lyapunov exponent is the square root of the largest eigenvalue of the matrix $\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Phi}$. Therefore, if λ denotes the largest Lyapunov exponent of $\underline{x}$, the following inequality holds for the difference variable $\underline{w}$ [16], [20]:

$$\|\underline{w}\| \leq C\ \exp\left(\lambda\,t\right) \tag{3.21}$$

Resubstituting the definition of $\underline{z}$ into upper equation yields the following inequality:

$$\|\underline{z}\| \leq C\ \exp\left(\left(\lambda - 2\,\gamma\right)t\right) \tag{3.22}$$

Reminding that $\underline{z}$ describes the difference of the two systems $\underline{x}$ and $\underline{y}$, we see that if the exponent in equation 3.22 is negative, the difference between the systems $\underline{x}$ and $\underline{y}$ will tend to 0 as time progresses. Therefore, we define a critical coupling γ_{c} so that the exponent is negative:

$$\gamma_{\mathrm{c}} = \frac{\lambda}{2} \tag{3.23}$$

If the coupling strength γ is bigger than γ_{c}, the two systems will synchronise. This finding is visualised in figure 3.6 for two chaotic systems, where we can clearly observe a sharp drop in the synchronisation loss between the two systems. For that, we defined the synchronisation loss as the average deviation from synchronisation between two systems as follows:

$$E_{\mathrm{s}} = \frac{1}{n}\sum_{i=1}^{n}\left\|\underline{x}_i - \underline{y}_i\right\| \tag{3.24}$$

Meaning that for each iterated step i in the number of analysing iteration steps n, we calculated the norm of $\underline{x}_i - \underline{y}_i$ and summed it in order to form an error over the interval n. This error shows how much, on average, the two systems differ from each other. Figure 3.6 shows that if the coupling between the two systems exceeds the critical coupling as defined in equation 3.23, the two systems will synchronise with each other: hence the synchronisation loss drops to zero after the red line indicating the critical coupling strength, which is half the Lyapunov exponent.

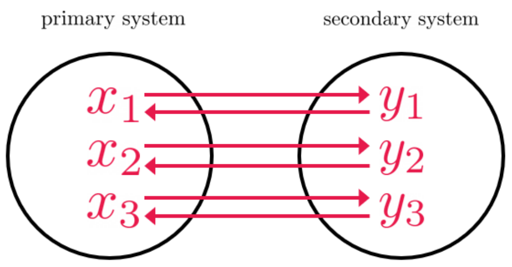
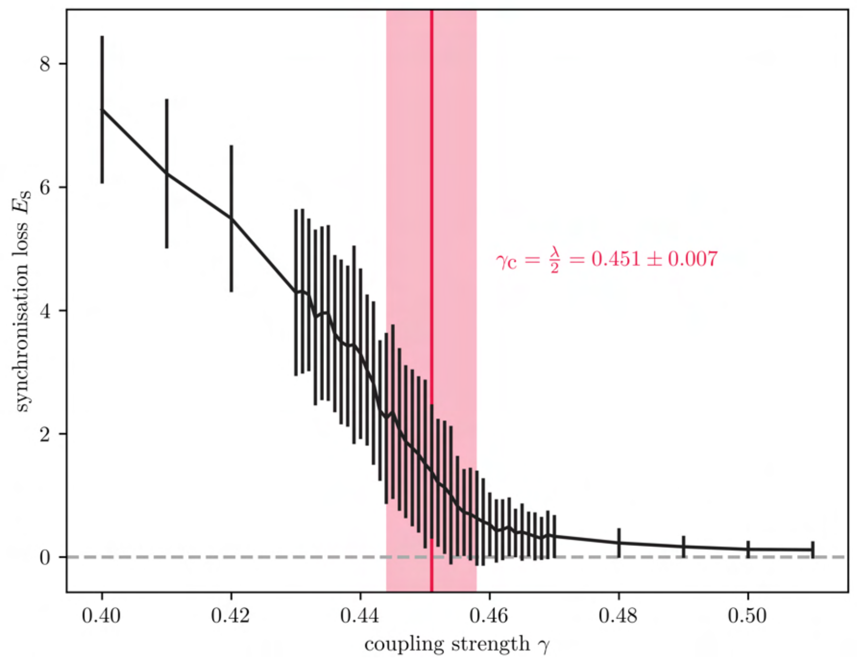


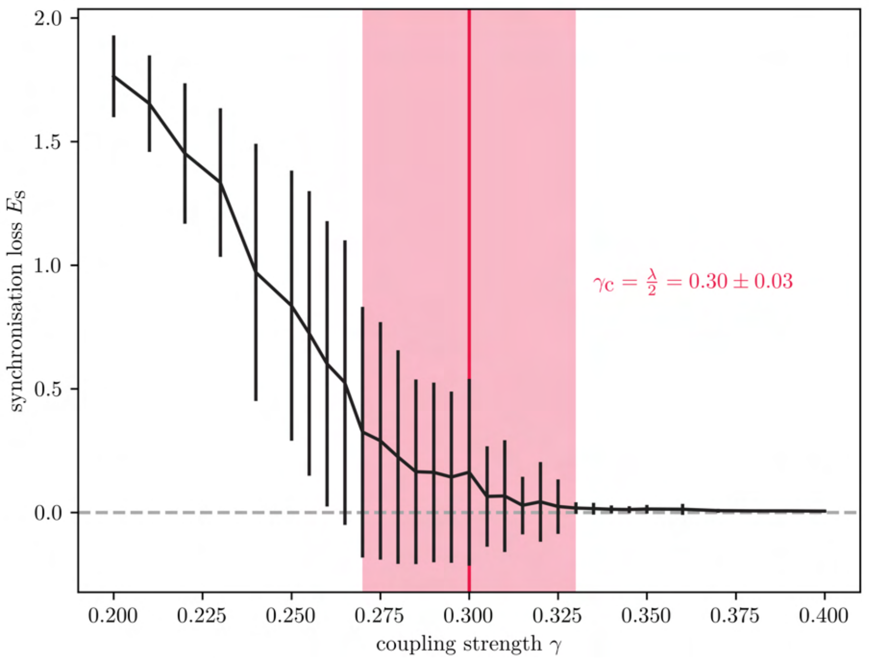
Figure 3.5: This figure shows the connection between a primary system and a secondary system for the bidirectional external coupling, as described in equations 3.15, using the identity matrix as coupling matrix **H**. This representation describes an arbitrary three-dimensional chaotic system.

This method of synchronising two systems has one caveat however. As sketched in figure 3.5, the influence of this coupling method is bidirectional. This is in major contrast to the previous coupling schemes in figures 3.1 & 3.4, where the coupling was directed from the primary system towards the secondary one. This means that with the external coupling, it is not possible to make a secondary system behave identically to a primary system and keep the primary system unchanged. Instead, the difference between primary and secondary system tends towards zero and both systems will behave identically to each other, but differently than they would have without the coupling term. For our calibration method however, we require a clear distinction between a driving system and a driven system. In the next section we will propose a coupling introducing this distinction while maintaining the generality of this method.

We want to mention that the schemes in figures 3.1 & 3.4 are specific to the Lorenz system, whereas the scheme in figure 3.5 is system-agnostic, meaning it represents any three-dimensional complex system. This illustrates the generality of this approach compared to the limited use case of previous methods.



(a) Lorenz system



(b) Dadras–Momeni system

Figure 3.6: Figure (a) shows the synchronisation loss E_s, as defined in equation 3.24, for the Lorenz system and figure (b) shows it for the Dadras–Momeni system. The definition of the latter system can be found in appendix A.1. For both systems their respective standard parameters have been used and for each coupling strength γ a hundred iterations have been calculated. For each iteration an integration step of 10^{-2} has been used and iterated for a total of 10^6 steps. For the calculation of the error the last 10^5 steps have been used. The initial condition for each Lorenz system were three random numbers between 1 and 20, and for each Dadras–Momeni system three random numbers between 0 and 1. The red line shows half the largest Lyapunov exponent as calculated in table 2.1.

3.2.2 Unidirectional Synchronisation

For our calibration algorithm we need a clear distinction between a driving system and a driven system, since we want vary the parameters of the driven system until the system synchronises to the primary system. In order to achieve the clear distinction, we propose the following new coupling method as a corollary to the previous coupling discussed in section 3.2.1.

Given two arbitrary N-dimensional systems $\underline{x}$, $\underline{y} \in \mathbb{R}^N$, we propose adding the coupling $\alpha\,\mathbf{H}\big(\underline{x} - \underline{y}\big)$ to the system $\underline{y}$; thus rendering it the driven system. The system $\underline{x}$ then becomes the primary, driving system. This leads to the following description:

$$\dot{\underline{x}} = F(\underline{x}) \tag{3.25a}$$

$$\dot{\underline{y}} = F\big(\underline{y}\big) + \alpha\,\mathbf{H}\big(\underline{x} - \underline{y}\big) \tag{3.25b}$$

Again, $F\colon \mathbb{R}^N \to \mathbb{R}^N$ describes the evolution of the dynamical system and the matrix $\mathbf{H}\colon \mathbb{R}^N \to \mathbb{R}^N$ describes the coupling of the secondary system to the primary system. The matrix $\mathbf{H}$ is subject to the same constraint as in section 3.2.1 and we fix it to the identity matrix $\mathbf{1}$ for the scope of this work. The parameter $\alpha \in \mathbb{R}^+$, named "coupling strength", describes the magnitude of the coupling of the secondary system $\underline{y}$ to the primary system $\underline{x}$.

Following an analogous argumentation as in previous section 3.2.1, we define a difference variable $\underline{z} = \underline{x} - \underline{y}$ and find the following inequality as analogue to equation 3.22:

$$\|\underline{z}\| \leq C \exp\left((\lambda - \alpha)\, t\right) \tag{3.26}$$

The parameter λ represents the largest Lyapunov exponent of system F. We can again define a critical coupling strength α_c, which when exceeded will cause the secondary system to synchronise to the primary:

$$\alpha_\mathrm{c} = \lambda \tag{3.27}$$

The results of this corollary are visualised in figures 3.7f. Figure 3.7 shows how each secondary path converges towards the primary, driving system leading to synchronisation via a unidirectional coupling. Figure 3.8 shows the rapid decline of the synchronisation error after the coupling α exceeds the critical coupling α_c, indicating that the systems behave identically. Using this coupling, we can synchronise two arbitrary complex systems to each other while maintaining the notion of a driving system and a driven system.

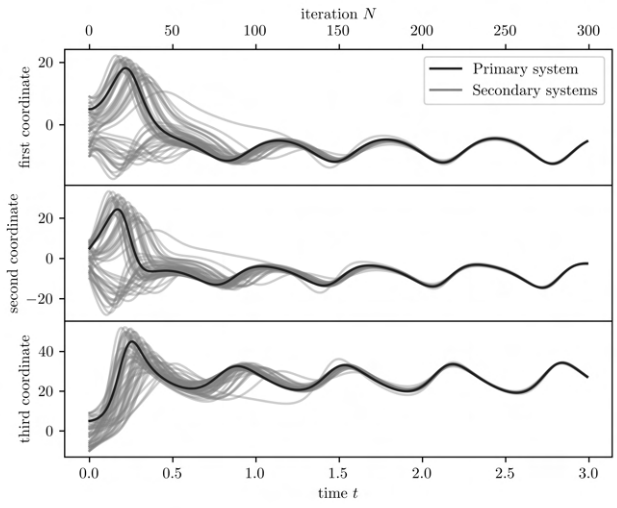
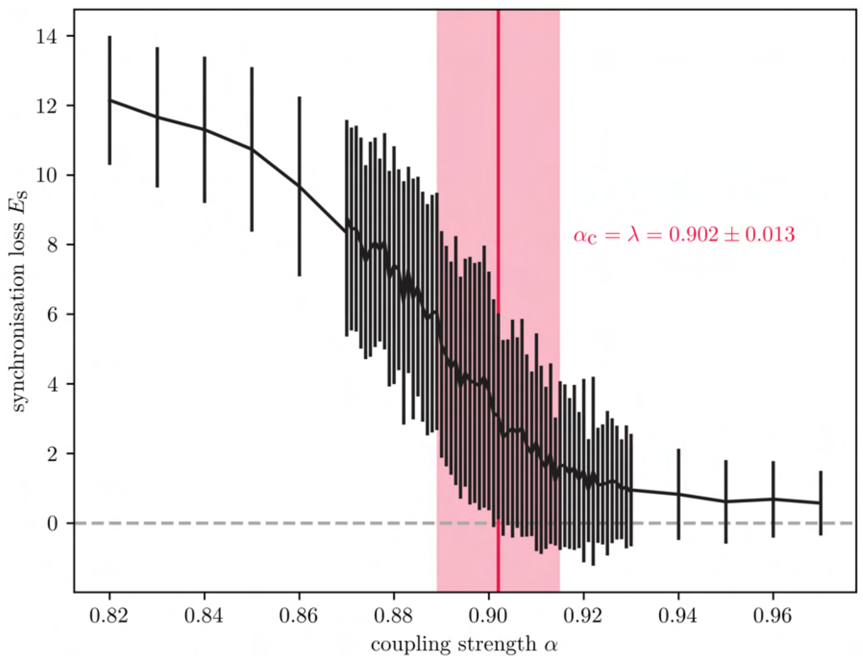
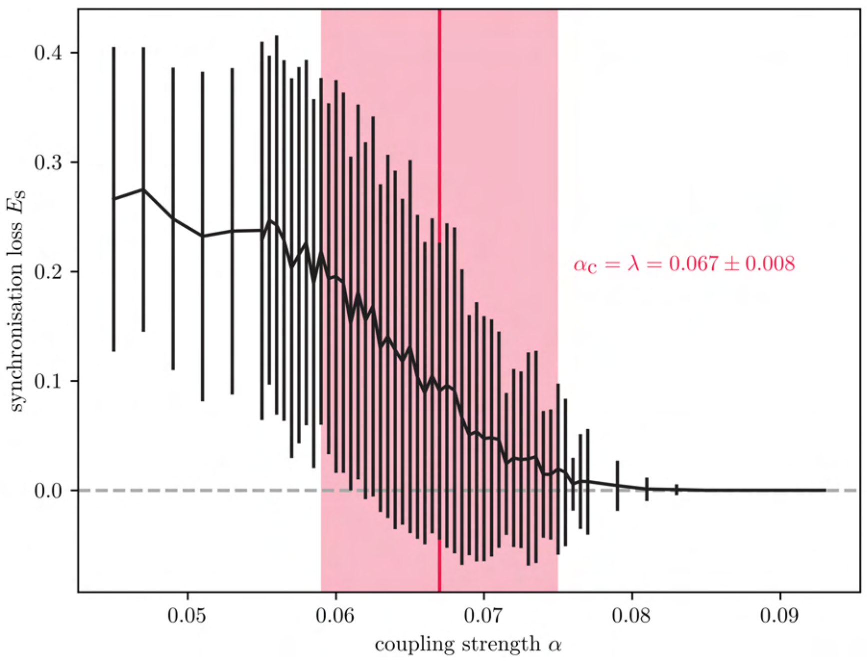


Figure 3.7: This figure visualises the unidirectional synchronisation of numerous secondary systems (each defined by equation 3.10) to a single primary system providing the signal as specified in equation 3.11. We can see clearly that eventually, each secondary system behaves identically to the primary system. For each Lorenz system the standard parameters have been used with a coupling strength $\alpha = 2$ and an integration step of 10^{-2}. The primary system has the initial condition $(5, 5, 5)^\intercal$, whereas for the initial condition of each secondary system three random numbers between -10 and 10 were used. This figure shows fifty secondary systems.



(a) Lorenz system



(b) Rössler system

Figure 3.8: Figure (a) shows the synchronisation loss E_s, as defined in equation 3.24, for the Lorenz system and figure (b) shows it for the Rössler system. The definition of latter system can be found in the appendix A.6. For both systems their respective standard parameters have been used and for each coupling strength α a hundred iterations have been calculated. For each iteration an integration step of 10^{-2} has been used and iterated for a total of 10^6 steps. For the calculation of the error the last 10^5 steps have been used. The initial condition for each Lorenz system were three random numbers between 1 and 20, and for each Rössler system three random numbers between 0 and 1. The red line shows the largest Lyapunov exponent as calculated in table 2.1.

While this coupling method guarantees a unidirectional synchronisation, equation 3.26 has one caveat: due to the constant C it is not possible to estimate a time t, after which the distance between the two systems $\|\underline{z}\|$ is sufficiently small [16]. Meaning that, while we can guarantee synchronisation eventually, we cannot make a statement on how quickly the synchronised state will be reached. However, experimentally, we find that with an increasing coupling strength α we are able to reduce the time until synchronisation. This finding is intuitive when analysing equation 3.26, since the higher the magnitude of the exponent, the quicker the distance $\|\underline{z}\|$ decays. The experimental results are plotted in figure 3.9, where we can clearly observe a reduction in time to synchronisation with increasing coupling strengths for different systems. For our optimisation algorithm, this means that we can fix the coupling length α to a comparably high value, since it will only affect the time to synchronisation. The sooner the systems synchronise, the better is the starting point of our optimisation algorithm, as we will discuss in the following chapter 4.

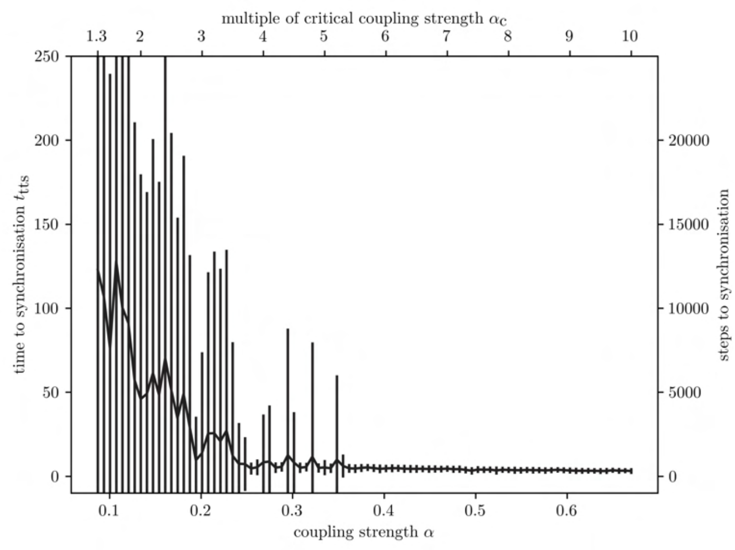


Figure 3.9: This figure visualises the decay time to synchronisation for increasing coupling strengths α for the Rössler system with standard parameters. For this case, we define the "time to synchronisation" as the time t_{tts}, for which the synchronisation error E_{s} of the interval $[t_{\text{tts}},\ t_{\text{tts}} + \tau_\lambda]$ is the first time smaller than 5% of the uncoupled baseline error. Here, τ_λ describes the Lyapunov time for each system. The baseline error was calculated by averaging over a hundred realisations of two uncoupled Rössler systems. For each coupling strength a hundred realisations were used too. For each realisation of the baseline error and coupling strength the primary system had the initial condition $(0.1, 0.2, 0.4)^\intercal$ and for the initial condition of the secondary system three random numbers between zero and one were drawn. In order to ensure synchronisation, the first coupling strength has been chosen as $1.3\,\alpha_{\text{c}}$. We used an integration step of 10^{-2}.

For the sake of completeness, we present the coupling scheme of our corollary in figure 3.10. Unlike as for the bidirectional coupling in previous section 3.2.1 represented with figure 3.5, we can now clearly distinguish the coupling influence stemming from the primary system influencing the secondary system. The secondary system does not influence the evolution of the primary system.

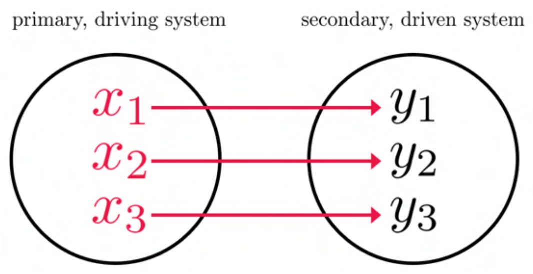


Figure 3.10: This figure shows the coupling scheme between a primary, driving system and a secondary, driven system for the unidirectional external coupling, as described in equations 3.25, using the identity matrix as coupling matrix **H**. This representation describes an arbitrary three-dimensional chaotic system.

Chapter 4

Optimisation Algorithm

Calibrating systems of **o**rdinary **d**ifferential **e**quation**s** (ODEs) to data is an important task when trying to understand causal connections and predicting future states of any system. In this chapter we present an algorithm for calibrating a system of governing equations to an experimentally obtained dataset. Our algorithm can be seen as an extension and generalisation of the algorithm described by [2]. In this article, our algorithm uses the property of synchronisation and a subsequent gradient descent-algorithm for finding the parameters that minimise a loss function. However, we extend the existing framework by introducing two novel aspects to it: firstly, we introduce a way of integrating a system of ODEs coupled to discrete data points — as opposed to integrating systems simultaneously. Secondly, we apply the coupling described in section 3.2.2 for ensuring applicability on arbitrary systems.

The general idea is to treat the discrete data points as a primary system in the sense of chapter 3. Then, we vary the parameters of a synthetic, secondary system until it synchronises to the data points making up the first system. After successful synchronisation, we can be sure that the parameters found in the secondary system are the ones describing the primary one. Experimentally we find, that the coupling used in the secondary system ensures a smooth loss surface, on which gradient descent-algorithms of different flavours can be applied. Here, the necessity of modifying the bidirectional external coupling to a unidirectional one becomes obvious, since we cannot modify the data points, making up the primary system, to induce synchronisation.

In this chapter we will describe our proposed optimisation algorithm detailedly focusing on the new features. In section 4.1 we will illustrate the way we integrate ODEs coupled to data. Afterwards in section 4.2, we will utilise the integration method and explain the coupling of the systems and the calculation of the gradient thereof. Lastly in section 4.3, we will present different flavours of gradient descent-algorithms and analyse their performance on this problem. Section 4.4 sums the algorithm up and presents it in a concise way.

4.1 Integration of Ordinary Differential Equations

Integrating ordinary differential equations is a well known problem with an established standard solution: the Runge–Kutta–Fehlberg method [21], [22], [23]. In this section we will first sketch the standard integration technique and then explain how we modified it for our needs.

4.1.1 Runge–Kutta–Fehlberg Integration

The Runge–Kutta–Fehlberg method is designed to solve first order ordinary differential equations of the following form [24]:

$$\dot{\underline{y}} = F\big(t, \underline{y}\big) \tag{4.1}$$

Here, $\dot{\underline{y}}$ is the derivative with respect to time t of the state $\underline{y}$ and therefore F: $\mathbb{R}^N \to \mathbb{R}^N$ describes the time-evolution of said system. We note that this method is also applicable to any

N-dimensional system F.

In order to get an intuition for the Runge–Kutta–Fehlberg method, we first introduce two simpler methods. A first approach would be to evaluate the system piecewise using a small step size of Δt, leading to the following integration rule: $\underline{y}(t+\Delta t) = \underline{y}(t) + \Delta t\,\dot{\underline{y}}(t) = \underline{y}(t) + \Delta t\,F\big(t,\,\underline{y}(t)\big)$. This is known as the "Euler method" of integration [25]. An important metric of integration schemes is the **l**ocal **t**runcation **e**rror (LTE). The LTE describes an upper bound of the error between the exact solution and the integrated one after each single integration step. Using Taylor-expansion, we can show that the local truncation error of the Euler method is of order $(\Delta t)^2$ [25]. For most applications this error is too big and more sophisticated integration methods are required.

In their works the authors developed the so-called set of "Runge–Kutta methods" [21], [22]. The main idea of those is to consider a specifically weighted average of specifically chosen, intermediate steps between two time steps. Those intermediate steps and their weighting stem from utilising the Taylor-expansion. This method reduces the error drastically. The fourth order variant of the Runge–Kutta method is commonly used and it has a local truncation error of the order $(\Delta t)^5$. Given a system in the setting of equation 4.1, the integration rule is defined the following way [25]:

$$\underline{y}(t+\Delta t) = \underline{y}(t) + \frac{1}{6}\,\Delta t\;(\underline{k}_1 + 2\underline{k}_2 + 2\underline{k}_3 + \underline{k}_4) \tag{4.2}$$

The intermediate steps $\underline{k}_i$ are defined as follows:

$$\begin{aligned}
\underline{k}_1 &= F\big(t,\,\underline{y}(t)\big) \\
\underline{k}_2 &= F\left(t + \frac{\Delta t}{2},\,\underline{y}(t) + \Delta t\,\frac{\underline{k}_1}{2}\right) \\
\underline{k}_3 &= F\left(t + \frac{\Delta t}{2},\,\underline{y}(t) + \Delta t\,\frac{\underline{k}_2}{2}\right) \\
\underline{k}_4 &= F\big(t + \Delta t,\,\underline{y}(t) + \Delta t\,\underline{k}_3\big)
\end{aligned} \tag{4.3}$$

So far, the step size Δt has been constant for the Euler method and the Runge–Kutta method. In his work, Fehlberg extended this framework and introduced an adaptive step size in order to further decrease the truncation error. The basic idea is to estimate the truncation error at each integration step and, depending on its size, to halve the integration step or to keep it at its current value [23]. The truncation error is estimated by calculating two different orders of the Runga–Kutta method. A modern implementation can be found in [26] and the remaining section follows the findings thereof. Consider the following intermediary steps:

$$\begin{aligned}
\underline{k}_1 &= F\big(t,\,\underline{y}(t)\big) \\
\underline{k}_2 &= F\left(t + \frac{1}{4}\Delta t,\,\underline{y}(t) + \frac{1}{4}\underline{k}_1\right) \\
\underline{k}_3 &= F\left(t + \frac{3}{8}\Delta t,\,\underline{y}(t) + \frac{3}{32}\underline{k}_1 + \frac{9}{32}\underline{k}_2\right) \\
\underline{k}_4 &= F\left(t + \frac{12}{13}\Delta t,\,\underline{y}(t) + \frac{1932}{2197}\underline{k}_1 - \frac{7200}{2197}\underline{k}_2 + \frac{7296}{2197}\underline{k}_3\right) \\
\underline{k}_5 &= F\left(t + \Delta t,\,\underline{y}(t) + \frac{439}{216}\underline{k}_1 - 8\underline{k}_2 + \frac{3680}{513}\underline{k}_3 - \frac{845}{4104}\underline{k}_4\right) \\
\underline{k}_6 &= F\left(t + \frac{1}{2}\Delta t,\,\underline{y}(t) - \frac{8}{27}\underline{k}_1 + 2\underline{k}_2 - \frac{3544}{2565}\underline{k}_3 + \frac{1859}{4104}\underline{k}_4 - \frac{11}{40}\underline{k}_5\right)
\end{aligned} \tag{4.4}$$

Again, the weighting of each coefficient in the arguments of the intermediate steps can be derived using the Taylor-expansion. We use those intermediate steps to calculate two estimations of different orders [26]:

$$\underline{y}(t+\Delta t) = \underline{y}(t) + \Delta t \left(\frac{25}{216}\underline{k}_1 + \frac{1408}{2565}\underline{k}_3 + \frac{2197}{4104}\underline{k}_4 - \frac{1}{5}\underline{k}_5 \right) \tag{4.5a}$$

$$\underline{z}(t+\Delta t) = \underline{y}(t) + \Delta t \left(\frac{16}{135}\underline{k}_1 + \frac{6656}{12825}\underline{k}_3 + \frac{28561}{56430}\underline{k}_4 - \frac{9}{50}\underline{k}_5 + \frac{2}{55}\underline{k}_6 \right) \tag{4.5b}$$

Equation 4.5a describes the fourth order Runge–Kutta step and equation 4.5b describes the fifth order integration step. The difference between the two numerical estimates provides a convenient estimation of the local truncation error: $\varepsilon_{\mathrm{LTE}} = \left|\underline{y}(t+\Delta t) - \underline{z}(t+\Delta t)\right|$. If the error exceeds a defined threshold ε, the integration step Δt can be reassigned to the following value for that single step, which will ensure the new error being below the desired threshold [26]:

$$\Delta t \leftarrow \Delta t \left(\frac{\varepsilon}{2\,\varepsilon_{\mathrm{LTE}}} \right)^{\frac{1}{4}} \tag{4.6}$$

In this work, this integration method was the default method when integrating systems, for which a continuous functional description existed: meaning that each system F could be evaluated at any arbitrary time t. In the cases where this was not possible — because the primary system consisted of discrete data points — the integration method presented in section 4.1.2 was used.

Integration of Coupled ODEs

For the calculations in chapter 3 we needed to integrate coupled systems. The systems typically consisted of a primary system $\{\underline{x}\}$ and a secondary system $\{\underline{y}\}$. Here the curly brackets describe the set of all states. Using the coupling scheme of section 3.2.2, a primary system is described by an autonomous function $F\colon \mathbb{R}^N \to \mathbb{R}^N$ and a secondary system is described by a non-autonomous function $G\colon \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}^N$. We can represent this as follows:

$$\dot{\underline{x}} = F(t,\, \underline{x}) \in \mathbb{R}^N \tag{4.7a}$$

$$\dot{\underline{y}} = G\big(t,\, \underline{y},\, \underline{x}\big) \in \mathbb{R}^N \tag{4.7b}$$

In order to integrate the systems simultaneously, we defined a super-system combining both systems resulting in the following:

$$\underline{\varphi} = \begin{pmatrix} \underline{x} \\ \underline{y} \end{pmatrix} \in \mathbb{R}^{2N} \tag{4.8a}$$

$$\dot{\underline{\varphi}} = \begin{pmatrix} \dot{\underline{x}} \\ \dot{\underline{y}} \end{pmatrix} = \begin{pmatrix} F(t,\, \underline{x}) \\ G\big(t,\, \underline{y},\, \underline{x}\big) \end{pmatrix} \in \mathbb{R}^{2N} \tag{4.8b}$$

After evaluating the system $\{\underline{\varphi}\}$, we can split it back to the systems $\{\underline{x}\}$ and $\{\underline{y}\}$. This method assumes however, that we can evaluate arbitrary states of the primary system. Since we want to develop our algorithm to work on a discrete set of data, this assumption cannot hold. It is not possible to evaluate a given set of data at points which the set does not contain. Therefore, we need a new method of integrating coupled ODEs, in which the primary system is described by discrete data. Such a method is described in the following section.

4.1.2 Integration of ODEs Coupled to Data

In this section we present our scheme for evaluating a secondary system coupled to a primary system consisting of discrete data points, considering the definitions of section 3.2.2. The main idea is to keep the primary data constant in each integration step of the secondary system. Meaning, when the Runge–Kutta step requires a function evaluation of the primary system at time where no value is available, we approximate that value with value of the beginning of that interval. This can be viewed as using the data points of the primary system as additional parameters, which change at each integration step. Using this method has two implications: firstly, we need to know the step size, at which the data constituting the primary system was sampled from. Secondly, we cannot use the Runge–Kutta–Fehlberg method, since a variable step size is not sensible any more. We are bounded by the step size the data describing the primary system provides and that defines the maximum precision we can obtain. For each single integration step we here use the unmodified Runga–Kutta method in fourth order.

Concretely this means, given a set of data stemming a primary system F with parameters $\underline{\theta}$, we need to define a secondary system G the following way:

$$\dot{\underline{y}} = G\big(\underline{y};\, \underline{\Theta}\big) = F\big(\underline{y};\, \underline{\theta}\big) + \mathbf{1}\,\alpha\,\big(\underline{\chi} - \underline{y}\big) \tag{4.9}$$

Here, capital $\underline{\Theta}$ holds the parameters $\underline{\theta}$ of the system F and the state of the primary system $\underline{\chi}$. Note that we changed the notation for the primary system from $\underline{x}$ to $\underline{\chi}$ for emphasising that the data of the primary system is treated as a constant in that specific step. The values of $\underline{\chi}$ change with every increment of time. The parameter α describes the coupling strength between the secondary system and the data, primary system. It has to be larger than the Lyapunov exponent of the system F. The algorithm 4.1 emphasises the use of the data in this scheme and the constant need for updating the parameters $\underline{\Theta}$.

```
input: A discrete realisation of the primary system, evaluated with parameters $\underline{\theta}$ and a
       step size of $\Delta t$, labelled ‘data’
$l \leftarrow$ length(data)
$\underline{y} \leftarrow$ initialise empty list($l$)
$\underline{y}[0] \leftarrow$ initial state
for $i \leftarrow 1 \dots l$ do
    $\underline{\Theta} \leftarrow \begin{pmatrix} \underline{\theta} \\ \text{data}[i-1] \end{pmatrix}$
    $\underline{y}[i] \leftarrow$ Runge-Kutta step$_G$ $\big(\underline{y}[i-1],\, \underline{\Theta},\, \Delta t\big)$
end
return $\underline{y}$
```

Algorithm 4.1: This algorithm describes the evolution of a secondary system $\underline{y}$ coupled to a primary system consisting of discrete data points. The function `Runge-Kutta step`$_G$ performs a single iteration of the fourth order Runge–Kutta method of the system of ODEs G defined in equation 4.9 with the step size Δt. The coupling strength α of aforementioned equation shall be encoded in the function description.

The effectiveness of the algorithm 4.1 is visualised in figure 4.1. There we are able to observe synchronisation for different initial conditions, effectively recreating the figure 3.7 in section 3.2.2 — with the main distinction being that the secondary systems are not coupled to a primary system F. Instead, they are coupled to discrete data points. Nevertheless, they synchronise to a given trajectory. This justifies our use of this integration technique for our optimisation algorithm.

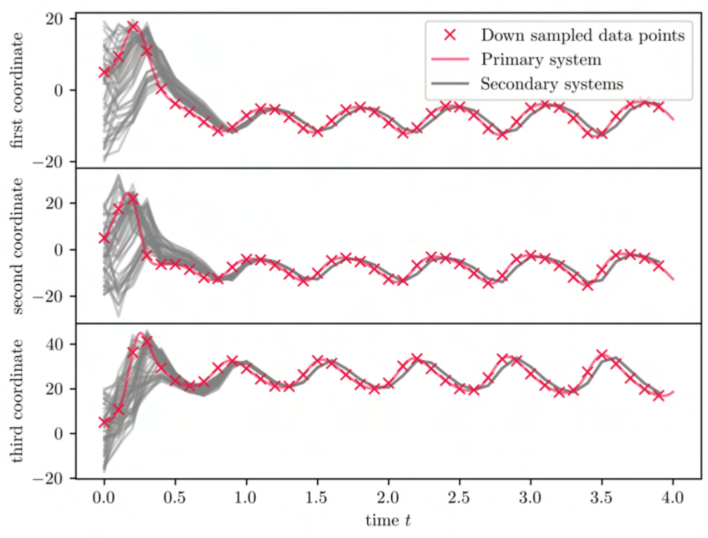


Figure 4.1: This figure visualises successful synchronisation of secondary paths to a primary system consisting of data. As an example the Lorenz system with the standard parameters has been used. The primary system, shown in red, was evaluated using a time step of 10^{-4} with an initial condition of $(5, 5, 5)^\intercal$. From that, every thousandth point was sampled resulting in the red crosses. Each grey line has a random number from -20 to 20 for each coordinate as initial condition and was integrated using the algorithm in 4.1 and hence coupled to the red crosses. It is important to emphasise that the grey lines have been coupled to the discrete, red points: the red line is solely a visual aid. This figure shows fifty realisations of the secondary system, and we can see synchronisation in each. Note, that the secondary systems do not show a smooth behaviour due to the effective step size of 0.1. A coupling strength of $\alpha = 5$ has been used.

4.2 Loss Function and Gradient

The main part of this optimisation algorithm is the loss function we are trying to minimise. In this section we will first present the loss function and show its properties. Afterwards, we calculate the gradient for performing steps to minimise the loss function.

4.2.1 The Loss Function

Following the article [2], we use a loss function $\mathcal{L}$ which describes the distance between the evolution of a primary system and the evolution of the secondary system. This means that the loss function is *not* dependent on the absolute states $\{\underline{x}\}$ and $\{\underline{y}\}$ (primary and secondary system respectively). Instead, it is dependent on their derivatives $\{\dot{\underline{x}}\}$ and $\{\dot{\underline{y}}\}$. The reason for this is as follows: in our algorithm we try to vary the parameters of a system in order to minimise a loss function. The states of each system are not explicitly dependent on the parameters. Rather, they depend on the integration method used. So by utilising the derivative, we cut back on an intermediary step when applying the chain rule. Additionally by construction, we know the the initial state of the secondary system, since it is the first data point available to us. If two systems have the same initial condition and the same derivative, they will evolve equally; being another argument for not requiring the absolute states of the system. We conclude that the loss function $\mathcal{L}$ is a function dependent on the primary and secondary system of the form $\mathcal{L}(\dot{\underline{x}}, \dot{\underline{y}})$. In general, the loss function can be any norm in the mathematical sense. In practice, we find the mean squared error a useful error measure.

For our algorithm we use a fixed number of time steps to perform the optimisation on. This fixed number of steps is denoted "evaluation length" l_e in this work. We calculate the loss as sum of the mean squared error over each step in the evaluation length. The loss for each single time step i is ℓ_i, so we obtain the complete loss $\mathcal{L}$ by summing over each loss of each individual step:

$$\ell_i = \left(\underline{\dot{x}}_i - \underline{\dot{y}}_i\right)^2 \tag{4.10a}$$

$$\mathcal{L}\big(\{\underline{\dot{x}}\}, \{\underline{\dot{y}}\}\big) = \frac{1}{l_\mathrm{e}} \sum_{i=1}^{l_\mathrm{e}} \ell_i \tag{4.10b}$$

Meaning that we add together the loss of each individual sample i in the set of all available derivatives $\{\underline{\dot{x}}\}$ and $\{\underline{\dot{y}}\}$. We note that the loss function has the secondary, coupled system in it, meaning the term of equation 4.9 with the coupling α is represented in $\{\underline{\dot{y}}\}$.

Experimentally, we noted that using the coupling we obtain a smooth loss surface, as visualised in figure 4.2. It is impossible to try to run an optimisation for finding the minimum on the surface in 4.2a, since a clear direction towards the minimum of the loss, represented by the black line, cannot be found. However, the loss surface 4.2b shows a convex shape meaning that the application of gradient-descent algorithms is justified and promising. We therefore can conclude that applying the coupling as discussed in section 3.2.2 smooths the loss surface making classical gradient-descent algorithms applicable.

4.2.2 Calculation of the Gradient

As we have shown in figure 4.2, utilising the external coupling presented in section 3.2.2 transforms a non-convex loss surface to a convex one. Therefore, calculating the gradient of the loss function with respect to the parameters $\underline{\theta}$ and making steps in the negative direction of the gradient will lead to those parameters minimising the loss function, and hence describe the primary system. Since our algorithm requires a primary system based on data, we need to calculate the numerical derivative of our measured states $\{\underline{\chi}\}$ — and thus exchanging $\underline{\dot{x}}$ with $\underline{\dot{\chi}}$ in equation 4.10a.

The loss function ℓ is not directly dependant from the parameters $\underline{\theta}$, instead the evolution of the secondary system $\underline{\dot{y}}$ is. Therefore, we need to apply the chain rule when calculating the the derivative of the loss function with respect a spefic parameter θ in the parameter vector $\underline{\theta}$. Additionally, equation 4.10a implies a sum over each coordinate n of an N-dimensional system, since we are dealing with vector-valued systems. So generally, the following equation for j-th entry g_j of the gradient $\underline{g}$ holds, where we calculate the derivative of equation 4.10a:

$$g_j = \frac{\partial \mathcal{L}}{\partial \theta_j} = \frac{1}{l_\mathrm{e}} \sum_{i=1}^{l_\mathrm{e}} \frac{\partial \ell_i}{\partial \theta_j} \tag{4.11a}$$

$$\frac{\partial \ell}{\partial \theta_j} = -2 \sum_{n=1}^{N} \left(\dot{\chi}_n - \dot{y}_n\right) \frac{\partial \dot{y}_n}{\partial \theta_j} \tag{4.11b}$$

For the sake of readability we ommited the index i in equation 4.11b. It is understood however, that equation 4.11b shows the loss term of a single sample i. Note also, that if a particular dimension n is not dependent on the parameter θ_i, that term in the sum over the coordinates will be 0.

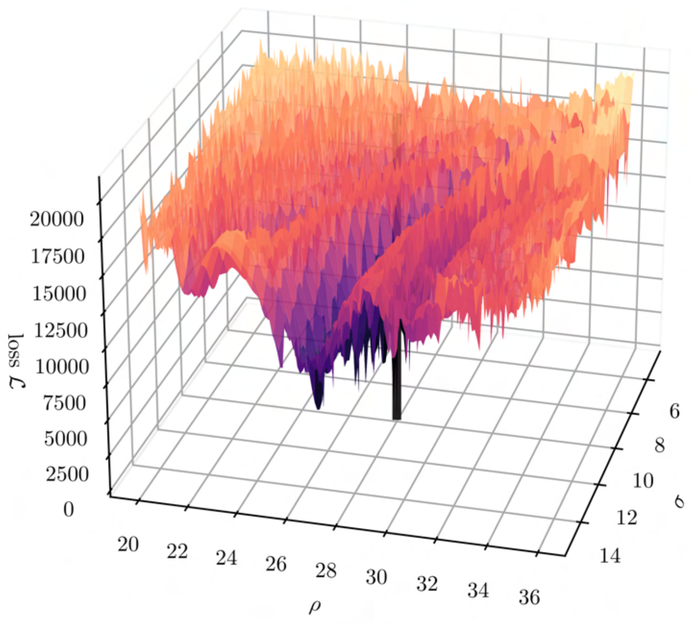
Gradient of the Lorenz system

In the following we perform the calculation of the gradient for the Lorenz system. The Lorenz system has three parameters $\underline{\theta} = (\sigma, \rho, \beta)^\intercal$ and is a three-dimensional system. Calculating the gradient in equation 4.11b for each parameter leads to the following result:

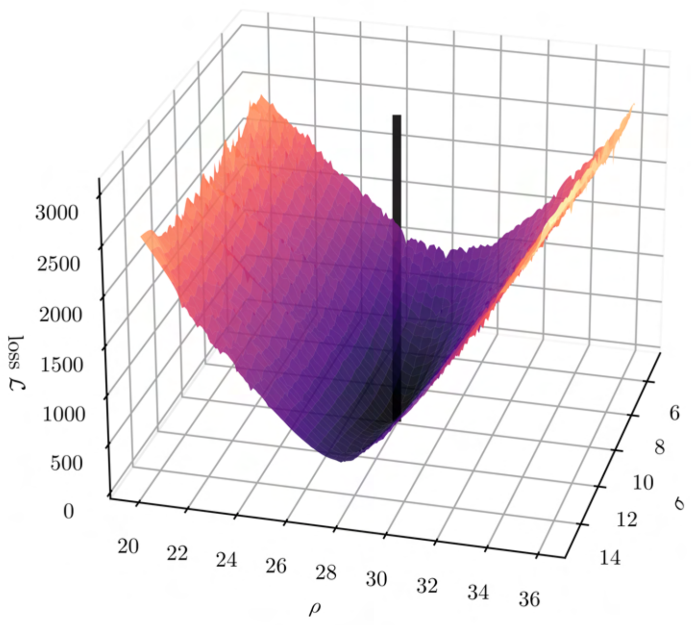
$$\frac{\partial \ell}{\partial \sigma} = -2\left(\dot{\chi}_1 - \dot{y}_1\right)\left(y_2 - y_1\right) \tag{4.12a}$$

$$\frac{\partial \ell}{\partial \rho} = -2\left(\dot{\chi}_2 - \dot{y}_2\right) y_1 \tag{4.12b}$$

$$\frac{\partial \ell}{\partial \beta} = +2\left(\dot{\chi}_3 - \dot{y}_3\right) y_3 \tag{4.12c}$$



(a) Loss for uncoupled systems



(b) Loss for coupled systems

Figure 4.2: This figure shows the loss $\mathcal{L}$ as defined in equations 4.10 for the reduced Lorenz system, where the parameter β has been kept constant at $\frac{8}{3}$. The primary system is the Lorenz system with the standard parameters. The secondary system is the Lorenz system with the parameters from the grid. Figure (a) shows the loss for the uncoupled systems. Figure (b) shows the loss for the systems, whereby the secondary system is coupled to the primary system via the unidirectional coupling from section 3.2.2 with a coupling strength of $\alpha = 10$. Each Lorenz system was integrated with an initial state of $(5,\ 5,\ 5)^\intercal$ and a step size of $\Delta t = 10^{-2}$. The losses were calculated over an evaluation length of $l_\mathrm{e} = 10^3$. The black line marks the true parameters of the primary system.

4.3 Update Step

Once we calculated the gradient, we need to update the system parameters in the direction minimising the loss function. The most basic approach is utilising the "gradient descent" algorithm, where we subtract the direction with the negative slope from the parameters:

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \underline{\eta} \odot \underline{g}_k \tag{4.13}$$

Here, $\odot$ symbolises the Hadamard product between two vectors. The index indicates the k-th step of the optimisation, and we perform the optimisation until the all the parameters have converged. The vector $\underline{\eta}$ describes the learning rate, meaning the size of the update step we take after each optimisation step k. Since complex systems can be more sensitive on certain parameters than on others, we choose a different learning rate for each parameter. Heuristically, we discovered that setting the learning for each parameter in such a way, that the change of the first update step is of order 10^{-1}, leads to convergence for a variety of systems:

$$\eta_1^j \, g_1^j \sim \mathcal{O}\big(10^{-1}\big) \tag{4.14}$$

The subscript 1 emphasises that we are considering the first optimisation step at $k = 1$. The superscript j indicates that this relationship holds for each single parameter. For each subsequent optimisation step we keep the learning rate constant to this value.

This optimisation works well, however with the rise of machine learning, more sophisticated first-order optimisers have been developed. In the following we will present two all-purpose optimisers and analyse their performance against the plain gradient descent-method.

Adam Optimiser

In this section we present the "Adam" optimiser proposed in [7] and we follow their findings hereafter. According to the authors, the name "Adam" is derived from "adaptive moment estimation", which describes the main idea of their optimiser: they compute individual learning rates utilising first and second moment estimations of the gradient. Those estimates are obtained by exponentially weighting past gradients (first moment) and past squared gradients (second moment). The first moment is captured in the vector $\underline{m}$ and the second moment is captured in the vector $\underline{v}$. With $\underline{g}_k$ being the gradient at optimisation step k, the moment estimations are updated as follows [7]:

$$\underline{m}_k = \beta_1 \, \underline{m}_{k-1} + (1 - \beta_1) \, \underline{g}_k \tag{4.15}$$

$$\underline{v}_k = \beta_2 \, \underline{v}_{k-1} + (1 - \beta_2) \left(\underline{g}_k \odot \underline{g}_k \right) \tag{4.16}$$

Both vectors, $\underline{m}$ and $\underline{v}$, are initialised with $\underline{0}$. The authors noted that by doing so, the estimates are biased towards $\underline{0}$. For that reason, they propose the following bias-corrected estimate:

$$\hat{\underline{m}}_k = \frac{1}{1 - \beta_1^k} \, \underline{m}_k \tag{4.17}$$

$$\hat{\underline{v}}_k = \frac{1}{1 - \beta_2^k} \, \underline{v}_k \tag{4.18}$$

Here the superscript k actually represents taking the value to the k-th power. This results in the following update step:

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \underline{\eta} \odot \hat{\underline{m}}_k \oslash \left(\sqrt{\hat{\underline{v}}_k} + \varepsilon \right) \tag{4.19}$$

Here, $\oslash$ symbolises the element-wise division and the square-root is understood to be taken over each element of $\hat{\underline{v}}_k$. The step size for each parameter $\underline{\eta}$ has been estimated using the method discussed in section 4.3. The authors recommend the following values for the remaining free parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-8}$. As can be seen in figure 4.3, this optimisation works and leads to convergence of the parameters.

AMSGrad Optimiser

In their article the authors found scenarios in which the Adam optimiser does not converge to the optimal solution [8]. They show that this is based on considering the moving average of past squared gradients — one of the key features of Adam. The issue is, that under certain circumstances the Adam optimiser may take steps too large [8]. This happens if the second moment estimate $\hat{v}_k^j$ for a parameter j at step k gets too small. They mitigated this issue by considering the element-wise maximum value of the new estimate and the previous estimate. This way they ensure that the step size will not increase for each parameter:

$$\hat{v}_k^j = \max\left(\hat{v}_{k-1}^j,\, v_k^j\right) \qquad \forall j \in \{1, ..., \dim \underline{\theta}\} \tag{4.20}$$

Additionally, they simplify the optimiser by discarding the bias correction terms. Therefore, the complete update step can be calculated by:

$$\underline{m}_k = \beta_1\, \underline{m}_{k-1} + (1-\beta_1)\, \underline{g}_k \tag{4.21a}$$

$$\underline{v}_k = \beta_2\, \underline{v}_{k-1} + (1-\beta_2)\, \left(\underline{g}_k \odot \underline{g}_k\right) \tag{4.21b}$$

$$\underline{\hat{v}}_k = \max\left(\underline{\hat{v}}_{k-1},\, \underline{v}_k\right) \tag{4.21c}$$

$$\underline{\theta}_k = \underline{\theta}_{k-1} - \underline{\eta} \odot \underline{m}_k \oslash \sqrt{\underline{\hat{v}}_k} \tag{4.21d}$$

The maximum function in equation 4.21c is applied element-wise for each variance estimation of each parameter. In their article the authors mainly use the following values for the coefficients: $\beta_1 = 0.99$, $\beta_2 = 0.999$. Unlike the original article, we use a different reference learning rate $\underline{\eta}$ for each parameter as described in section 4.3.

Comparison

In this section we briefly compare the different optimisation methods and showcase their concrete usage in figure 4.3. There we can see that all optimisation algorithms eventually converge to the right solution. However, the sophisticated optimisers converge faster than plain gradient descent. Additionally, we note that the AMSGrad optimiser oscillates around the true parameters while the Adam optimiser does not show any. We did not note any negative effect on the optimisation from the oscillations. In the following we will use the AMSGrad optimiser, since it converges in more cases than the Adam optimiser [8].

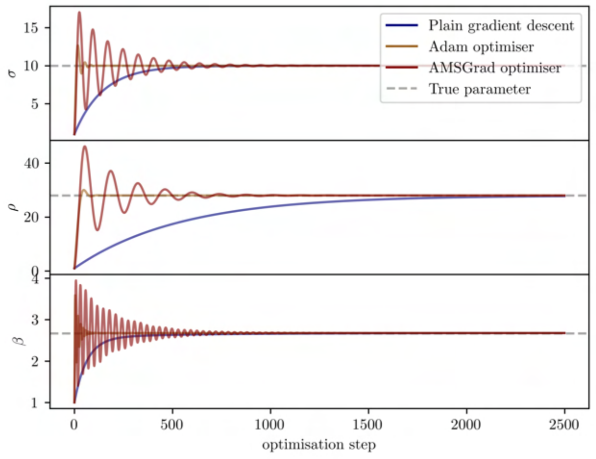


Figure 4.3: This figure visualises the convergence of the three different optimisation algorithms presented in this work. For these experiments we tried to fit the parameters of a Lorenz system. The true system was created using the standard parameters $\underline{\theta} = (\sigma,\, \rho,\, \beta)^\intercal = \left(10,\, 28,\, \frac{8}{3}\right)^\intercal$, a time step of $\Delta t = 10^{-3}$, and an initial state of $(5,\, 5,\, 5)^\intercal$. The optimisation was performed as described in algorithm 4.2 with the respective optimisation step. For an equal comparison, the hyperparameters for each optimisation algorithm were kept constant and had following values: we used an evaluation length of $l_\mathrm{e} = 10^3$ and a coupling strength of $\alpha = 10^3$.

4.4 Summary

In this section we want to present a concise summary of our optimisation algorithm for finding the correct parameters of complex systems using data. The complete algorithm is found in algorithm 4.2. This algorithm solves a specific use case: algorithms exist, which are capable of reconstructing governing equations, in the sense of ODEs, out of data. However, some algorithms can only reconstruct the equations parameterless meaning that the coefficients are not calibrated (concretely, they are given a value of one). We developed our algorithm specifically for this case. We are given data stemming from a system ODEs, and we know those ODEs even though they have the wrong, uncalibrated parameters. By synchronising the secondary system to the given data, we are able to find the original parameters.

In chapter 5 we will apply this algorithm to an ample selection of synthetic systems and we will show that we are able to reconstruct the correct set of parameters for each of them. Additionally, this algorithm can work as an error correction to some degree: we will show, that it is possible to cancel out wrong terms in the systems of ODEs. This means that if the data stems from a set of ODEs, but our initial believe of that set is a superset of the actual ODEs, we are able to cancel out the wrong terms which are non-existent in the data. We will give an example for this in section 5.2.2.

```
input: This optimisation algorithm needs three external inputs: first, the 'data', a
       discrete realisation of the primary system, stemming from the ODE in question
       with parameters $\underline{\theta}$ and a step size of $\Delta t$. Secondly, we need a coupling strength
       $\alpha$ between the data and the system. Lastly, we need the evaluation length $l_\mathrm{e}$.
true data ← data[: $l_\mathrm{e}$]
true derivative ← derive numerically(true data)
$\underline{\theta}$ ← initialise list of ones($n_\mathrm{p}$)
$\underline{m}$, $\underline{v}$, $\underline{\hat{v}}$ ← initialise empty list($n_\mathrm{p}$)
$\underline{\eta}$ ← find learning rate()
$\bar{c}$ ← 1
while $\bar{c}$ do
    estimated system ← data-based Runge-Kutta step$_G$ (true system, $\underline{\theta}$, $\Delta t$, $\alpha$)
    estimated derivative ← derive numerically(estimated data)
    $\underline{g}$ ← calculate gradient(true derivative, estimated derivative)
    $\underline{\theta}$, $\underline{m}$, $\underline{v}$, $\underline{\hat{v}}$ ← AMSGrad update step($\underline{g}$, $\underline{\eta}$, $\underline{m}$, $\underline{v}$, $\underline{\hat{v}}$)
    $\bar{c}$ ← check for convergence($\underline{\theta}$)
end
return $\underline{\theta}$
```

Algorithm 4.2: This algorithm gives a high-level overview over our optimisation algorithm utilising the AMSGrad optimiser. The function `find learning rate` chooses the learning rate $\underline{\eta}$ in such a way that it satisfies equation 4.14. The function `data-based Runge-Kutta step`$_G$ integrates the estimated system coupled to the first one with coupling strength α as described in algorithm 4.1 in section 4.1.2. Again the subscript G implies that our knowledge about the parameterless ODE, on which we try to fit the data on, is embodied in this function. The function `calculate gradient` calculates the gradient as described in equations 4.11. The variable $\bar{c}$ holds the information whether the problem is not converged: if all the parameters converged, $\bar{c}$ will be 0.

Chapter 5

Results and Applications to Synthetic Systems

We applied our new algorithm on an ample selection of different, synthetic, complex systems and found promising results in each of them. In this section we will present our findings for those synthetic systems and explore the limitations of our algorithm. Additionally, we will study the effect of different choices for the hyperparameters.

For assessing the quality of our optimisation, we monitor two metrics: firstly, we calculate the predictable time t_p. This time describes the time after which the prediction differs substantially from the true time series. Since the absolute predictable time is highly correlated to the largest Lyapunov exponent λ of the system, we measure it in units of the Lyapunov time τ_λ for ensuring comparability among different systems:

$$\tau_\mathrm{p} = \frac{t_\mathrm{p}}{\tau_\lambda} = t_\mathrm{p}\,\lambda \tag{5.1}$$

For our work, we define a prediction as wrong when it differs substantially from the true time series and follows a different path on the attractor. Small deviations of the prediction from the true time series are acceptable as long as they do not persist. For that reason, we define the predictable time, as that time when the error in the interval $[t_\mathrm{p},\ t_\mathrm{p} + \frac{1}{10}\,\tau_\lambda]$ is for the first time bigger than 15% of the scale s of the system. The mathematical formulation of the prediction time can be found in the following equation:

$$s = \frac{1}{\tau_\lambda}\int_0^{\tau_\lambda} \|\underline{x}(t)\|\,\mathrm{d}t \tag{5.2}$$

$$\int_{t_\mathrm{p}}^{t_\mathrm{p}+\frac{1}{10}\,\tau_\lambda} \|\underline{x}(t) - \hat{\underline{x}}(t)\|\,\mathrm{d}t > 15\%\,s \tag{5.3}$$

Here, $\underline{x}$ describes the true time series and $\hat{\underline{x}}$ describes the predicted time series. By using this metric, we can compare different systems with each other as this metric relies solely on system properties and uses the characteristic time and size of each system. The fractions in equation 5.3 were chosen heuristically to reflect the requirement to allow small temporary deviations, but trigger at bigger deviations over a period of time.

The second metric we monitor is the mean absolute error E_θ of the fitted parameters $\hat{\underline{\theta}}$ with respect to the real parameters $\underline{\theta}$:

$$E_\theta = \frac{1}{\dim \underline{\theta}} \sum_{i=1}^{\dim \underline{\theta}} \left|\theta_i - \hat{\theta}_i\right| \tag{5.4}$$

This metric allows us to understand the numerical precision of our optimisation.

In order to analyse the general applicability of our proposed algorithm, we apply it on synthetic systems. The results are shown in figure 5.1, where we can clearly see that the parameter reconstruction is precise to the first digit. Using this precision we are able to predict the systems a handful of Lyapunov times ahead. Solely the “Four wing system” is the one odd out in this scenario, since despite a good parameter reconstruction, we cannot predict that system for a meaningful amount of time. Additionally, we observe due to the proximity of points for one system in this figure, that our optimisation algorithm is stable against the initial condition and hence the position on the attractor.

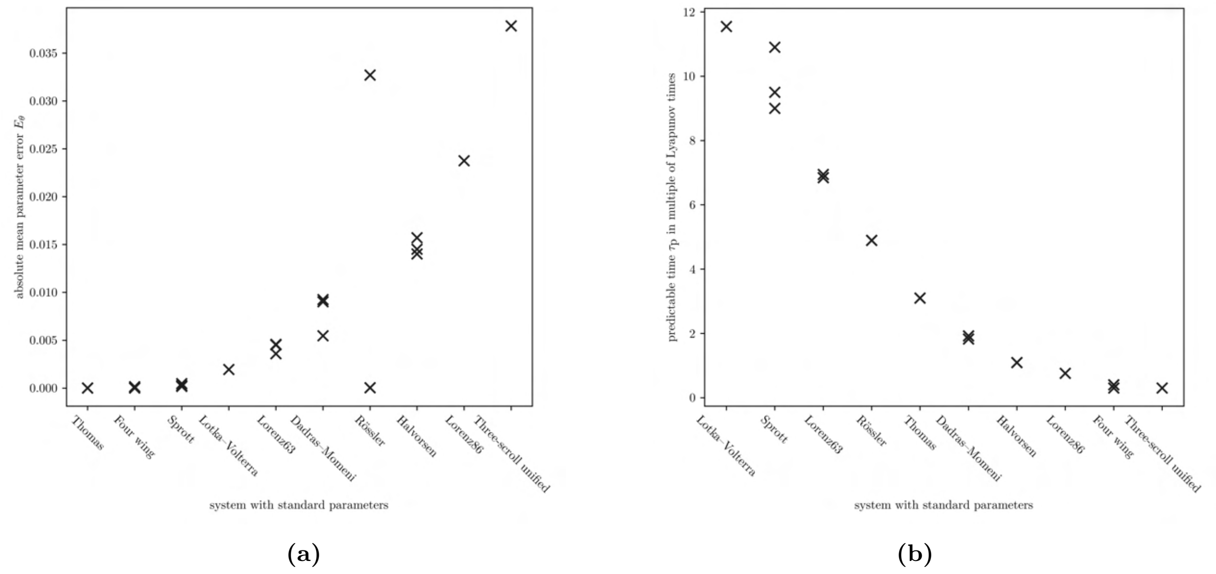


Figure 5.1: This figure shows the quality of the parameter reconstruction for a selection of different systems. Here, figure (a) shows the parameter error E_θ as defined in equation 5.4 and figure (b) shows the predictable time as defined in equation 5.3 in units of the Lyapunov time. Each system, besides the “Lorenz63 system”, and its respective standard parameters $\underline{\theta}$ are defined in the appendix A. For most systems the optimisation has been performed multiple times with a different initial condition for each run. The initial condition has been chosen as a random point on the attractor. For each run and each system an integration step of $\Delta t = 10^{-3}$, an evaluation length of $l_\text{e} = 10^4$, and a coupling strength of $\alpha = 10^3$ have been used.

Lorenz96 System

So far, we only applied our optimisation algorithm on three-dimensional systems with a handful of parameters. In the following we want study the applicability of this optimisation on high dimensional problems. For this purpose, we consult the “Lorenz96 system”, as it describes a system where the dimensionality can be increased as wished. The “Lorenz96 system” was introduced as a climate model and the dimensionality was intended to split up the earth's surface in different zones [27]. For that reason, all the parameters of the system are equal and an N-dimensional system is defined as follows:

$$\dot{x}_d = \left(x_{d+1} - x_{d-2}\right) x_{d-1} - x_d + F \qquad d \in \{1, ..., N\} \tag{5.5}$$

We assume cyclic boundary conditions described by: $x_{-1} = x_{N-1}$, $x_0 = x_N$, and $x_{N+1} = x_1$.

We need to extend this model for our purposes, since our initial guess for the parameters almost perfectly aligns with the real parameters in equation 5.5. For this, we introduce a parameter in front of each variable in each dimension, which leads to the following, parametrised Lorenz96 system:

$$\dot{x}_d = \left(p_{3(d-1)+1}\, x_{d+1} - p_{3(d-1)+2}\, x_{d-2}\right) x_{d-1} - p_{3(d-1)+3}\, x_d + p_{3N+1} \qquad d \in \{1, ..., N\} \tag{5.6}$$

The parameters are stored in a $(3N+1)$-dimensional vector $\underline{p}$. Note that in our parametrised Lorenz96 system we followed the original Lorenz96 system and kept the forcing constant F equal among each dimension d. We stored the forcing constant as the last value in the vector $\underline{p}$, which is why it was renamed to p_{3N+1} in equation 5.6. The parameters in front of the variables, p_1 to p_{3N}, were drawn from a uniform distribution in interval [0, 1] and the forcing constant, p_{3N+1}, was assigned the value 8 in accordance to the original model [27].

For the calculation of the gradient for our optimisation we utilise the nomenclature introduced in section 4.2, where the primary system consisting of discrete data points is denoted by $\{\chi\}$ and the secondary system coupled to the primary one is denoted by $\{y\}$. For each dimension $d \in \{1, ..., N\}$ the gradients with respect to the parameters are calculated by:

$$\frac{\partial \ell}{\partial p_{3(d-1)+1}} = -2\ (\dot{\chi}_d - \dot{y}_d)\ y_{d+1}\, y_{d-1} \tag{5.7a}$$

$$\frac{\partial \ell}{\partial p_{3(d-1)+2}} = +2\ (\dot{\chi}_d - \dot{y}_d)\ y_{d-2}\, y_{d-1} \tag{5.7b}$$

$$\frac{\partial \ell}{\partial p_{3(d-1)+3}} = +2\ (\dot{\chi}_d - \dot{y}_d)\ y_d \tag{5.7c}$$

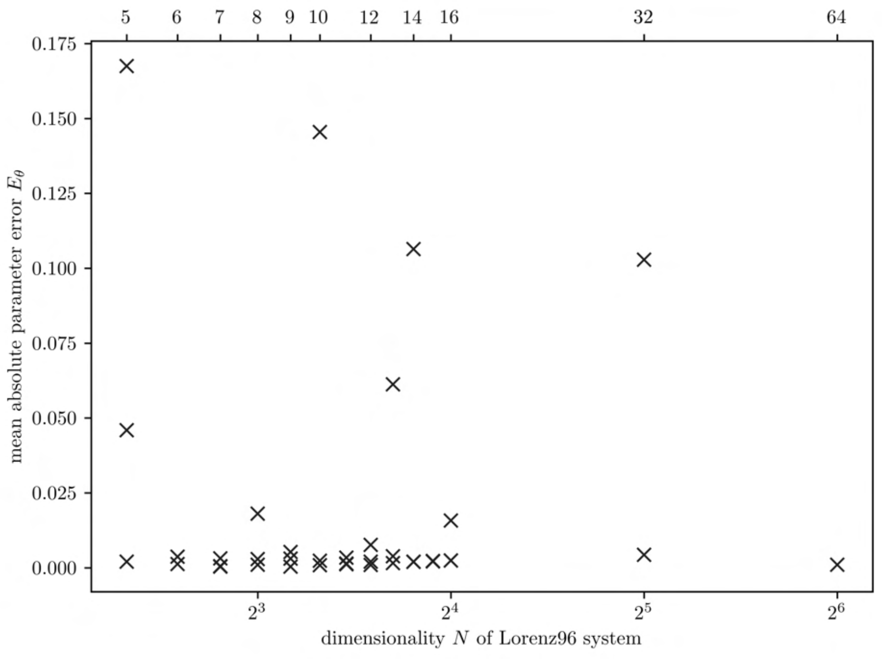
$$\frac{\partial \ell}{\partial p_{3N+1}} = \sum_{d=1}^{N} -2\ (\dot{\chi}_d - \dot{y}_d) \tag{5.7d}$$

The results of the optimisations of the Lorenz96 system are shown in figure 5.2. There we can see that the optimisation works for a wide range of dimensions and is reproducible for vectors with different random parameters. It has to be noted, that due to considerations of computation time, the high dimensional Lorenz96 systems were only calculated once. As expected, we noted that with an increasing dimensionality each step in our algorithm gets computationally more expensive. For a feeling for the optimisation duration, we report that a single optimisation iteration for the five-dimensional Lorenz96 system takes around twenty seconds. For the 64-dimensional Lorenz96 system in contrast, it takes around four-hundred seconds, which is roughly a factor of twenty more.

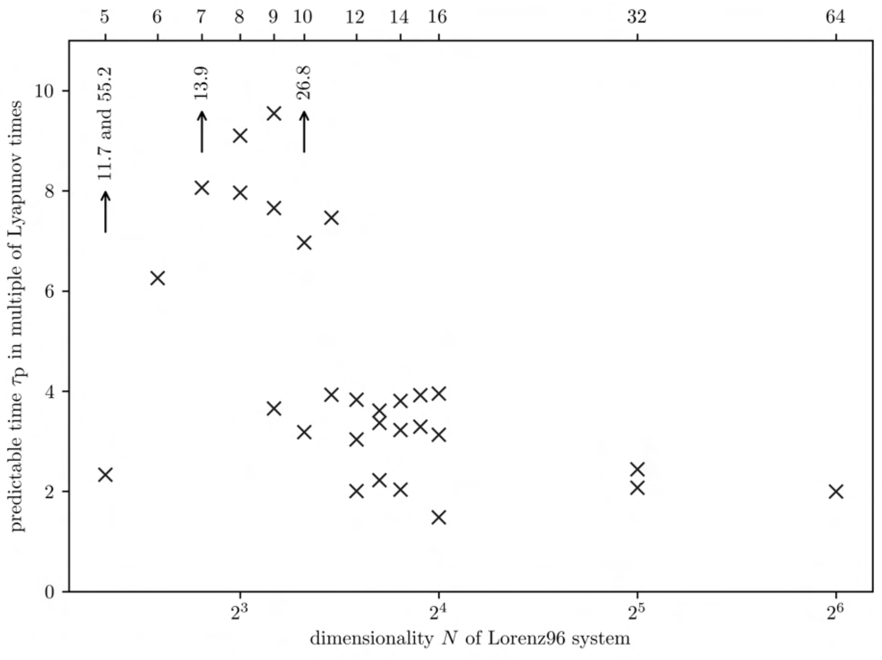
Nevertheless, we have shown that this algorithm is capable of finding the correct set of parameters in a 64-dimensional system with 193 different free parameters. Since we used algorithm 4.2, we did not have any prior knowledge about any parameter. Figure 5.2b shows that the predictability does not suffer with an increasing dimensionality. For each dimension we are able to predict a couple of Lyapunov times ahead, as we are able with the three-dimensional systems in figure 5.1b. Instead, we note that our algorithm does not prefer a specific region of dimensions. This demonstrates the applicability of this algorithm to high-dimensional systems with a large number of free parameters.

In figure 5.2a, we note that the mean absolute parameter error E_θ is of order 10^{-3} across some dimensions. The uncertainty is on that digit due to our choice on how we check for convergence: in this setting we say a parameter has converged, if its change is of order 10^{-4} for twenty consecutive optimisation steps. This leads to a precision on the third digit.

We have shown in this section that our proposed optimisation algorithm works for a wide variety of different systems and a wide variety of different dimensionalities. This emphasises the general applicability of this algorithm.



(a)



(b)

Figure 5.2: This figure shows the quality of the parameter reconstruction for different dimensionalities of the Lorenz96 system. Figure (a) shows the mean absolute parameter error E_θ as defined in equation 5.4 and figure (b) shows the predictable time as defined in equation 5.3 in units of the Lyapunov time. The arrows point to outlier values which do not fit on this scale. For each dimension and each run in that dimension, a different vector of random factors $\underline{p}$ was created. The Lyapunov exponent of each different parameter vector was calculated using the algorithm described in section 2.1.1. The initial condition in each case was the vector $\underline{x}(0) = (8 + 10^{-3}, 8, ..., 8)^\intercal \in \mathbb{R}^N$, as described in the original paper [27]. Due to the high dimensionality of these systems, a finer update step of $\Delta t = 10^{-4}$ was used for countering the effects of numerical imprecisions. The evaluation length was chosen to be $l_\mathrm{e} = 10^4$ and the coupling strength was chosen to be $\alpha = 10^4$ for each run. For most dimensions, multiple runs with different random parameter vectors $\underline{p}$ were performed.

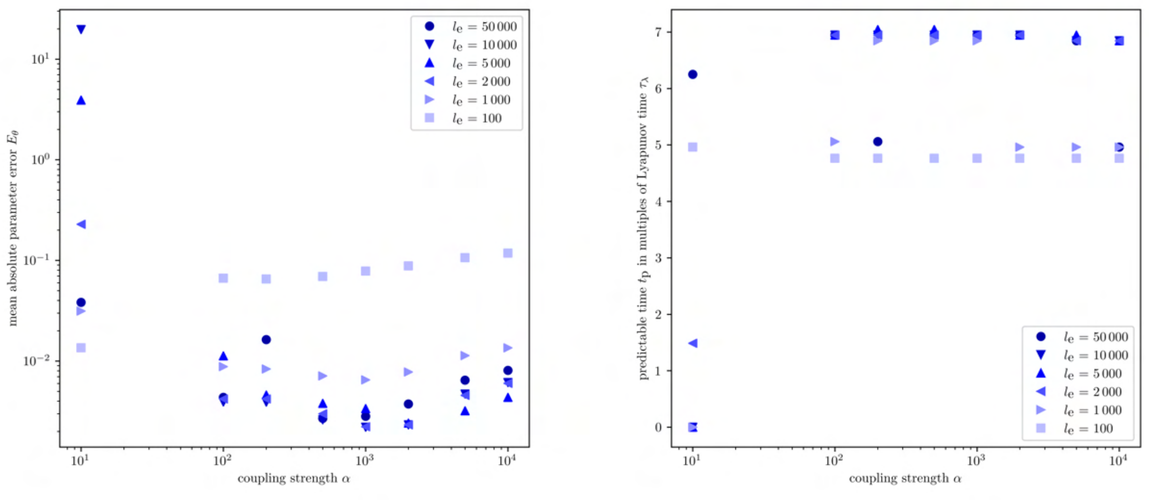
5.1 Effect of Hyperparameters

In our proposed optimisation algorithm we introduced two new hyperparameters: the coupling strength α and the evaluation length l_e. The coupling strength indicates how strongly the secondary system is coupled to the data points making up the primary system, and the evaluation length describes the amount of data used as primary system. In this section we analyse the dependency of our algorithm from the choice of those hyperparameters.

For that we performed a grid-search over different values for the two new hyperparameters. The grid was spanned over several orders of magnitude, since small changes in the hyperparameters proved to have virtually no effect. The details of the calculations and the results thereof are shown in figure 5.3: analysing the dependence on the coupling strength, we can observe that all systems with a coupling strength greater equal to 100 converge and are predictable to the expected extend of a couple of Lyapunov times. For the converging coupling strengths, a longer evaluation length may seem to lead to a better performance compared to a shorter one, since both plots tend to get darker in the direction of better performance. Interestingly, for the longest (50 000) and the shortest (100) evaluation length the coupling strength has no effect and we observe convergence across the whole spectrum of coupling strengths. This is a surprising result and at this point we do not have a hypothesis explaining this phenomenon.

At this point, we have to note that finding the correct hyperparameters is a trade-off between computation time and performance. Increasing the evaluation length naturally leads to an increased computation time, since we simply need to calculate more steps. However unintuitively, we also observed that the computation time increased with a higher coupling strength. Meaning that we cannot set the coupling strength to an arbitrarily high value, since this would mean non-feasible computation time.

In the end we conclude that our optimisation algorithm is generally robust against the exact choice of hyperparameters. As a reasonable starting point, we propose a coupling strength of 10^3 and an evaluation length of 10^4. This is based on our results from figure 5.1 and on the findings in this section. Starting from there, if needed a better performance may be obtained by trial-and-error.



(a) **(b)**

Figure 5.3: This figure shows the results of the grid-search for different hyperparameters utilising the Lorenz63 system with the standard parameters as primary system. For each run we used an initial condition of $(5, 5, 5)^\intercal$ for the primary system and an integration step of $\Delta t = 10^{-3}$. In our scan we used the following values for the coupling strength α: 10, 100, 200, 500, 1 000, 2 000, 5 000, and 10 000. For the evaluation length l_e we used the following values: 100, 1 000, 2 000, 5 000, 10 000, and 50 000. We calculated the mean absolute prediction error E_θ and the predictability time t_p for each possible combination of the two hyperparameters.

5.2 Robustness Against Noise & Incorrect Knowledge

So far, all calculations have been performed on synthetic systems, which are precise and error-free by construction[1]. However, data obtained from real-world systems is never error-free and has always a certain noise, an uncertainty, with it. Additionally, until now we assumed perfect knowledge of the term structure of the governing ODEs producing the primary data. This assumption may not necessarily hold for real-world systems, as the deriving of the equations also suffers from the uncertainty in the data in its measurements [1]. Therefore, in this section we want to analyse the robustness of our proposed optimisation algorithm. We analyse the robustness with respect to two factors: firstly, noise in the data, and secondly an imperfect assumption about the governing equations.

5.2.1 Robustness Against Noisy Data

We begin by analysing the robustness against noise in the data making up the primary system. For that, we simulate the classical Lorenz63 system with its standard parameters and add a Gaussian noise on each data point and each coordinate. The Gaussian noise is drawn from a normal distribution with zero mean and a standard deviation σ. In our experiments, the standard deviations σ range from 0.01 to 2. Before adding the Gaussian noise to the precise data, we calculate the **s**ignal-to-**n**oise **r**atio (SNR) as the ratio of the power of the signal to the power of the noise [28]. With the power being calculated as the squared amplitude A, we can express the SNR with the following equation:

$$\mathrm{SNR} = \left(\frac{A_\mathrm{signal}}{A_\mathrm{noise}}\right)^2 \tag{5.8}$$

[1]At this point we want to note that different integration methods may lead to different results when integrating over many steps due to the finite precision of computers and the chaotic nature of these systems.

The signal-to-noise ratio is often reported on a logarithmic scale in decibel:

$$\mathrm{SNR} = 10\,\log_{10}\left(\frac{A_{\text{signal}}}{A_{\text{noise}}}\right)^2\ \mathrm{dB} \tag{5.9}$$

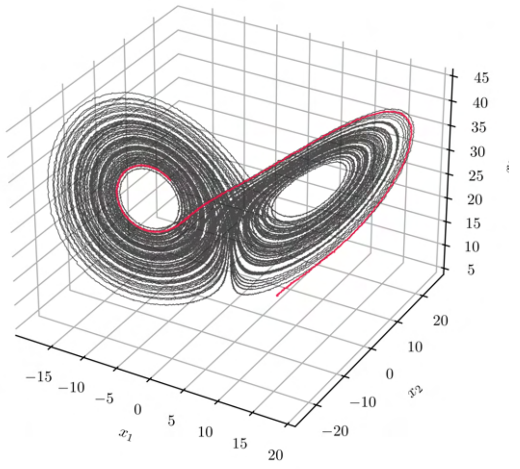
The amplitude A of a time series $\underline{x}(t)$ can be calculated as follows:

$$A = \sqrt{\frac{1}{t}\int_0^t \left\|\underline{x}(\tilde{t})\right\|^2 \mathrm{d}\tilde{t}} \tag{5.10}$$

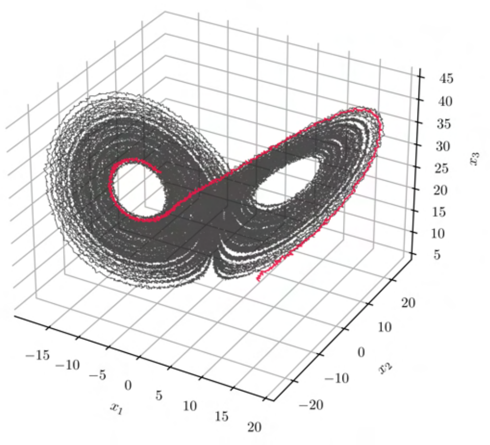
We ran two distinct experiments and in the first we analysed the dependency of the noisy data on the coupling strength, and the other time the dependency on the evaluation length. In general, we can state that using the proposed configuration in previous section 5.1, we obtain a meaningful predictability starting from around 30 dB, meaning that the power of the signal has to be 10^3 stronger than the power of the noise. To get a feeling for the noise levels, figure 5.4 shows the Lorenz attractor for different noises.

Dependency on the Coupling Strength

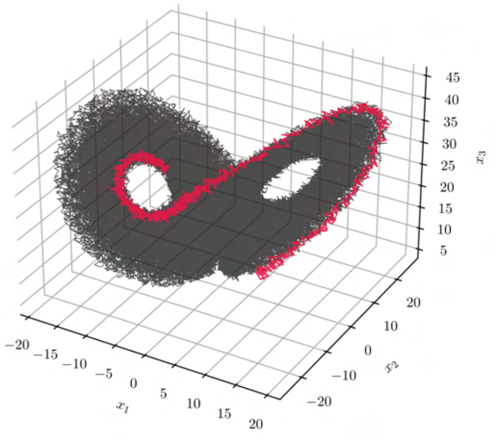
Regarding the dependency on the coupling strength, we consult figure 5.5. There we see an interesting pattern emerge in figure 5.5a: the parameter errors seem to follow a power law in a certain region. The cut-off of this power law-behaviour at an error of about 10^{-2} stems from our convergence criterion for these runs. We observe that a lower coupling strength leads to a more negative exponent in the fit of said observed power law. This implies a better performance for lower coupling strengths, since it means that the error does not grow as quick with a rising noise level. This finding can be confirmed by figure 5.5b, where we see that the lower the coupling strength, the more noise is acceptable for levels of predictability. Again confirming the finding that a lower coupling strength works better on noisy data. This result is not intuitive as we would expect a higher coupling strength to perform better. Our working hypothesis for this effect is that the original data is noisy and therefore imprecise. If we couple too strongly to the imprecise data, we optimise on a wrong system. However, if the coupling is not too strong, the effect of the noisy data remains limited.



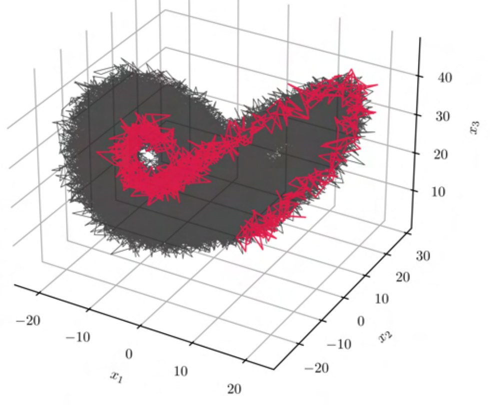
(a) SNR = 50.15 dB, $\sigma = 0.05$



(b) SNR = 40.61 dB, $\sigma = 0.15$

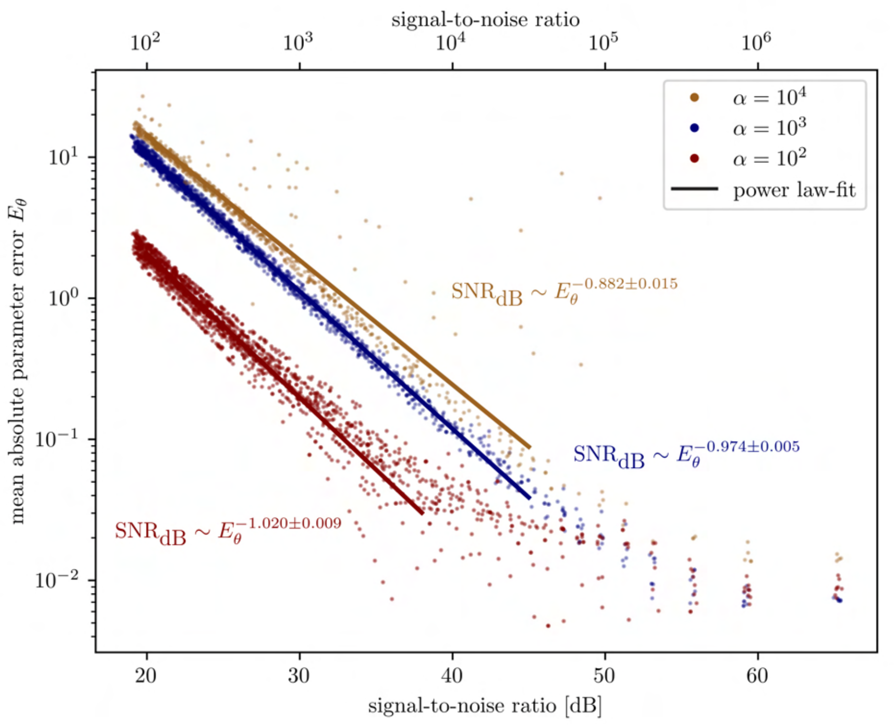


(c) SNR = 30.15 dB, $\sigma = 0.5$

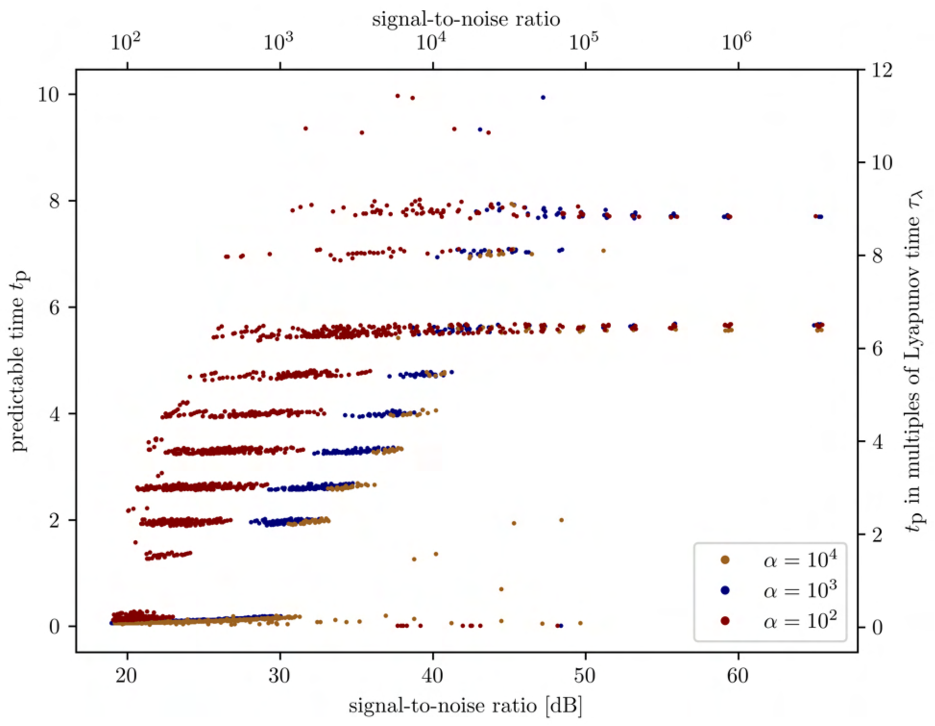


(d) SNR = 20.04 dB, $\sigma = 1.6$

Figure 5.4: This figure shows the Lorenz attractor for different levels of noise. The Lorenz attractor was calculated with the standard parameters, an initial condition of $(5, 5, 5)^\intercal$, and an integration step of $\Delta t = 10^{-3}$. A Gaussian noise with zero mean and standard deviation σ was added on each time step and each coordinate. The red part of each plot visualises an evaluation length of 10^3: the part of the attractor which the parameters are actually calibrated upon.



(a)



(b)

Figure 5.5: This figure shows the results of the optimisation algorithm in dependence to the level of noise present. Figure (a) shows the mean absolute parameter error E_θ and figure (b) shows the predictable time t_p in terms of Lyapunov time. This experiment was performed with the Lorenz63 system with the classical parameters and the initial condition $(5, 5, 5)^\intercal$. The possible standard deviations for the Gaussian noise ranged from 0.01 to 2 in increments of 0.01. The signal-to-noise ratio has been calculated using the equations 5.8f. The coupling strength 10^2 shows 1 718 realisations, the coupling strength 10^3 shows 1 594 realisations, and the coupling strength 10^4 shows 726 realisations. For all runs the evaluation length l_e was 10^3.

Dependency on the Evaluation Length

In this section we analysed the performance of our optimisation algorithm on noisy data with regard to the evaluation length while keeping the coupling strength constant. For that, we chose the best performing coupling strength from previous section, namely $\alpha = 10^2$ and perform the same experiment as before — only this time we vary the evaluation length and keep the coupling strength constant.

The findings of this experiment are shown in figure 5.6, where we observe inconclusive results: we note that an evaluation length of 10^2 seems to be too short for the algorithm. Meaning that a hundred data points is not enough. Even on data sets where the noise level is very low, e.g. the signal being a million times more powerful than the noise, we find that the evaluation length of 10^2 performs an order of magnitude worse than the longer evaluation lengths. However, when comparing the results of the evaluation lengths of 10^3 and 10^4 with each other, we see a very similar performance with the shorter one performing a little bit better. This confirms our findings from section 5.3, where we did not observe a big dependence on the evaluation length. For noisy data, this experiment puts the additional constraint on the evaluation length of it being sufficiently large. If the evaluation length is too small, the algorithm will not perform well on noisy data.

Figure 5.6 nicely illustrates the trade-off discussed in previous section 5.1. We can clearly see the effect of the evaluation length on the required computational time by the amount of performed experiments. The experiments for evaluation lengths $l_e = 10^3$ and $l_e = 10^4$ ran on the same computing cluster with the same configuration for each two days. While we were able to calculate 1 718 optimisations for the evaluation length of 10^3, we were only able to run 295 optimisations for the longer evaluation length. Nevertheless, both show similar performance. Studying these results, we recommend starting with small evaluation lengths and increasing them until an acceptable precision is reached, since a longer evaluation length offers no tangible benefit.

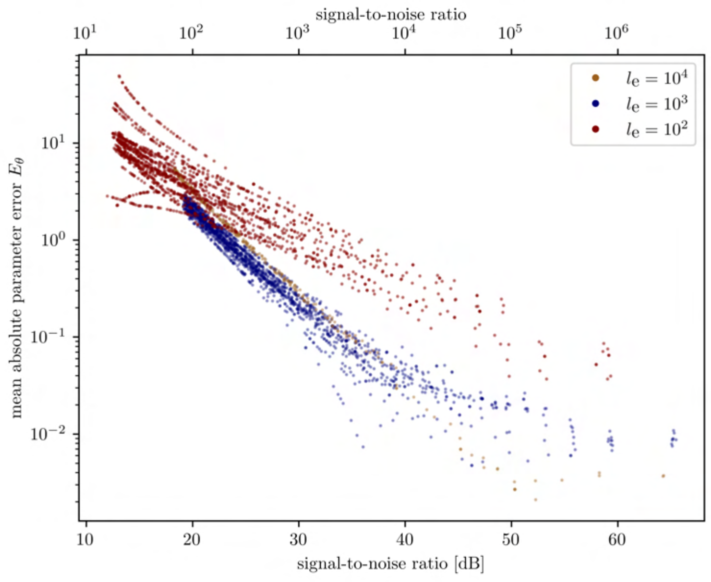


Figure 5.6: This figure shows the results of the optimisation algorithm working on different noise levels with different evaluation lengths l_e. We present the the mean absolute parameter error as defined in equation 5.4. This experiment was performed using the Lorenz63 system with its standard parameters and the initial condition $(5, 5, 5)^\intercal$. The coupling strength was kept at a constant value of $\alpha = 10^2$ for each experiment with the possible values for the standard deviations of the Gaussian noise ranging from 0.01 to 2. The evaluation length $l_e = 10^2$ shows 1 506 realisations, the evaluation length $l_e = 10^3$ shows 1 718 realisations and the evaluation length $l_e = 10^4$ shows 295 realisations.

5.2.2 Robustness Against Imprecise Knowledge of the Governing Equations

Up until now, we assumed perfect knowledge of the term structure of the underlying equations producing the data. However, this assumption may not hold in every case, which is why in this section we analyse the performance of our algorithm for an imprecise assumption of our governing equations.

There are two ways in which the underlying equations can be incorrect: they can either include terms which are not responsible for creating the data, or they could leave terms out which are indeed responsible for creating the data. The strategy for countering the over-inclusion of terms is setting the factor of each wrong term to 0 — or at least very close to it — so its effect on the governing equations is highly limited. However, at this point we note that our algorithm is helplessly exposed to the under-inclusion of terms and we have to viable strategy to deal with that problem.

In their article the authors presented a framework with which the structure of governing equations can be reconstructed from data; however, they also noted that the reconstruction can include wrong terms in the equations [1]. Their algorithm is built in a way to over-include terms, rather than not including them. Therefore, they are more concerned with the first type of uncertainty mentioned previously.

In the following we present a synthetic example and analyse how well our algorithm handles the over-inclusion of terms. For this purpose we introduce an alternate form of the Lorenz63 system with two additional terms, a linear one and a non-linear one:

$$\begin{aligned} \dot{x}_1 &= -\sigma\, x_1 + \sigma\, x_2 + \kappa\, x_2\, x_3 \\ \dot{x}_2 &= -x_1\, x_3 + \rho\, x_1 - x_2 + \xi\, x_3 \\ \dot{x}_3 &= x_1\, x_2 - \beta\, x_3 \end{aligned} \tag{5.11}$$

The gradient of the parameters σ, ρ, and β of this system are the same as for the classical Lorenz63 system in equations 4.12. However, with the introduction of the two additional parameters κ and ξ, we obtain two two additional gradients of the following form:

$$\frac{\partial \ell}{\partial \kappa} = -2\left(\dot{\chi}_1 - \dot{y}_1\right)\, y_2\, y_3 \tag{5.12a}$$

$$\frac{\partial \ell}{\partial \xi} = -2\left(\dot{\chi}_2 - \dot{y}_2\right)\, y_3 \tag{5.12b}$$

Again we used the notation introduced in section 4.2.2, with y describing the secondary system and χ the data points making up the primary system.

We perform the optimisation as described in algorithm 4.2: Our data, primary system, consists of data points of the classical Lorenz63 with its normal parametrisation. However, when we calculate the secondary system ('estimated system' in aforementioned algorithm) we use modified Lorenz96 system in equations 5.11 as foundation. Consequently, we need to calculate the gradients for all five parameters using the combined equations 4.12 and 5.12. In the end this means, that we try to fit the data coming from the classical Lorenz63 system onto the equations 5.11 containing wrong terms, which did not contribute to the data.

The results of the parameter optimisation are shown in figure 5.7. There we can see that our algorithm is capable of correctly identifying that the terms '$+\kappa\, x_2\, x_3$' & '$+\xi\, x_3$' of equations 5.11 were not contributing to the observed data points, and therefore put their coefficient to a value close to zero. However, we note that the parameters are not exactly zero. This result means that this algorithm can be used to a certain extent as a corrective measure when deriving the governing equations of systems from data.

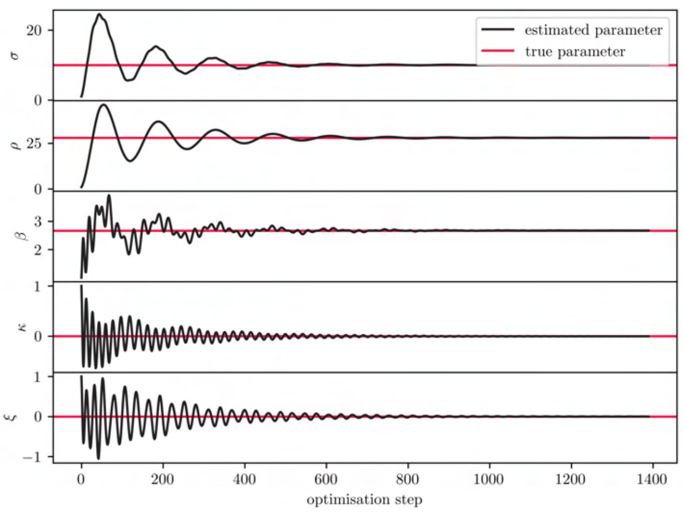


Figure 5.7: This figure shows the results of the optimisation algorithm using a wrong assumption on the governing equation. In fact, we are trying to fit data stemming from the classical Lorenz63 system with parameters $(\sigma, \rho, \beta)^\intercal = \left(10, 28, \frac{8}{3}\right)^\intercal$ and initial condition $(5, 5, 5)^\intercal$ on the wrong equations 5.11. We used an integration step of $\Delta t = 10^{-3}$, a coupling strength of $\alpha = 10^2$, and an evaluation length of $l_\mathrm{e} = 10^3$.

In figure 5.8 we analyse the effect of incorrect assumptions on the predictability. For that, we optimise the same run on two different assumptions: the correct and incorrect one. We then compare the predictability of each run. Additionally, we compare the predictability when we discard the terms with a coefficient close to zero. Our findings are as expected: the predictability decreases with the incorrect equation assumption. Removing the terms with the coefficients close to zero, leads to an increase in the predictability. However, we note that these effects are marginal and barley noticeable. Additionally, removing terms leads to another judgement call when deciding which coefficients are "close to zero". This can get especially problematic when dealing with systems which have small coefficients by construction.

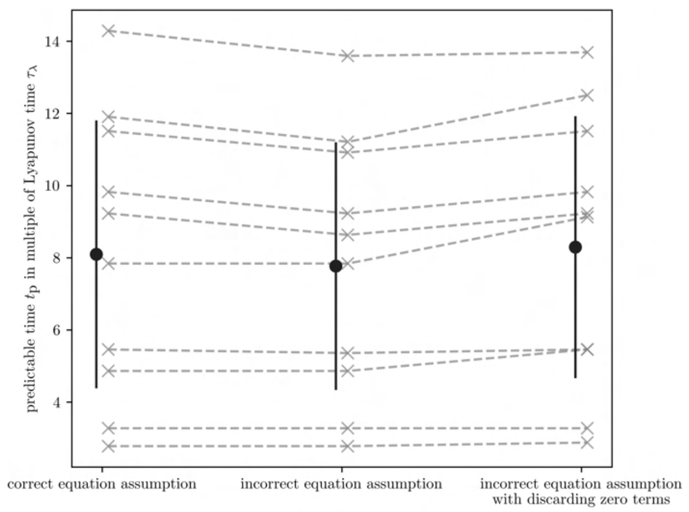


Figure 5.8: This figure compares the predictable time t_p for different assumptions of the governing equations of the Lorenz system. We performed ten runs and the grey markers connected by a dotted line show the result for a single run. The black dot with the bar shows the mean with one standard deviation for each category. For each run we performed two different optimisations one with the correct assumption for the governing equation and one with the incorrect one. For the last column, we removed the terms where the optimisation set the coefficients close to zero. For each run we used a different initial condition, where each coordinate was chosen as a random number between -10 and 10. As incorrect assumption of the Lorenz system we used equations 5.11. We used an integration step of $\Delta t = 10^{-3}$, a coupling strength of $\alpha = 10^2$, and an evaluation length of $l_\mathrm{e} = 10^3$.

Looking at these results it can be prompting to use this optimisation as an algorithm for finding the governing equations from a time series using a library of terms. The basic idea being that we chose a highest order and include all possible terms for that order. The hypothesis being that the optimisation will set all the terms to zero not responsible for creating the data; just as it did in previous example.

For a three-dimensional time series with a library including terms up to order two, the skeleton system would be defined as the following system of ODEs:

$$\begin{aligned} \dot{x}_1 &= c_1\, x_1 + c_2\, x_2 + c_3\, x_3 + c_4\, x_1^2 + c_5\, x_2^2 + c_6\, x_3^2 + c_7\, x_1\, x_2 + c_8\, x_1\, x_3 + c_9\, x_2\, x_3 + c_{10} \\ \dot{x}_2 &= c_{11}\, x_1 + c_{12}\, x_2 + c_{13}\, x_3 + c_{14}\, x_1^2 + c_{15}\, x_2^2 + c_{16}\, x_3^2 + c_{17}\, x_1\, x_2 + c_{18}\, x_1\, x_3 + c_{19}\, x_2\, x_3 + c_{20} \\ \dot{x}_3 &= c_{21}\, x_1 + c_{22}\, x_2 + c_{23}\, x_3 + c_{24}\, x_1^2 + c_{25}\, x_2^2 + c_{26}\, x_3^2 + c_{27}\, x_1\, x_2 + c_{28}\, x_1\, x_3 + c_{29}\, x_2\, x_3 + c_{30} \end{aligned} \tag{5.13}$$

Here the vector $\underline{c} \in \mathbb{R}^{30}$ holds the parameters for this system. Performing this optimisation, using Lorenz data as primary system, shows very interesting results: the parameters converge towards 'wrong' values, as can be seen in figure 5.9.

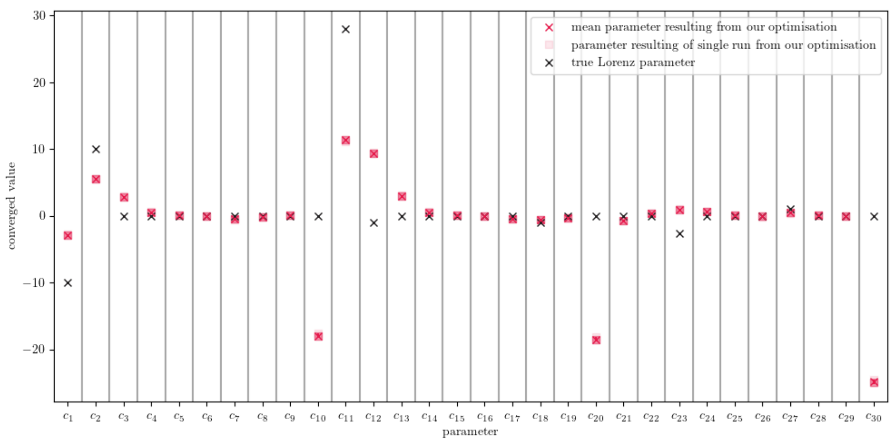


Figure 5.9: This figure shows the results of applying our optimisation algorithm utilising data from the Lorenz system as primary system and the shell equations 5.13 as assumption for the governing equations. We performed the optimisation five times using different initial conditions on the attractor, and we can see that the results are stable. We used an integration step of $\Delta t = 10^{-3}$, a coupling strength of $\alpha = 10^3$, and an evaluation length of $l_e = 10^4$.

However, it is hard to classify the found parameters as wrong: the shell equation 5.13 together with the found parameters shown in figure 5.9 produce an attractor, which is a carbon copy of the Lorenz attractor. This can be seen in figure 5.10, where the black line shows a path on the Lorenz attractor, and the colourful paths show trajectories following equation 5.13. These colourful paths with different initial conditions show that the equations 5.13 with found coefficients behave chaotically and form an attractor.

Looking at characteristic numbers classifying an attractor, the correlation dimension D [29] and the largest Lyapunov exponent are often used. In our calculations we discovered that both attractors differ in their largest Lyapunov exponents: $\lambda_{\text{Lorenz63}} = 0.902 \pm 0.0013$, $\lambda_{\text{Shell system}} = 0.64 \pm 0.06$. Nevertheless, this result raises questions about the uniqueness of attractors. How many sets of equations exist reproducing the butterfly shape of the Lorenz attractor? We have shown that using the library of equations 5.13 results in always the same set of coefficients. To which coefficients other libraries of equations converge to, and whether they reproduce the same attractor remain questions for further research.

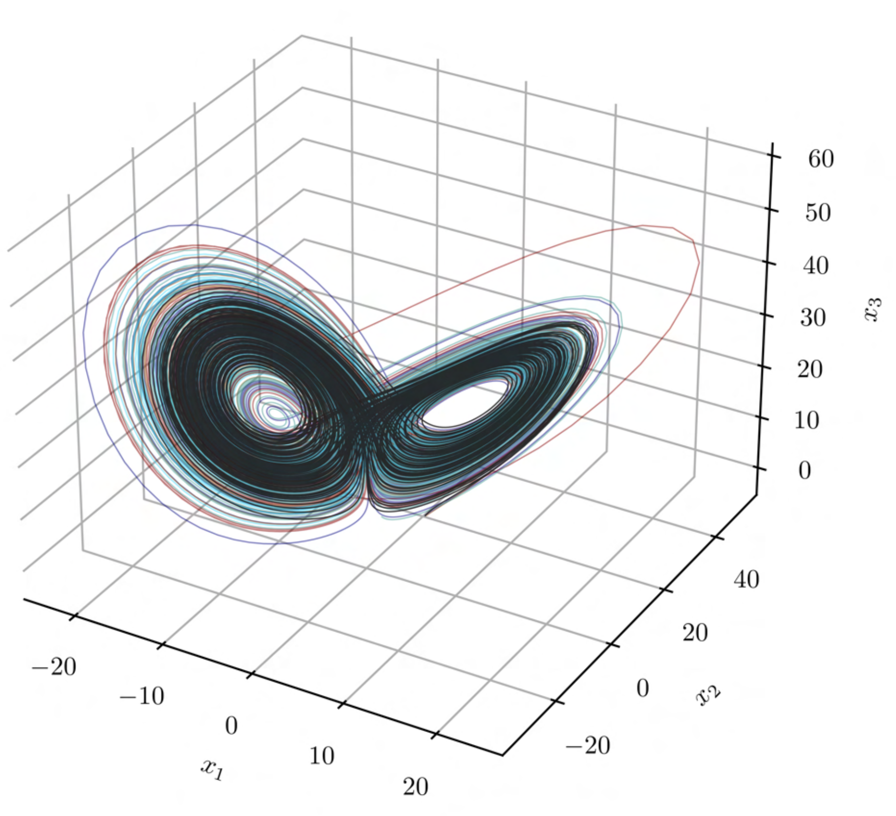


Figure 5.10: This figure shows the classical Lorenz attractor with its standard parameters in black with the initial condition $(5, 5, 5)^\intercal$. The colourful trajectories follow the equations 5.13 with the parameters from figure 5.9. Each trajectory has been integrated for 10^5 steps with an integration step size of $\Delta t = 10^{-2}$.

5.3 Comparison to Similar Work

In this section we compare our algorithm to a similar piece of work presented in [2]. Since our algorithm describes a further development of the algorithm in [2], it is not surprising that these two approaches are similar in nature. In their article they propose the following algorithm, where the starting situation is a primary system and a secondary system coupled to the primary one: they change the parameters of the secondary system in the direction minimising a loss function until the secondary system synchronises with the primary one. Once they are synchronised, the parameters of the secondary system are an estimation of the parameters of the primary system [2]. The special property of their algorithm is, that it works "online" meaning that they do not visit the same dataset multiple times. Instead, they only visit the data once and perform the parameter update-step after each integration step. This is in contrast to our algorithm, where we iterate multiple times over the data containing l_{e} data points.

The authors state that the primary system may consist of discrete data points. However, in their article the authors only presented the results for simultaneous integration, where the primary system is given by equations, which can be evaluated for any arbitrary values. In order to compare it to our proposed algorithm, we use our integration method presented in section 4.1.2. With that integration method, we implemented their algorithm to work with discrete data as primary system: the results for a optimisation are shown in figure 5.11, visualising that their approach works for the Lorenz system.

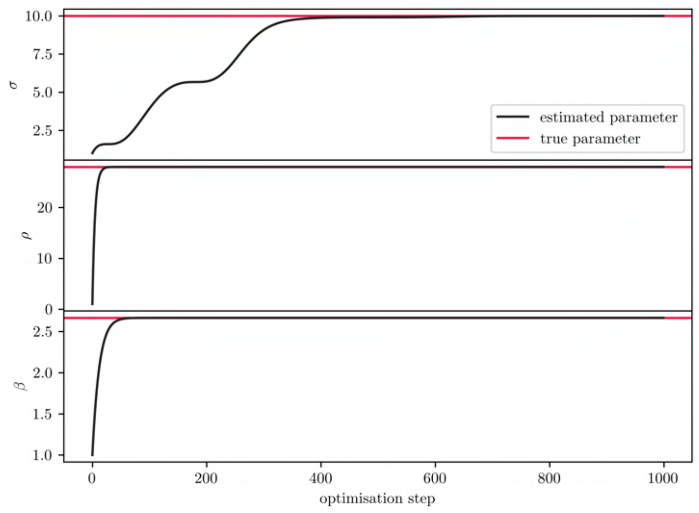


Figure 5.11: This figure shows the result of the optimisation for the Lorenz63 system using the algorithm in [2]. The primary system consists of discrete data points of a Lorenz realisation with the standard parameters, an initial condition of $(10, 8, 20)^{\mathsf{T}}$, and an integration step size of $\Delta t = 10^{-3}$. The learning rate vector used in equation 5.16 is $\underline{\eta} = \left(10^{-3}, 10^{-3}, 10^{-4}\right)^{\mathsf{T}}$.

In the following, we want to present the algorithm in [2] in more detail, where we use their article as source. The authors consider the following loss function $\mathcal{L}$ in the interval $\mathcal{I}$ of length T, which is the continuous version of our loss function in equations 4.10:

$$\mathcal{L} = \frac{1}{T} \int_{\mathcal{I}} \left\| \underline{\dot{x}}(t) - \underline{\dot{y}}(t) \right\|^2 \mathrm{d}t \tag{5.14}$$

They then calculate the gradient and the update step resulting from that gradient the following way:

$$\nabla \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \theta^1}, \ldots, \frac{\partial \mathcal{L}}{\partial \theta^N} \right)^{\mathsf{T}} \tag{5.15}$$

$$\underline{\theta}_{i+1} = \underline{\theta}_i - \underline{\eta} \odot \nabla \mathcal{L} \tag{5.16}$$

Here the superscript signalises the n-th entry in a N-dimensional vector and the subscript indicates the current iteration step. Again, $\odot$ symbolises the Hadamard product.

In their article they used the Lorenz system as an example with a continuous primary system, which they could evaluate at arbitrary values. However, since the case with a discrete primary system is more application-related[2], we present that in the following. We use data from the standard Lorenz as defined in equations 2.10 as primary system. As a secondary system they use the system coupled via the active-passive-decomposition introduced in section 3.1.1. Here the coupling is visualised in red.

$$\begin{aligned} \dot{y}_1 &= -\sigma\, y_1 + \sigma\, \chi_2 \qquad (5.17) \\ \dot{y}_2 &= \rho\, y_1 - y_1\, y_3 - y_2 \\ \dot{y}_3 &= -\beta\, y_3 + y_1\, y_2 \end{aligned}$$

Note, that these equations are structurally identical to equations 3.12; however, we replaced x with χ to highlight the fact, that the primary system consists of discrete points treated as coefficients in this system.

Calculating the partial derivative with respect to the parameters leads to the following gradients, where we introduce $\|\underline{e}\|^2 = \left\|\dot{\underline{\chi}}(t) - \dot{\underline{y}}(t)\right\|^2$ for the sake of readability:

$$\frac{\partial \|\underline{e}\|^2}{\partial \sigma} = -2(\dot{\chi}_1 - \dot{y}_1)\,(\chi_2 - y_1) \qquad (5.18a)$$

$$\frac{\partial \|\underline{e}\|^2}{\partial \rho} = -2(\dot{\chi}_2 - \dot{y}_2)\, y_1 \qquad (5.18b)$$

$$\frac{\partial \|\underline{e}\|^2}{\partial \beta} = +2(\dot{\chi}_1 - \dot{y}_1)\, y_3 \qquad (5.18c)$$

Here, it is worth pointing out, that these gradients are not identical to the gradients of our algorithm for the Lorenz system in equations 4.12. More precisely, the gradient with respect to the parameter σ is different for our system (equation 4.12a) and this system (equation 5.18a). The difference is signalised by the red colour: due to the coupling by inserting a variable of the primary system, this gradient is directly dependant from the data. This could possibly lead to numerical instabilities when considering noisy data.

As mentioned, the optimisation is performed online, meaning: first, we integrate a single step of the secondary system $\underline{y}_{i+1}$ using our integration method of section 4.1.2. We then calculate the gradients using equations 5.18. Lastly, we perform the update step in equation 5.16 and obtain a new estimation for our set of parameters. We repeat these steps until the parameters have converged. A result of this optimisation is shown in figure 5.11, where we can see the algorithm working.

The advantage of this algorithm is, that it is very fast compared to our proposed one, since it only visits the data once. However, in their presentation the algorithm has a major limitation: it only works for the Lorenz63 system or systems for which an active-passive-decomposition can be found. As we have stated before, there is no guarantee that such a decomposition must exist and therefore the applicability remains limited. Additionally, the optimiser is not sophisticated and the determination of the learning rate $\underline{\eta}$ in equation 5.16 has to be done by trial-and-error.

In the following we analyse the stability of the algorithm in [2] against noisy data and incorrect assumptions of the governing equations and compare it to the performance of our algorithm in said categories.

[2]For integrating the primary system on the spot, the correct parameters have to be known a priori. This defeats the purpose of calibrating data to a set of equations.

Robustness Against Noisy Data

In this section we compare the algorithms with respect to their performance on noisy data. For this purpose, we use the same experimental setting as in section 5.2.1 and produce the same noisy data as described there. Figures 5.13 show the results for a single optimisation run. There, we observe the algorithm of [2] to be very unstable compared to ours. This is expected behaviour, since in the algorithm of [2] the gradient is calculated after each data point and since each data point contains noise, the gradient jumps with that noise. Whereas in our proposed algorithm the noise is averaged over the evaluation length leading to smoother gradients and smoother evolution of the parameters with an actual convergence. Nevertheless, we need to point out that taking the average over a late window of optimisation steps leads to good results for the algorithm of [2], as visualised as grey boxes in figure 5.13a.

In figure 5.12 we can see a comparison between the two algorithms for multiple runs with different noise levels. Interestingly, the points stemming from the algorithm in [2] do not show an apparent power law-behaviour, instead the results seem to be more scattered compared to the ones of our algorithm. For a given noise level, our algorithm seems to produce more stable results than the algorithm in [2]. Additionally, while the algorithm in [2] produces sometimes better results, our proposed algorithm seems to perform better on average.

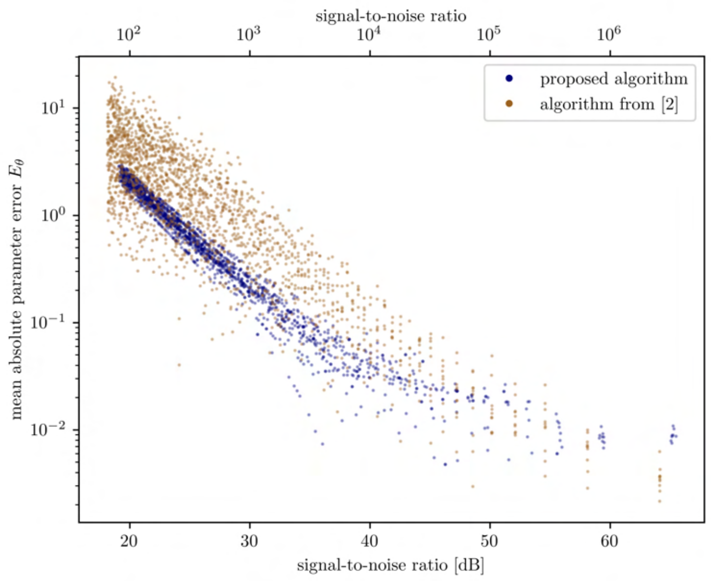
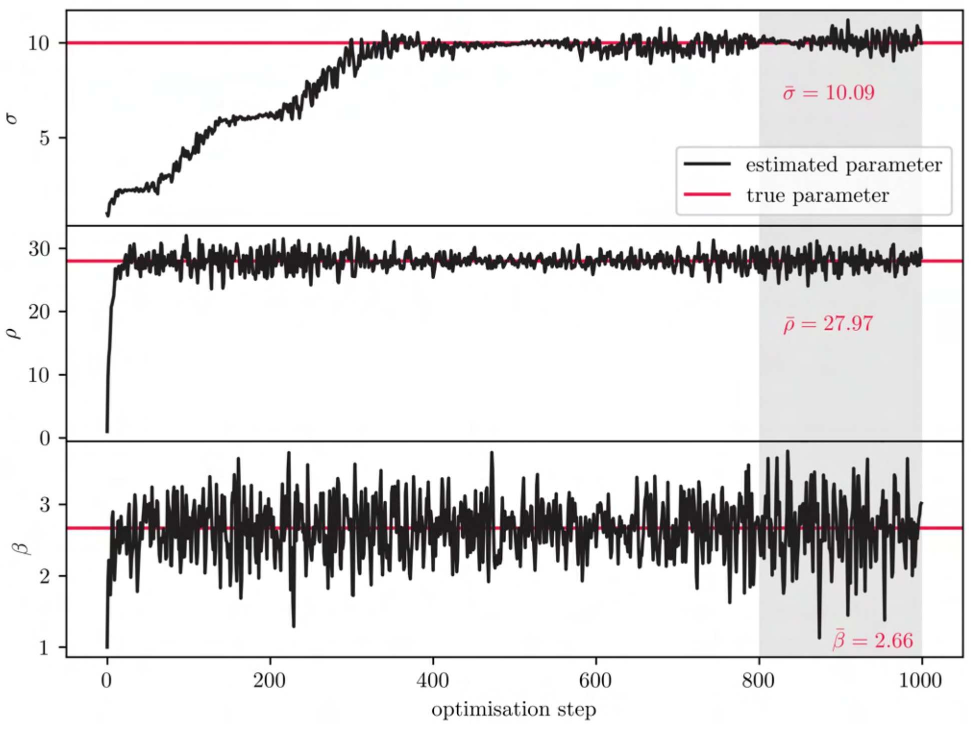
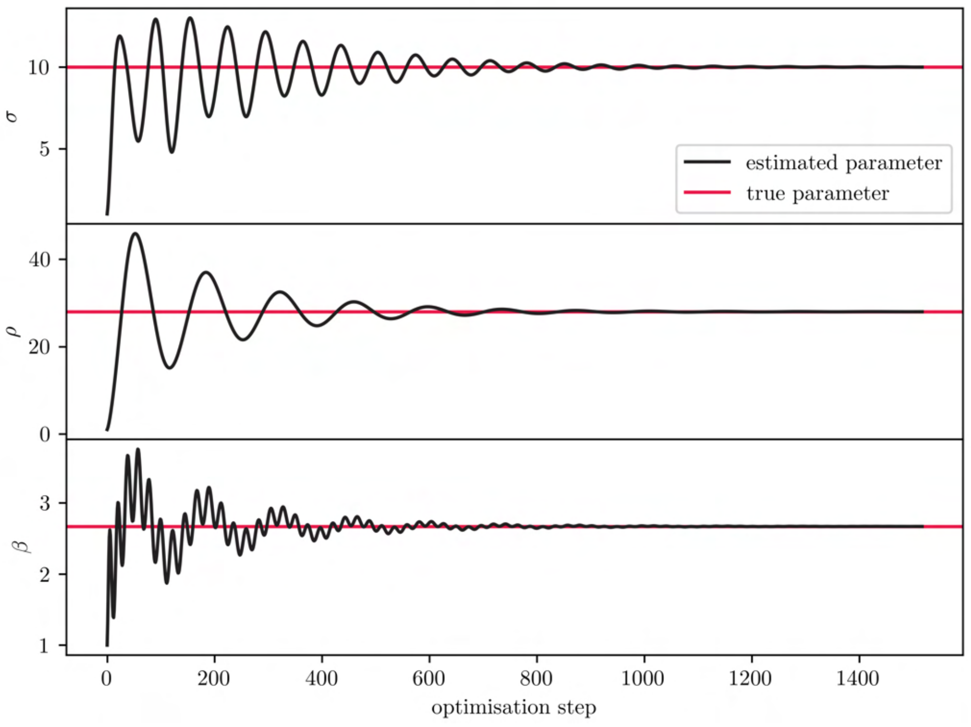


Figure 5.12: This figure compares the results from the algorithm in [2] to our proposed algorithm on the optimisation on noisy data. This experiment was performed using the noisy Lorenz63 system with its standard parameters, an initial condition of $(5, 5, 5)^\intercal$, and an integration step size of $\Delta t = 10^{-3}$. For each time step and each variable, a noise value was drawn from a Gaussian distribution with zero mean and a standard deviation of σ. The standard deviations σ ranged from 0.01 to 2 in increments of 0.01. For our algorithm we used an evaluation length of $l_\mathrm{e} = 10^3$ and a coupling strength of $\alpha = 10^2$. For the algorithm in [2] we calculated a thousand steps and obtained the parameters by averaging over the last hundred iterations. For our algorithm we show 1 718 realisations and the algorithm of [2] shows 1 990 realisations.



(a) Algorithm presented in [2]



(b) Our proposed algorithm

Figure 5.13: This figure compares the optimisation for the algorithm presented in [2] in figure (a), and figure (b) shows the optimisation from our proposed algorithm. The data to be fitted on consists of a noisy Lorenz system with standard parameters, an initial condition of $(10, 8, 20)^\intercal$, and an integration step of $\Delta t = 10^{-3}$. The noise was created by adding a random value drawn from a Gaussian distribution with zero mean and a standard deviation $\sigma = 1$ to each time step and each coordinate. This resulted in a signal-to-noise ratio of $\mathrm{SNR} \approx 26\,000 = 44.15\,\mathrm{dB}$. For the algorithm in [2] in figure (a) we used a learning rate of $\underline{\eta} = \left(10^{-3}, 10^{-3}, 10^{-4}\right)^\intercal$. The number in the grey box indicates the averaged value in that interval for that parameter. For our optimisation algorithm in figure (b) we used an evaluation length of $l_\mathrm{e} = 10^3$ and a coupling strength of $\alpha = 10^2$.

Robustness Against Imprecise Knowledge of the Governing Equations

In this section we analyse the performance based on incorrect assumption on the governing equations. Here, we perform the same experiment as in section 5.2.2 and we use the same Lorenz with additional terms as introduced in equations 5.11. Due to the active-passive-decomposition chosen by the authors, we need to modify the governing equations of 5.11 and exchange y_2 with the primary system x_2 for the equation of $\dot{y}_1$ to ensure the coupling stays as desired. This results in the following equations, where the changing term is visualised in red:

$$\begin{aligned} \dot{y}_1 &= -\sigma\, y_1 + \sigma\, \chi_2 + \kappa\, \chi_2\, y_3 \\ \dot{y}_2 &= -y_1\, y_3 + \rho\, y_1 - y_2 + \xi\, y_3 \\ \dot{y}_3 &= y_1\, y_2 - \beta\, y_3 \end{aligned} \tag{5.19}$$

This also changes the gradients; while the gradients for the parameters σ, ρ, and β are the as in equations 5.18, the gradients for the parameters κ and ξ are different to 5.12 due to the different coupling of this algorithm:

$$\frac{\partial\, \|\underline{e}\|^2}{\partial \kappa} = -2(\dot{\chi}_1 - \dot{y}_1)\, \chi_2\, y_3 \tag{5.20a}$$

$$\frac{\partial\, \|\underline{e}\|^2}{\partial \xi} = -2(\dot{\chi}_2 - \dot{y}_2)\, y_3 \tag{5.20b}$$

The result for this optimisation is shown in figure 5.14, where we can see the parameters converging to the wrong value. Note the absurdly huge values the parameters σ and κ reach: orders of 10^{17}. Interestingly, the parameter β converges to the correct value. The parameter β is in the only equation containing correct terms.

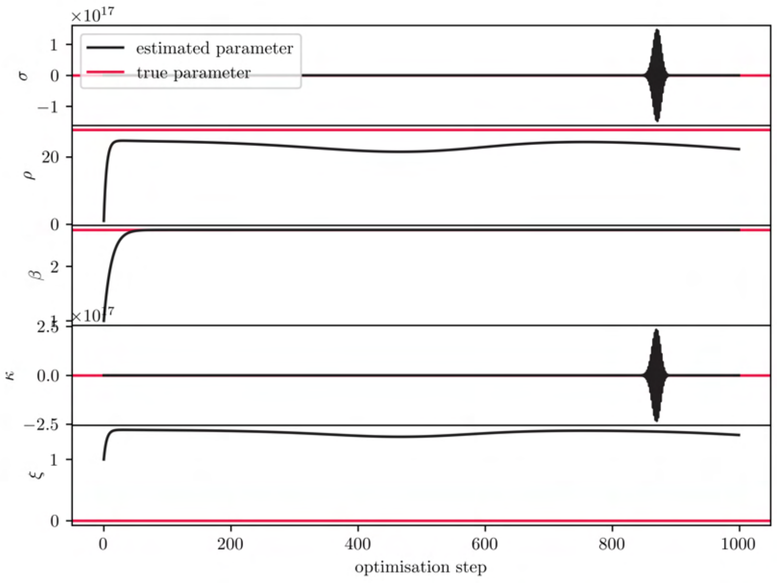


Figure 5.14: This figure shows the result for the optimisation using the algorithm in [2]. The data, primary system, was created using the Lorenz system with its classical parameters, an initial condition of $(10, 8, 20)^\intercal$, and an integration step of $\Delta t = 10^{-3}$. The secondary system was described by equations holding incorrect assumptions and therefore wrong terms. Specifically, the equations 5.19 with the gradients in equations 5.18 & 5.20 were used. As learning rate we used $\underline{\eta} = \left(10^{-3}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-5}\right)^\intercal$.

Conclusion

When comparing the algorithm presented in [2] to our proposed algorithm, we first note that due to our coupling, our algorithm is applicable to arbitrary systems. The algorithm in [2] is orders of magnitude faster than ours, since it has to visit the data only once and performs an online optimisation. However, this online optimisation leads to instabilities against noise and incorrect knowledge of the governing equations. We observed that our algorithm is more robust against those two instabilities. Therefore, unless computational speed is a concern, we recommend to use our proposed algorithm for calibrating governing equations of complex systems to data.

Chapter 6

Applications to Real Systems

So far we have shown that our optimisation algorithm works on a plethora of synthetic systems, in each of which the data making up the primary system was solely created by equations. In that sense we only analysed "pure" systems — with an additive noise at best. However, the real world is noisy and messy. Models which may be created to reflect real world-phenomena may only capture so much of the measurable dynamics. In this section we analyse how well our algorithm works on models built for real-world applications and using measured data as primary system. For that we experiment on the Lotka–Volterra equations [30], [31], the coupled logistic map [32], and a financial model introduced in [1]. Lastly, we apply our optimisation on data masked by complex systems and show that it is possible to reconstruct the masked signal; thus rendering such algorithms obsolete.

6.1 Lotka–Volterra System

In their works the Lotka–Volterra equations have been introduced as a system of two non-linear, first order, coupled ODEs describing the interplay between predators and prey [30], [31]. The basic idea of the model is simple and intuitive: if the number of predators is low, the prey can reproduce and the number of prey increases. However, the high number of prey makes it easier for predators to reproduce themselves and therefore increases the number of predators. The high number of predators in turn decreases the number of prey. This results in oscillating prey-predators numbers, as can be seen in red in figure 6.2a. This model was inspired by biological arguments and this oscillating behaviour can also be observed in nature [33]. It has been shown that the Lotka–Volterra equations behave chaotically and the generalised form of the equations form an attractor for more than two agents in the system [34]. The equations for this model reproducing the previously described behaviour are as follows [31]:

$$\begin{aligned} \dot{x}_1 &= a\,x_1 - b\,x_1\,x_2 \\ \dot{x}_2 &= -c\,x_2 + d\,x_1\,x_2 \end{aligned} \tag{6.1}$$

Here, x_1 describes the evolution of the prey population and x_2 describes the population of the predators. The gradients with respect to the parameters of upper equation for our algorithm are calculated as follows:

$$\frac{\partial \ell}{\partial a} = -\left(\dot{\chi}_1 - \dot{y}_1\right)\, y_1 \tag{6.2a}$$

$$\frac{\partial \ell}{\partial b} = +\left(\dot{\chi}_1 - \dot{y}_1\right)\, y_1 y_2 \tag{6.2b}$$

$$\frac{\partial \ell}{\partial c} = +\left(\dot{\chi}_2 - \dot{y}_2\right)\, y_2 \tag{6.2c}$$

$$\frac{\partial \ell}{\partial d} = -\left(\dot{\chi}_2 - \dot{y}_2\right)\, y_1 y_2 \tag{6.2d}$$

Again, we used the notation introduced in section 4.2.2, where y describes the secondary system and χ describes the data measured points making up the primary system. The measured data, acting as primary system, shows the abundance of Didinium (predator) and Paramecium (prey) and was obtain from the supplementary material from [35]. It is shown in red in figure 6.2a. The fit is shown and in black. While we see the parameters converging in figure 6.2b, we can clearly see that the found parameters do not describe the data adequately. Our algorithm seems to find a Lotka–Volterra structure in the data, however it cannot reconstruct the data correctly. Running a stability analysis, we at least observe out algorithm to be stable we, as we can see in figure 6.1. We ran the algorithm with five different initial conditions taking the fist data point in the first run, the second data point in the second run, and so on. For each different initial condition we found almost identical parameters.

This bad performance can have a few reasons: firstly, our assumption is, that the data can be completely described by equations 6.1. However, this assumption is very weak since the data stems from a real world system where additional effects are present, which are not accounted for in these equations. So with respect to the equations 6.1, we have to treat this data as very noisy. Secondly, we observed in figure 5.6 that even for synthetic systems with a bit of noise an evaluation length of 10^2 is not sufficient. With the dataset providing only 71 observations, this renders a successful application difficult.

Additionally, we can exclude that the optimisation fails because of specific properties of the Lotka–Volterra system, since we have shown in figures 5.1 that the optimisation works for the Lotka–Volterra system utilising synthetic data as primary system.

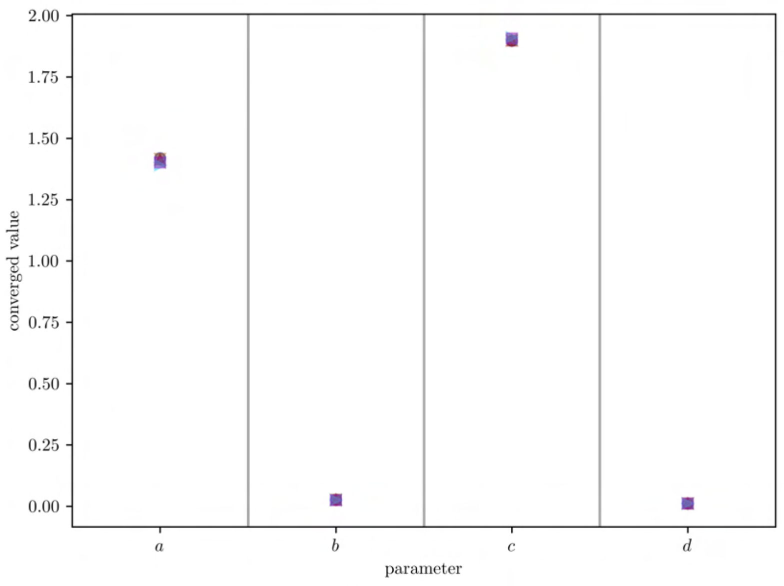
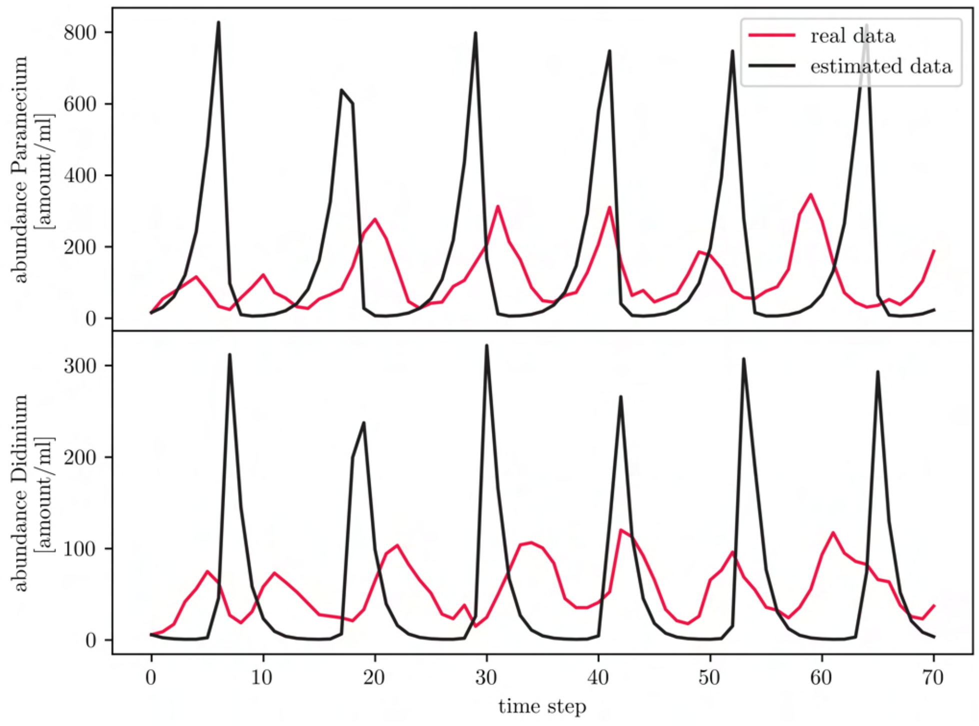
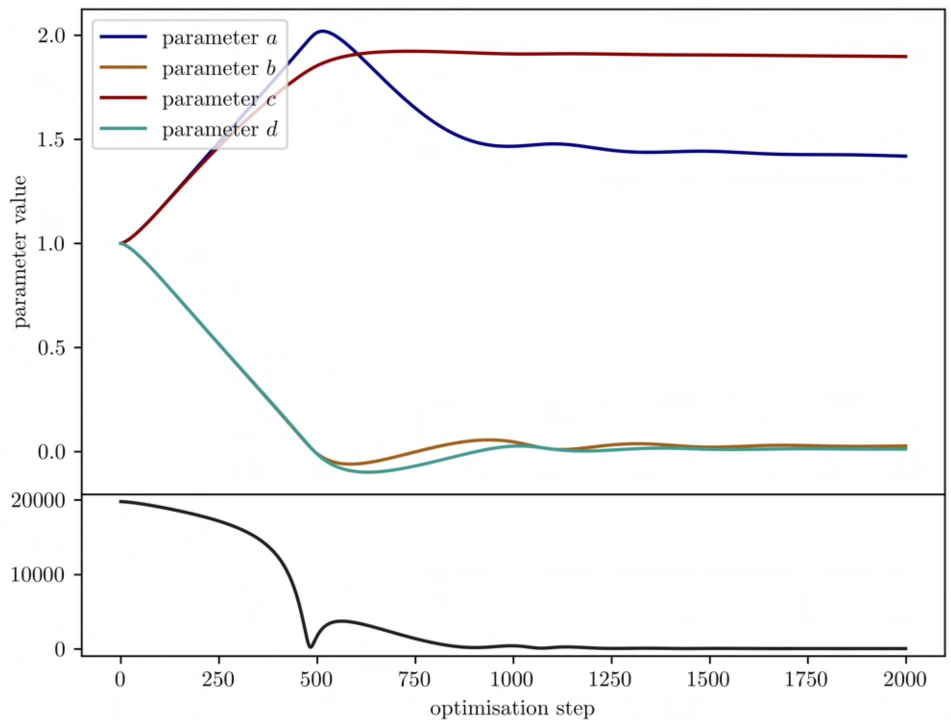


Figure 6.1: This figure shows the converged parameters for fitting the data from [35] on the Lotka–Volterra system using different initial conditions. For the first run we used the first data point as initial condition, for the second run we used the second, and so on. We used a coupling strength of $\alpha = 10$, and the evaluation length was the number of available data points. We used an integration step of $\Delta t = 0.5$, since in [35] it was implied that the data was measured every twelve hours. We chose to omit a legend, since all the data points lie so close on each other.



(a)



(b)

Figure 6.2: Figure (a) shows the measured data in red, which should in theory follow a Lotka–Volterra structure. The data was taken from the supplementary material in [35]. The black line shows our estimation of the data after the parameters converged. Figure (b) shows evolution of the parameters using our optimisation algorithm. We used the algorithm with a low coupling strength of $\alpha = 10$, an evaluation length of $l_\mathrm{e} = 71$ consisting of the whole available dataset. For better performance we set the order of the first gradient in equation 4.14 to 10^{-4}. The parameters converged to the following values: $a = 1.419$, $b = 0.0267$, $c = 1.8978$, and $d = 0.01158$. The precision was determined by the standard deviation of each parameter for the last hundred steps.

6.2 Financial Markets

In their work the authors presented a framework from which the form of governing equations can be reconstructed by utilising causality measures and measuring causality within discrete data [1]. However, they are not able to reconstruct the parameters for each term, which is where our algorithm becomes applicable.

In their article they considered the stock indices of six major world economies: USA (US), China (CN), Europe (EU), Japan (JP), Emerging Markets (EM), and Pacific excluding Japan (PX). They analysed the impact of the COVID-19 pandemic on the structure of the governing equations [1]. For that, they used daily logarithmic returns of the respective stock indices as data for their algorithm. For the baseline equations describing the pre-COVID-19 time, they used the data from October 2019 until January 2020 and found the following set of equations for the logarithmic returns [1], where we added the corresponding coefficient in grey:

$$\begin{aligned}
\dot{x}_{\text{US}} &= c_1\, x_{\text{US}} + c_2\, x_{\text{CN}} + c_3\, x_{\text{EU}} + c_4\, x_{\text{JP}} + c_5\, x_{\text{EM}} + c_6\, x_{\text{PX}} + c_7\, x_{\text{CN}}\, x_{\text{PX}} \qquad (6.3)\\
\dot{x}_{\text{CN}} &= c_8\, x_{\text{US}} + c_9\, x_{\text{CN}} + c_{10}\, x_{\text{EU}} + c_{11}\, x_{\text{EM}} + c_{12}\, x_{\text{PX}} + c_{13}\, x_{\text{US}}\, x_{\text{EU}}\\
\dot{x}_{\text{EU}} &= c_{14}\, x_{\text{US}} + c_{15}\, x_{\text{CN}} + c_{16}\, x_{\text{EU}} + c_{17}\, x_{\text{JP}} + c_{18}\, x_{\text{EM}} + c_{19}\, x_{\text{PX}} + c_{20}\, x_{\text{EU}}\, x_{\text{PX}}\\
\dot{x}_{\text{JP}} &= c_{21}\, x_{\text{CN}} + c_{22}\, x_{\text{JP}} + c_{23}\, x_{\text{PX}} + c_{24}\, x_{\text{US}}\, x_{\text{CN}} + c_{25}\, x_{\text{US}}\, x_{\text{EU}} + c_{26}\, x_{\text{US}}\, x_{\text{JP}}\\
&\quad + c_{27}\, x_{\text{US}}\, x_{\text{EM}} + c_{28}\, x_{\text{CN}}\, x_{\text{EU}} + c_{29}\, x_{\text{CN}}\, x_{\text{JP}} + c_{30}\, x_{\text{CN}}\, x_{\text{PX}} + c_{31}\, x_{\text{EU}}\, x_{\text{EM}}\\
&\quad + c_{32}\, x_{\text{EU}}\, x_{\text{PX}} + c_{33}\, x_{\text{EM}}\, x_{\text{PX}}\\
\dot{x}_{\text{EM}} &= c_{34}\, x_{\text{US}} + c_{35}\, x_{\text{CN}} + c_{36}\, x_{\text{EU}} + c_{37}\, x_{\text{JP}} + c_{38}\, x_{\text{JP}} + c_{39}\, x_{\text{PX}} + c_{40}\, x_{\text{JP}}\, x_{\text{PX}}\\
\dot{x}_{\text{PX}} &= c_{41}\, x_{\text{CN}} + c_{42}\, x_{\text{EU}} + c_{43}\, x_{\text{JP}} + c_{44}\, x_{\text{EM}} + c_{45}\, x_{\text{PX}} + c_{46}\, x_{\text{US}}\, x_{\text{PX}}
\end{aligned}$$

We applied our optimisation algorithm on upper equations using the introduced data: while we observed the parameters converging, we noted that the evaluation of the found set of equations does not describe the data as can be seen in figure 6.3. The first evaluation step seems to have the correct direction towards the true returns shown in red, however the evaluated set of equations converges towards zero after a few iterations for all dimensions. The found equations seem to not maintain the dynamics observed in the data. Smoothing out the real returns before the optimisation using a Wiener filter [36] did not lead to better performance; however, it changes the values the parameters converge to.

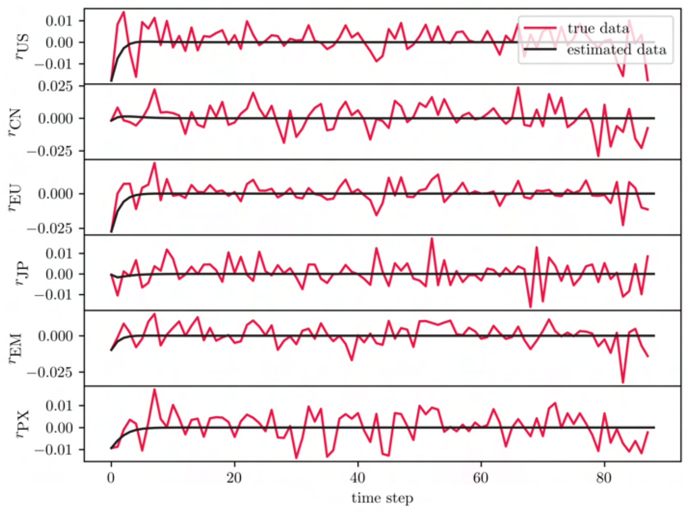


Figure 6.3: This figure shows the result of our optimisation algorithm applied on financial data. In red the historical logarithmic returns r of the respective stock market are shown. In black we show the evaluation of equations 6.3 with the parameters found in our optimisation. For our optimisation we used a coupling strength of $\alpha = 10$, and the evaluation length was the available number of data points: 88. As an integration step we used $\Delta t = 1$, since we are handling daily data. The data was obtained from the author in [1].

As a consistency check we try to fit random returns on the equations 6.3. Even though the returns are not normally distributed and exhibit a skewness and excess kurtosis [37], for the sake of simplicity we fit a Gaussian distribution onto the returns. Disregarding the correlation between assets, we sample from the six Gaussian distributions independently, obtaining a dataset made up from random returns. We then apply the same optimisation algorithm as with the real data, and try to fit equations 6.3 onto the random data. We would expect the parameters to diverge, since we would assume that the equations 6.3 cannot adequately describe the dynamics exhibited by the random data.

However, we still observe the parameters converging. The time series, with the parameters converged on the random data, can be seen in figure 6.4. Evaluating the system at the found parameters, we observe a similar behaviour as in figure 6.3: the estimated system cannot describe the "true" system. Therefore, at this point we must conclude that our optimisation algorithm is not able to find dynamics in financial data, better than it does in a random dataset, since in both cases the algorithm converges. As the random data is assumed to have no inherent dynamics[1], our optimisation algorithm also converges in the absence of dynamics. This finding does not give the results in 6.3 too much weight.

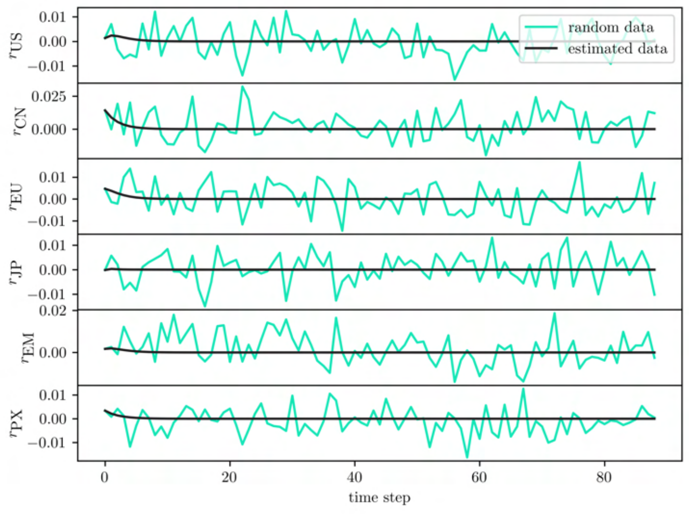


Figure 6.4: This figure shows the result of our optimisation algorithm applied on random data. In green we show the random logarithmic returns simulating their respective market. In black we show again the evaluation of equations 6.3 with the parameters found in our optimisation. We used the same parameters as in previous experiment: a coupling strength of $\alpha = 10$, an evaluation length of $l_e = 88$ for setting equal conditions, and an integration step of $\Delta t = 1$, since we are still handling daily data.

[1]Here, we disregard a possible dynamics stemming from only being able to generate pseudorandom numbers.

6.3 Reconstruction of Signals Masked by Complex Systems

Due to their nature, predicting complex system is an inherently difficult task. For this reason a lot of authors had the idea of hiding a transmitting signal in the seemingly random behaviour of a complex system: a lot of research was published utilising this idea [38], [39], [40], [41]. Even though each published algorithm works a bit differently and introduces a new nuance, most of them rely on the same basic idea: synchronise two chaotic systems with each other, so both the sender and the receiver can evaluate the same trajectory of a chaotic system. Then, the sender adds his signal on top of the chaotic system with a relatively low power and transmits it to the receiver. The receiver can then subtract the chaotic signal from the received total signal — since he knows exactly how the chaotic signal propagates due to synchronisation — and obtains the pure message. The parameter of the chaotic system then make up the key. This method seems secure, since chaotic systems ought to be unpredictable. However, we have shown in this work that it is possible to reconstruct the exact parameters of complex systems using their time series. Thus making them predictable. This works even if the time series is noisy, meaning that it carries an interesting signal on top of it.

For this work we rebuild the most basic idea of these masking algorithm presented in [38], since it is the most intuitive one. In their article they use the Lorenz system as working example. For the secondary system they induced synchronisation via the active-passive-decomposition introduced in section 3.1.1. For that, they replaced y_1 of the secondary system in $\dot{y}_2$ and $\dot{y}_3$ with the signal r. The signal r is the sum of the first coordinate x_1 of the primary system with the signal of interest s: $r = x_1 + s$. This results in the following secondary system, where the coupling is visualised in red:

$$\begin{aligned} \dot{y}_1 &= \sigma\, y_2 - \sigma\, y_1 \\ \dot{y}_2 &= -r\, x_3 + r\, \rho - y_1 \\ \dot{y}_3 &= r\, y_2 - \beta\, y_3 \end{aligned} \tag{6.4}$$

The idea is now to add the signal s onto the x_1 coordinate of the emitting Lorenz system to create the total signal r. This r is then used to synchronise the secondary system in equations 6.4 to the primary Lorenz delivering x_1. Once the secondary system is synchronised to the primary one, the estimate of the signal $\hat{s}$ can be reconstructed by:

$$\hat{s} = r - y_1 \tag{6.5}$$

As we have stated in section 3.1.1, this type of synchronisation only works when the primary and secondary systems are identical. Meaning that the set of parameters σ, ρ, and β need to be identical in the primary and secondary system for this type of transmission to work. Hence, the set of parameters σ, ρ, and β can be viewed as a key for this kind of "encryption"[2] [16].

We implemented this method with the following signal

$$s(t) = \frac{1}{10}\left(\sin(10\,t) + \sin(50\,t) + \cos(30\,t)\right) \tag{6.6}$$

We added the signal onto the x_1-coordinate of the Lorenz system with the following parameter vector in accordance to [38]: $\underline{\theta} = (\sigma, \rho, \beta)^\intercal = (16, 45.92, 4)^\intercal$. We then reconstructed the signal by using the Lorenz coordinate with signal to drive a secondary system with equations 6.4 and reconstructed the signal using equation 6.5. The results can be seen in figure 6.5: There we can see that it is possible to reconstruct the signal s from the driven signal. However, we also note that the reconstruction is not perfect and a reconstruction error is clearly visible in figure 6.5. This reconstruction error was expected [16]. Figure 6.5 also shows, that assuming a wrong set of parameters worsens the reconstruction of the signal. This means that the correct set of parameter is needed for the best reconstruction.

[2]We refrain from calling this method "encryption", since the data is only masked by a stronger signal and not altered by itself.

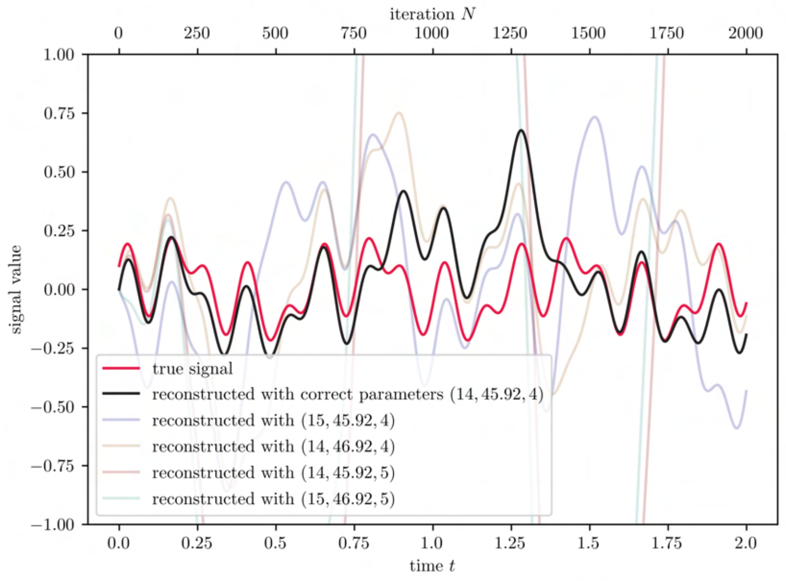


Figure 6.5: This figure visualises the functionality of the algorithm presented in [38], where the red line shows the original signal s, and the black line shows the reconstructed signal $\hat{s}$ using the correct parameters. The faint colours show the reconstruction using a wrong set of parameters, where the parameters are written in the form (σ, ρ, β). For each simulation we used a step size of $\Delta t = 10^{-3}$.

We applied our parameter reconstruction-algorithm on the problem stated in previous paragraph and reconstructed the following parameter vector from the data: $\underline{\hat{\theta}} = \left(\hat{\sigma}, \hat{\rho}, \hat{\beta}\right)^{\mathsf{T}} = (13.98, 45.87, 4.01)$, where we just reported two decimal places. A certain error in the reconstruction of the parameter is expected, since in this case the signal s acts as a noise for our algorithm working on pure data. In fact, we have a signal-to-noise ratio of 40.00 dB on the first coordinate of the Lorenz system. As we have seen in section 5.2.1, at that level of noise we expect an imprecision in the reconstruction of the parameters. We can see the results of the application of our algorithm in figure 6.6: there we find that for about a single Lyapunov time we can reconstruct the signal to the same precision as the original method intended. After about a Lyapunov time, our reconstruction fails. However, nothing stops us from repeating the optimisation and the subsequent prediction for each interval of length one Lyapunov time. Using that strategy, we should be able to reconstruct the signal as long as it persists.

We note that one possible caveat of reconstructing the parameters using this method may be that the chaotic systems has to be known. However, we point out that if the attractor is not known, the method presented in [1] can be used to reconstruct the governing equations. Additionally, if only one coordinate was transmitted, Takens' theorem [42] could be used to construct a shadow manifold using delay coordinates. This should hint at the correct complex system in question, since only a handful of them are published and used regularly.

The goal of this section was to show that in general, encryption algorithms based on the chaotic behaviour of systems are inherently unsafe. The assumption that complex systems are not predictable simply does not hold any more. Instead, we have shown that they are indeed predictable by reconstructing the parameters. Other algorithms presented in [39], [40], or [41] are more sophisticated than this example and introduce additional steps. However, as long as those steps are differentiable, or their derivative can be approximated reasonably, a respective loss function can be built. Applying the chain rule on the modified loss function should result in gradients leading to the correct parameters. Therefore, we strongly recommend to not use these kind of encryption algorithms outside of academic exercises.

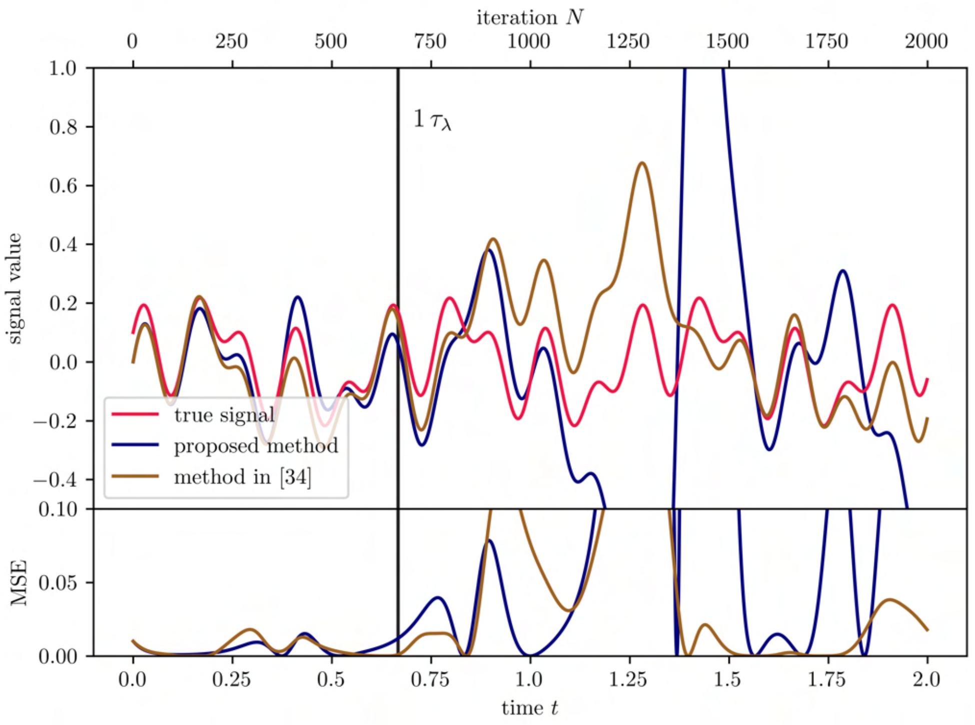


Figure 6.6: This figure compares the reconstruction of the red signal using the intended method presented in [38] against the reconstruction using our proposed algorithm. The original signal is described by equation 6.6. For our proposed algorithm we used a coupling strength of $\alpha = 10^2$ and the whole data as evaluation length: $l_\text{e} = 2\,000$. A step size of $\Delta t = 10^{-3}$ was used for this experiment. The upper plot shows the true signal and its reconstructions. The lower plot shows the mean squared error (MSE) of each method. The Lyapunov exponent of the Lorenz system with the parameters $\underline{\theta} = (\sigma, \rho, \beta)^\intercal = (16, 45.92, 4)^\intercal$ is $\lambda = 1.499 \pm 0.012$, so the black line shows one Lyapunov time.

Chapter 7

Summary & Outlook

The aim of this work was to develop an algorithm for estimating the parameters of complex systems using data. The posed problem was rather specific and asked whether it was possible to reconstruct the parameters, if the structure of the governing equations and a time series stemming from the target system were known. In this work we have shown that it is possible to reconstruct the parameters given those assumptions, and we presented a framework for applying this on arbitrary systems. However, the real world is not limited to the Lorenz system, which is why we needed to expand this method for application on arbitrary sytems.

For achieving this, we borrowed ideas from synchronisation and machine learning. Specifically, we coupled a secondary system, of which we can control the parameters, to a primary system being entirely described by the data. The secondary system encodes our assumptions on the structure of governing equations. We then modified the parameters of the secondary system using gradient descent-based algorithms to minimise the loss between the two systems. The values of the parameters minimising the loss are a good estimate of parameters describing the real data.

In this work we have shown that our algorithm is applicable to a plethora of different synthetic systems with predictabilities of a few Lyapunov times: figure 5.1. Additionally, we have shown that this algorithm also works in a high dimensional space with a large number of free parameters: figure 5.2. Up to a certain extent, our algorithm is robust against noise in the data and incorrect assumptions of the structure of the governing equations: figures 5.5 & 5.7. We are more stable than the comparable algorithm in [2]. Solely the applicability on real world-systems remains a challenge: figures 6.2 & 6.3. This can be mainly attributed to the fact that data stemming from real world systems is rarely described by a single set of governing equations. While equations provide a useful model for gaining valuable insights, they may fail to capture *all* aspects making up the dynamics. Lastly, we have shown that this algorithm can be used for reconstructing signals which are masked by complex systems and thus cracking these kinds of 'encryptions': figure 6.6.

At this point we want to mention that our group is currently performing research on reservoir computing [43] and such a calibration as explained here could be achieved using 'next generation reservoir computing' [44]. However, details about a comparison with regards to applicability and stability remain subject to further research.

When trying to strain this algorithm for deducing not only the coefficients but also the general form of governing equations, we found hints that the governing equations of complex system may not be unique: figure 5.10. This path needs more attention and future work needs to be done in order to analyse this finding in greater detail. Specifically, the different sets of equations describing the visually same attractor need to be evaluated with regard to commonly used metrics for classifying chaotic system. Additionally, we need to analyse different chaotic systems and need to examine whether the finding in section 5.2.2 is a more general fact or a peculiarity of the Lorenz system.

This work, together with [1], can be embedded in the greater topic of reconstructing governing equations from time series. Using [1] for deriving the structure of the governing equations and afterwards our algorithm for calibrating the coefficients can be a way to infer the dynamical equations describing the data at hand. This method could then be used across different disciplines for making interpretable, data-driven forecasts.

Appendix A

Complex Systems Studied in this Work

In this section we introduce the governing equations and the respective gradients for all systems studied in this work. For each system we introduce, we use the notation used throughout this work: with x we denote the primary system, y symbolises the secondary system coupled to a primary one, and χ represents the primary system made up from discrete data points. The gradients are calculated as described in section 4.2.2.

A.1 Dadras–Momeni System

The Dadras–Momeni system was introduced in [45] and is described by the following governing equations:

$$\begin{aligned} \dot{x}_1 &= x_2 - a\,x_1 + b\,x_2\,x_3 \\ \dot{x}_2 &= c\,x_2 - x_1\,x_3 + x_3 \\ \dot{x}_3 &= d\,x_1\,x_2 - e\,x_3 \end{aligned} \tag{A.1}$$

Its standard parameters are: $\underline{\theta} = (a, b, c, d, e)^\intercal = (3, 2.7, 1.7, 2, 9)^\intercal$. The gradient of the loss function with respect to the parameters can be calculated as follows:

$$\begin{aligned} \frac{\partial \ell}{\partial a} &= +2\,(\dot{\chi}_1 - \dot{y}_1)\;y_1 \\ \frac{\partial \ell}{\partial b} &= -2\,(\dot{\chi}_1 - \dot{y}_1)\;y_2\,y_3 \\ \frac{\partial \ell}{\partial c} &= -2\,(\dot{\chi}_2 - \dot{y}_2)\;y_2 \\ \frac{\partial \ell}{\partial d} &= -2\,(\dot{\chi}_3 - \dot{y}_3)\;y_1\,y_2 \\ \frac{\partial \ell}{\partial e} &= +2\,(\dot{\chi}_3 - \dot{y}_3)\;y_3 \end{aligned} \tag{A.2}$$

A.2 Four-Wing System

The Four-wing system was introduced in [46]. For our analysis we use a simplified version, which the authors present in their work:

$$\begin{aligned} \dot{x}_1 &= a\,x_1 + x_2\,x_3 \\ \dot{x}_2 &= b\,x_1 + c\,x_2 - x_1\,x_3 \\ \dot{x}_3 &= -x_3 - x_1\,x_2 \end{aligned} \tag{A.3}$$

Its standard parameters are: $\underline{\theta} = (a, b, c)^\intercal = (0.2, 0.01, -0.4)^\intercal$. The gradient of the loss function with respect to the parameters can be calculated as follows:

$$\begin{aligned} \frac{\partial \ell}{\partial a} &= -2\left(\dot{\chi}_1 - \dot{y}_1\right)\, y_1 && \text{(A.4)} \\ \frac{\partial \ell}{\partial b} &= -2\left(\dot{\chi}_2 - \dot{y}_2\right)\, y_1 \\ \frac{\partial \ell}{\partial c} &= -2\left(\dot{\chi}_2 - \dot{y}_2\right)\, y_2 \end{aligned}$$

A.3 Halvorsen System

The Halvorsen system was never published by itself, however it can be found in [47]. It is defined as follows:

$$\begin{aligned} \dot{x}_1 &= -a\, x_1 - 4\, x_2 - 4\, x_3 - x_2^2 && \text{(A.5)} \\ \dot{x}_2 &= -a\, x_2 - 4\, x_3 - 4\, x_1 - x_3^2 \\ \dot{x}_3 &= -a\, x_3 - 4\, x_1 - 4\, x_2 - x_1^2 \end{aligned}$$

Standardly, its parameter is set to $a = 1.89$. The gradient of the loss function with respect to its parameter can be calculated as follows:

$$\frac{\partial \ell}{\partial a} = +2\left(\dot{\chi}_1 - \dot{y}_1\right) y_1 + 2\left(\dot{\chi}_2 - \dot{y}_2\right) y_2 + 2\left(\dot{\chi}_3 - \dot{y}_3\right) y_3 \qquad \text{(A.6)}$$

A.4 Lorenz86 System

The Lorenz86 system was introduced in [48] and it is governed by the following set of equations:

$$\begin{aligned} \dot{x}_1 &= -a\, x_1 - x_2^2 - x_3^2 + a\, f && \text{(A.7)} \\ \dot{x}_2 &= -x_2 + x_1\, x_2 - b\, x_1\, x_3 + g \\ \dot{x}_3 &= -x_3 + b\, x_1\, x_2 + x_1\, x_3 \end{aligned}$$

Its standard parameters are: $\underline{\theta} = (a, b, f, g)^\intercal = (0.25, 4, 1.1, 8)^\intercal$. The gradient of the loss function with respect to the parameters can be calculated as follows:

$$\begin{aligned} \frac{\partial \ell}{\partial a} &= -2\left(\dot{\chi}_1 - \dot{y}_1\right)\left(-y_1 + f\right) && \text{(A.8a)} \\ \frac{\partial \ell}{\partial b} &= +2\left(\dot{\chi}_2 - \dot{y}_2\right) y_1\, y_3 - 2\left(\dot{\chi}_3 - \dot{y}_3\right) y_1\, y_2 && \text{(A.8b)} \\ \frac{\partial \ell}{\partial f} &= -2\left(\dot{\chi}_1 - \dot{y}_1\right) a && \text{(A.8c)} \\ \frac{\partial \ell}{\partial g} &= -2\left(\dot{\chi}_2 - \dot{y}_2\right) && \text{(A.8d)} \end{aligned}$$

Here, it is interesting to note that this is the only system where the gradients depend on the current state of the parameters as can be seen in equations A.8a & A.8c.

A.5 Lotka–Volterra System

The Lotka–Volterra system was introduced independently from each other in [30], [31]. It is defined in equations 6.1 and its gradient is defined in equations 6.2. As standard parameters we use: $\underline{\theta} = (a, b, c, d)^\intercal = (2, 0.4, 2, 0.8)^\intercal$.

A.6 Rössler System

The Rössler system was introduced in [49] and is described by the following equations:

$$\begin{aligned}\dot{x}_1 &= -x_2 + x_3 \\ \dot{x}_2 &= x_1 + a\,x_2 \\ \dot{x}_3 &= b + x_3\,(x_1 - c)\end{aligned} \tag{A.9}$$

Its standard parameters are $\underline{\theta} = (a, b, c)^\intercal = (0.1, 0.1, 14)^\intercal$. The gradient of the loss function with respect to the parameters can be calculated as follows:

$$\begin{aligned}\frac{\partial \ell}{\partial a} &= -2\,(\dot{\chi}_2 - \dot{y}_2)\,y_2 \\ \frac{\partial \ell}{\partial b} &= -2\,(\dot{\chi}_3 - \dot{y}_3) \\ \frac{\partial \ell}{\partial c} &= +2\,(\dot{\chi}_3 - \dot{y}_3)\,y_3\end{aligned} \tag{A.10}$$

A.7 Sprott System

The Sprott system was introduced in [50] and is described by the following set of equations:

$$\begin{aligned}\dot{x}_1 &= x_2 + a\,x_1\,x_2 + x_1\,x_3 \\ \dot{x}_2 &= 1 - b\,x_2^2 + x_2\,x_3 \\ \dot{x}_3 &= x_1 - x_1^2 - x_2^2\end{aligned} \tag{A.11}$$

Its standard parameters are: $\underline{\theta} = (a, b)^\intercal = (2.07, 1.79)^\intercal$. The gradient of the loss function with respect to the parameters can be calculated by:

$$\begin{aligned}\frac{\partial \ell}{\partial a} &= -2\,(\dot{\chi}_1 - \dot{y}_1)\,y_1\,y_2 \\ \frac{\partial \ell}{\partial b} &= +2\,(\dot{\chi}_2 - \dot{y}_2)\,y_1^2\end{aligned} \tag{A.12}$$

A.8 Thomas System

In his article, the author introduced a set of complex systems described by the following equation [51]:

$$\begin{aligned}\dot{x}_1 &= -b\,x_1 + f(x_2) \\ \dot{x}_2 &= -b\,x_2 + f(x_3) \\ \dot{x}_3 &= -b\,x_3 + f(x_1)\end{aligned} \tag{A.13}$$

In aforementioned article multiple choices for the function f are presented. In our work we limit ourselves to the following function: $f(x_i) = \sin x_i$. The parameter b is standardly set to $b = 0.21$. The gradient with respect to the parameter can be calculated as follows:

$$\frac{\partial \ell}{\partial b} = +2\,(\dot{\chi}_1 - \dot{y}_1)\,y_1 + 2\,(\dot{\chi}_2 - \dot{y}_2)\,y_2 + 2\,(\dot{\chi}_3 - \dot{y}_3)\,y_3 \tag{A.14}$$

A.9 Three-Scroll Unified Chaotic System

The Three-scroll unified chaotic system was introduced in [52] and is described by the following set of equations:

$$\begin{aligned}\dot{x}_1 &= a\,(x_2 - x_1) + d\,x_1\,x_3 \\ \dot{x}_2 &= -x_1\,x_3 + f\,x_2 \\ \dot{x}_3 &= -e\,x_1^2 + x_1\,x_2 + c\,x_3\end{aligned} \tag{A.15}$$

Its standard parameters are $\underline{\theta} = (a, c, d, e, f)^\intercal = (40, \frac{5}{6}, 0.5, 0.65, 20)^\intercal$. The gradient of the loss function with respect to the parameters can be calculated the following way:

$$
\begin{aligned}
\frac{\partial \ell}{\partial a} &= -2\,(\dot{\chi}_1 - \dot{y}_1)\,(y_2 - y_1) \\
\frac{\partial \ell}{\partial c} &= -2\,(\dot{\chi}_3 - \dot{y}_3)\,y_3 \\
\frac{\partial \ell}{\partial d} &= -2\,(\dot{\chi}_1 - \dot{y}_1)\,y_1\,y_3 \\
\frac{\partial \ell}{\partial e} &= +2\,(\dot{\chi}_3 - \dot{y}_3)\,y_1^2 \\
\frac{\partial \ell}{\partial f} &= -2\,(\dot{\chi}_2 - \dot{y}_2)\,y_2
\end{aligned}
\tag{A.16}
$$

Bibliography

[1] H. Ma, A. Haluszczynski, D. Prosperino, et al., "Identifying causality drivers and deriving governing equations of nonlinear complex systems," *unpublished — **submitted***, 2022.

[2] I. P. Mariño & J. Míguez, "An approximate gradient-descent method for joint parameter estimation and synchronisation of coupled chaotic systems," *Physics Letters A*, vol. 351, no. 4-5, pp. 262–267, 2006.

[3] S. Jafari, J. C. Sprott, V.-T. Pham, et al., "A New Cost Function for Parameter Estimation of Chaotic Systems Using Return Maps as Fingerprints," *International Journal of Bifurcation and Chaos*, vol. 34, no. 10, p. 1 450 134, 2014.

[4] C. Tao, Y. Zhang, & J. J. Jiang, "Estimating system parameters from chaotic time series with synchronization optimized by a genetic algorithm," *Physical Review E*, vol. 76, p. 016 209, 2010.

[5] S. Mukhopadhyay & S. Banerjee, "Global optimization of an optical chaotic system by Chaotic Multi Swarm Particle Swarm Optimization *[sic!]*" *Expert Systems with Applications*, vol. 19, pp. 917–924, 1 2012.

[6] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014.

[7] D. P. Kingma & J. L. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio & Y. LeCun, Eds., 2015.

[8] S. J. Reddi, S. Kale, & S. Kumar, "On the Convergence of Adam and Beyond," *Computing Research Repository*, 2019.

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

[10] C. Grebogi, E. Ott, S. Pelikan, et al., "Strange attractors that are not chaotic," *Physica D: Nonlinear Phenomena*, vol. 13, no. 1, pp. 261–268, 1984.

[11] A. Wolf, J. B. Swift, H. L. Swinney, et al., "Determining lyapunov exponents from a time series," *Physica D: Nonlinear Phenomena*, vol. 16, no. 3, pp. 285–317, 1984.

[12] W. Greiner, *Classical mechanics: systems of particles and Hamiltonian dynamics.* Springer, 2003.

[13] B. P. Bezruchko & D. A. Smirnov, *Extracting knowledge from time series: an introduction to nonlinear empirical modeling.* Springer, 2010.

[14] C. Skokos, "The Lyapunov characteristic exponents and their computation," *Lecture Notes in Physics*, pp. 63–135, 2009.

[15] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[16] D. Eroglu, J. S. W. Lamb, & T. Pereira, "Synchronisation of chaos and its applications," *Contemporary Physics*, vol. 58, no. 2, pp. 207–243, 2017.

[17] L. M. Pecora & T. L. Carroll, "Synchronization in chaotic systems," *Physical Review Letters*, vol. 64, no. 8, pp. 821–824, 1990.

[18] U. Parlitz, L. Junge, & L. Kocarev, “Synchronization-based parameter estimation from time series,” *Physical Review E*, vol. 54, no. 6, pp. 6253–6259, 1996.

[19] G. A. Leonov, *Strange Attractors and Classical Stability Theory*. St. Petersburg University Press, 2008.

[20] L. Barreira & Y. B. Pesin, *Lyapunov Exponents and Smooth Ergodic Theory*. American Mathematical Society, 2002.

[21] C. Runge, “Über die numerische Auflösung von Differentialgleichungen,” *Mathematische Annalen*, vol. 46, pp. 167–178, 1895.

[22] W. Kutta, “Beitrag zur näherungsweisen Integration totaler Differentialgleichungen,” *Zeitschrift für Mathematik und Physik*, vol. 46, pp. 435–453, 1901.

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

[24] E. Hairer, S. P. Nørsett, & G. Wanner, *Solving Ordinary Differential Equations I*. Springer, 1993.

[25] K. A. Atkinson, *An Introduction to Numerical Analysis*. John Wiley & Sons, 1989.

[26] J. H. Mathews & K. D. Fink, *Numerical Methods Using MATLAB*. Prentice Hall, 1999.

[27] E. N. Lorenz, “Predictability — a problem partly solved,” in *Predictability of Weather and Climate*, T. Palmer & R. Hagedorn, Eds. Cambridge University Press, 2006.

[28] R. Kieser, P. Reynisson, & T. J. Mulligan, “Definition of signal-to-noise ratio and its critical role in split-beam measurements,” *ICES Journal of Marine Science*, vol. 62, no. 1, pp. 123–130, 2005.

[29] P. Grassberger & I. Procaccia, “Characterization of Strange Attractors,” *Physical Review Letters*, vol. 50, no. 5, pp. 346–349, 1983.

[30] A. J. Lotka, *Elements of Physical Biology*. Williams & Wilkins Company, 1925.

[31] V. Volterra, “Variations and Fluctuations of the Number of Individuals in Animal Species living together,” *ICES Journal of Marine Science*, vol. 3, pp. 3–51, 1 1928.

[32] A. L. Lloyd, “The Coupled Logistic Map: A Simple Model for the Effects of Spatial Heterogeneity on Population Dynamics,” *Journal of Theoretical Biology*, vol. 173, pp. 217–230, 1995.

[33] B. G. Veilleux, “An analysis of the predatory interaction between *Paramecium* and *Didinium*,” *Journal of Animal Ecology*, vol. 48, no. 3, pp. 787–803, 1979.

[34] J. A. Vano, J. C. Wildenberg, M. B. Anderson, et al., “Chaos in low-dimensional Lotka–Volterra models of competition,” *Nonlinearity*, vol. 19, pp. 2391–2404, 2006.

[35] G. Sugihara, R. May, H. Ye, et al., “Detecting Causality in Complex Ecosystems,” *Science*, vol. 338, pp. 496–500, 2012.

[36] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press, 1949.

[37] R. Cont, “Empirical properties of asset returns: Stylized facts and statistical issues,” *Quantitative Finance*, vol. 1, pp. 223–236, 2 2001.

[38] A. V. Oppenheim, G. W. Wornell, S. H. Isabelle, et al., “Signal processing in the context of chaotic signals,” in *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, 1992, pp. 117–120.

[39] Ö. Morgül & M. Feki, “A chaotic masking scheme by using synchronized chaotic systems,” *Physics Letters A*, vol. 251, pp. 169–176, 3 1999.

[40] J. Pan, Q. Ding, & B. Du, “A new improved scheme of chaotic masking secure communication based on Lorenz system *[sic!]*” *International Journal of Bifurcation and Chaos*, vol. 22, no. 5, pp. 169–176, 2012.

[41] R. Martínez-Guerra, J. J. Montesinos García, & S. M. Delfín Prieto, “Secure communications via synchronization of Liouvillian chaotic systemes,” *Journal of the Franklin Institute*, vol. 353, pp. 4384–4399, 17 2016.

[42] F. Takens, “Detecting strange attractors in turbulence,” in *Dynamical Systems and Turbulence, Warwick 1980*, D. Rand & L.-S. Young, Eds., Springer, 1981, pp. 366–381.

[43] M. Lukoševičius & H. Jäger, “Reservoir computing approaches to recurrent neural network training,” *Computer Science Review*, vol. 3, pp. 127–149, 3 2009.

[44] D. J. Gauthier, E. Bollt, A. Griffith, et al., “Next generation reservoir computing,” *Nature Communications*, vol. 12, 2021.

[45] S. Dadras & H. R. Momeni, “A novel three-dimensional autonomous chaotic system generating two, three and four-scroll attractors,” *Physics Letters A*, vol. 373, pp. 3637–3642, 40 2009.

[46] Z. Wang, Y. Sun, B. J. van Wyk, et al., “A 3-D four-wing attractor and its analysis,” *Brazilian Journal of Physics*, vol. 39, no. 3, pp. 547–553, 2009.

[47] J. C. Sprott, *Elegant Chaos [—] Algebraically Simple Chaotic Flows.* World Scientific Publishing, 2010.

[48] E. N. Lorenz, “Atmospheric models as dynamical systems,” in *Perspectives in Nonlinear Dynamics: 28 - 30 May 1985 Naval Surface Weapons Center*, M. F. Shlesinger, R. Cawley, A. W. Saenz, et al., Eds., World Scientific Publishing, 1986.

[49] O. E. Rössler, “An equation for continuous chaos,” *Physics Letters A*, vol. 57, pp. 397–398, 5 1976.

[50] J. C. Sprott, “A dynamical system with a strange attractor and invariant tori,” *Physics Letters A*, vol. 378, pp. 1361–1363, 20 2014.

[51] R. Thomas, “Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis, “labyrinth chaos”,” *International Journal of Bifurcation and Chaos*, vol. 9, no. 10, pp. 1889–1905, 1999.

[52] L. Pan, W. Zhou, J. Fang, et al., “A New Three-Scroll Unified Chaotic System Coined,” *International Journal of Nonlinear Science*, vol. 10, no. 4, pp. 462–474, 2010.

Ehrenwörtliche Erklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen, als die in der Arbeit angegebenen, Quellen und Hilfsmittel verwendet zu haben. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Ort, Datum Davide Prosperino