Informatics

Chaos, Complexity and Neural Network Time Series Predictions

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

Sebastian Raubitzek, BSc. MSc.
Registration Number 01011689

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dr. Andreas Rauber
Second advisor: Dipl.-Ing. Mag. Dr.techn. Thomas Neubauer

The dissertation has been reviewed by:

Claudia Plant	Holger Kantz

Vienna, 24th April, 2023

Sebastian Raubitzek

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Declaration of Authorship

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Signed: ______________________________

Date: ______________________________

Kurzfassung

Neuronale Netzwerke und maschinelles Lernen sind erfolgreich-verwendete Werkzeuge um Zeitreihen zu analysieren und vorauszusagen. Neuronale Netze wie auch maschinelles Lernen sind datenbasierte Algorithmen, d.h. sie erfordern Datensätze um ein bestimmtes Verhalten zu erlernen, und sind daher stark abhängig von den zugrundeliegenden Datensätzen. In den Agrar- oder Umweltwissenschaften sind Datensätze oft sehr begrenzt, d.h. sie wurden nur über eine kurze Zeit (z.B. wenige Jahre) aufgenommen oder sind lückenhaft. Zusätzlich ist die Vorhersage dieser Datensätze oft sehr schwierig, da viele unterschiedlich Faktoren wie z. B. das Wetter diese Datensätze verkomplizieren und ihnen daher eine gewisse Unvorhersehbarkeit innewohnt.

Das Problem fehlender oder weniger Datenpunkte kann durch die Verwendung von Interpolationstechniken überwunden werden. Hierzu gibt es verschiedene Ansätze wie z. B. lineare, polynomiale oder fraktale Interpolationstechniken.

Zusätzlich kann die Vorhersage von schwierigen Datensätze verbessert werden, indem man einen Ensemble-Ansatz wählt, d.h. die Vorhersagen verschiedener Modelle und/oder Algorithmen werden kombiniert.

Die hier präsentierten Ansätze verbinden Interpolationstechniken, datenbasierte Ensemble-Methoden und Chaos-/Komplixtätsmetriken um die Vorhersage schwieriger Zeitreihendaten zu verbessern.

Hierzu wurden zwei, teils neue, Interpolationstechniken verwendet. Die erste ist eine Hurst-Exponent-basierte Fraktal-Interpolation, welche die Fluktuationen von Zeitreihen berücksichtigt und fortsetzt. Die zweite ist eine stochastische Interpolationsmethode, welche das Konzept der Brownschen Brücken mit rekonstruierten Phasenräumen chaotischer Systeme verbindet, um möglichst glatte Phasenraumtrajektorien zu erzeugen.

Die entwickelte Ensemble-Technik beruht auf zufällig parametrisierten neuronalen Netzen, deren Vorhersagen dann bezüglich der Komplexität und Chaotizität der zugrunde liegenden Daten gefiltert werden. Hierbei wurden neuronale Netze mit einer LSTM (**L**ong **s**hort-**t**erm **M**emory) Architektur gewählt.

Die Resultate zeigen, dass die Vorhersagen mittels neuronaler Netzen durch die verwendeten Interpolationstechniken verbessert werden können. Weiters können Vorhersagen von neuronalen Netzen basierend auf deren Komplexität und Chaotizität gefiltert werden.

Abstract

Many of today's most successful approaches for predicting time series data use machine and/or deep learning approaches such as different neural network architectures. These approaches strongly depend on the data available to train the employed algorithm. For, e.g., agricultural or environmental relevant applications, long-term data sets are rare and often sparsely sampled. Apart from that are often difficult to predict because of numerous influences that affect these data sets. Thus, these data sets have an inherent randomness to them.

The problem of sparsely sampled data can be overcome by employing different interpolation techniques, such as linear, polynomial, or fractal interpolation. On the other hand, the inherent randomness of difficult time series data can be treated by employing ensemble predictions.

This research attempts to combine interpolation techniques and neural network ensemble predictions and further improve these combined approaches by taking into account the complexity and chaotic properties of the underlying data.

The presented research introduces two interpolation techniques. One is a Hurst-exponent-based fractal interpolation considering the fluctuating nature of stochastic time series data. And the other one is a stochastic interpolation method that considers the reconstructed phase space properties of chaotic time series to produce an interpolation with a rather smooth phase space trajectory.

Further, this research presents an ensemble technique that takes into account the complexity and/or reconstructed phase-space properties of the data under study. This is achieved by randomly parameterizing a multitude of long short-term memory neural networks (LSTM), having them produce an autoregressive prediction, and afterward filtering this multitude of different predictions based on their signal complexity and/or reconstructed phase space properties.

First, the results show that neural network time-series predictions can be improved by employing the discussed interpolation techniques. And second, predictions can effectively be filtered based on their inherent complexity and phase space properties to improve ensemble predictions.

Acknowledgements

I want to thank Thomas Neubauer for his support throughout this research.

I would also like to offer my special thanks to Andreas Rauber for all the help in finishing this thesis.

I'm also particularly grateful for being part of DILAAG and the company of the other DILAAG Ph.D. candidates.

Contents

Chapter 1

Introduction

The rise of artificial intelligence, i.e., machine learning and deep learning, motivates many researchers to make predictions and analysis based on historical data using these methods rather than employing mechanistic expert models. The reason for making predictions in the first place is to answer important questions, e.g., future population estimates, predicting epileptic seizures, estimating future stock market prices. The outcomes of these predictions are encouraging, e.g., in solid-state physics, first-principle calculations can be sped up, [1], or solar radiation can be predicted using machine learning methods, [2]. In biology, machine learning can be applied to, e.g., genomics, proteomics, and evolution, [3]. When it comes to agriculture, one can use machine learning to predict yields and give estimates on the nitrogen status, [4]. Also, in medicine, one can apply machine learning to improve diagnosis using collected information from the past, [5]. In finance, the applications range from risk management or the construction of portfolios to designing and pricing securities, [6].

Though these approaches work pretty well in some research areas, in some others, depending on the available data and the system's complexity, these techniques seem not to work at all. Though it can not be generalized what sort of data is necessary to perform machine learning successfully, the two main reasons, assuming one has optimized the algorithm for the regarded task, for machine learning to fail are:

(i) Lack of data; meaning that the overall amount of data is not enough to train a model properly.

(ii) Lack of good data; meaning that despite a sufficient amount of data available, the inherent information is not enough to achieve good results.

Complexity measures also feature a broad spectrum of applications. *R/S analysis* (i.e. rescaled range analysis) and the corresponding *Hurst exponent* were invented for determining optimum dam sizes for the river of Nile, [7]. Since then, the Hurst exponent was used for various applications such as financial market analysis, [8], characterizing potentials in particle physics, [9], or to identify chaotic behaviors in the flow of alloys, [10]. The Hurst exponent is not the only complexity measure capable of characterizing time series and identifying random behavior. The *fractal dimension*, closely related to the Hurst exponent, is also a valuable tool. In [11], the fractal dimension is used to characterize environmental pollution. One can also define several types of entropy that prove useful for identifying complex/chaotic behavior, e.g., approximate entropy, as used in [12].

Though there are numerous applications of these methods, a combination of artificial intelligence and complexity measures is not that common yet. There are some examples where this has been done, e.g. a hybrid approach of fuzzy logic, fractal dimension, and a neural network is used to make predictions of time series in [13]. In [14], several indices of emerging economics are examined using the Hurst exponent and the fractal dimension of a time series, thus indicating a long term memory in the time series data, and afterward are forecast using machine learning methods. Ref. [15] examines several different time series using R/S analysis to show the inherent long-term memory of the data. The next step is to forecast the time series using recurrent neural networks (RNNs). The results show that even though R/S analysis suggests a persistent behavior, the RNN cannot forecast real-life time series with high accuracy. Given these examples, there is still potential for future applications such as time series analysis/prediction in agriculture.

Another related discipline is chaos theory, or the study of nonlinear dynamics. Here, the concept of reconstructed phase spaces of the system's dynamics can be linked with machine and deep learning approaches, as done in [16], where classical machine learning approaches and neural networks are used to reconstruct a multi-dimensional phase space from a univariate signal.

The previously mentioned phase space reconstruction, which is based on Takens' theorem, [17], can be used for time-delayed neural networks, where the phase space embedding of a time series is used to adapt the architecture of a neural network to the reconstructed phase space, as done in [18], for predicting stock market data, and in [19], for predicting wind speed. In Ref. [20] a phase space reconstruction is used to extract features then to be used to detect ventricular fibrillation with a neural network.

The presented thesis aims to show ideas on how to combine machine learning, measures of signal complexity, and reconstructed phase spaces. It is part of **DILAAG** (Digitalization and Innovation Laboratory in Agricultural Sciences). Therefore, the presented thesis has an aspect of applying the developed methods to agriculture and interdisciplinary research between the University of Natural Resources and Life Sciences Vienna, the University of Technology Vienna, and the University of Veterinary Medicine Vienna.

1.1 Fundamental of the Problem

For machine learning to make good predictions, data of good quality is necessary. Here a qualitative overview of the three main points that characterize good time series data for machine learning applications are given:

(i) **Long data sets:** A long data set is usually an indicator for machine learning algorithms to perform well.

(ii) **Fine-grained data sets:** As a rule of thumb, machine learning algorithms perform better the more fine-grained the data is. A very fine-grained data set features a lot of information of inherent interactions of the observed system, and therefore a machine learning algorithm can learn these interactions and behaviors. Whereas data sets that are not fine-grained may provide a more random behavior.

(iii) **Persistency/Antipersistency/Periodicity:** There is a difference between a random process and a process that features persistency, antipersistency, or periodicity/seasonality. Where a random process cannot be forecast effectively, a process that features persistency, antipersistency, and/or periodicity/seasonality can (as a rule of thumb) be forecast more effectively.

As this research is part of DILAAG, there is an aspect related to Austrian agriculture in the presented research. Fine-grained and long-time series data sets in Austrian agriculture are usually unavailable. The reasons for this are: [1]

- **Missing fine-grainedness:**

  Many data sets are collected annually or monthly; thus, many interactions are meaned out because of the missing fine-grainedness. Further, many measurements are not taken equidistantly.

- **Experimental design in agricultural-related research:**

  Usually, experiments do not run for more than five years[2]. Further, as all measurements in agriculture are expensive, the researchers cannot provide coherent fine-grained measurements for their experiments, be it daily, weekly, or even monthly.

Exemplary for the problems mentioned above are the annual wheat yields data set and the annual maize yield data set, which are described in Sections 7.6 and 7.7, respectively.

Complex and non-linear behavior is part of many real-life systems. The proposed research focuses on merging ideas from chaos theory and complexity measures with machine/deep learning to perform time series predictions. This task is manifold and involves several concepts that cannot be dealt with during the duration of the proposed research, see Section 1.4. To ensure the presented research is focused and specifically targeted, the following hypothesis and research question were employed as guidelines throughout the whole research process.

Hypothesis

Real-life systems, as occurring in agriculture, are highly complex and non-linear, therefore non-linear dynamics and the corresponding complexity have an influence on the predictions of those systems [22–25][3].

[1]There are no references to support the following list. Instead, this list results from many discussions and meetings with other researchers who are part of DILAAG.

[2]Of course, there are exceptions to this, e.g., the eternal rye experiment, [21].

[3]Note that, though the hypothesis is based on the cited books and articles, the hypothesis itself does not occur in the references. Instead, these references provide evidence that one can find chaos and complexity in various disciplines and systems.

Research Question

To what extent can measures of signal complexity, the entropy of time series data, and the concept of reconstructed phase spaces be used to improve machine and deep learning predictions for univariate and short/sparsely sampled time series data?

1.2 Solutions and Novel Research Contributions

The presented research is an attempt to give an answer to the research question and solve the stated problem. Thus we sum up and dissect the stated problem and the research question into several parts and answer them separately.

1.2.1 Sparsely Sampled Data and Data Scarcity:

When it comes to sparsely sampled time series data and, in general, data scarcity, we suggest using interpolation techniques. However, we need to mention that interpolation is not the only way to deal with sparsely sampled and short univariate time series data. In general, the concept of increasing the amount of data from a certain given amount of data is referred to as data augmentation. The basic idea is to generate slightly altered/*augmented* or synthetic data, which is then added to the original data to increase the overall amount of training data to improve a machine learning algorithm's capability of learning the data's inherent information. Here we take into account the work done by Semenoglou et al. [26], where the researchers test various data augmentation techniques for improving the accuracy of neural networks to predict univariate time series data. The nine data augmentation techniques employed by the researchers are vertical flipping, horizontal flipping, random noise injection, time series combinations, magnitude warping, time series interpolation, bootstrapping, time series generation, and upsampling. The general conclusion of the researchers is that the smaller the data set, the more the employed data augmentation matters. However, given the multitude of employed data augmentation techniques, the researchers cannot state which data augmentation is best in general, but rather state that the difference in the performance of the different data augmentation techniques can be attributed to the specific characteristics of the data under study and the corresponding synthetic data created via data augmentation. However, in the presented work, we restrict ourselves to only one data augmentation technique, i.e., time series interpolation. However, we discuss different interpolation techniques and emphasize those that consider the specific characteristics of the data under study. Thus, this research presents two interpolation techniques, considering the complexity and/or underlying dynamics of the data under study.

- **A Hurst Exponent Based Fractal Interpolation Method:**

  This technique is based on the *fractal curve fitting* method developed by [27]. The basic idea is to use a fractal interpolation to increase the fine-grainedness of the data. Further, the additional data points should fit the fluctuations of a given sub-interval of the data to replicate multi-fractal data. This is achieved by measuring the Hurst exponent of a sub-interval of a time series and then generating several hundred different randomly parameterized fractal interpolations. Here, the one with the Hurst exponent closest to the original one is used as the final interpolation.

  This method is described and tested to what degree it can improve time series predictions using LSTM neural networks in [28].

  Section 5.1 introduces the concept of a fractal interpolation and Chapter 8 presents the results of combining the developed interpolation technique with neural network time series predictions.

- **An Attractor-Based Stochastic Interpolation Method:**

  After using an interpolation technique based on the stochastic and multi-fractal nature of real-life time series data, i.e., the previously mentioned fractal interpolation approach, the author aimed to create an interpolation technique that considers the phase-space properties of time series data. Here, phase space properties means finding a smoothly interpolated trajectory in phase space. To achieve this, time series are interpolated using multipoint fractional Brownian Bridges, [29]. Several hundred interpolations with varying Hurst exponents are generated. These randomly initialized interpolations then serve as a population for a genetic algorithm. Piece by Piece, this algorithm picks interpolation intervals based on the smoothness of the corresponding reconstructed phase space curve. The smoothness is estimated using the variance of second derivatives along the reconstructed phase space trajectory. For simplicity, we refer to the developed method as *PhaSpaSto* interpolation, which is an abbreviation for **pha**se-**spa**ce-trajectory-smoothing **sto**chastic interpolation. The results show that this method is applicable to the Lorenz system as it finds interpolations very close to the actual data points. Further, it was tested on non-model data where it provides similar results as in the case of the Lorenz system, i.e., the phase space trajectory was smoothed out, and other interpolation methods are outperformed on reconstructing missing data points for some data sets, [30].

The developed interpolation technique is described in Section 5.2 and the results are shown and discussed in Chapter 10. Further, it's applicability to neural network time series predictions is discussed in Chapter 11 and Ref. [31].

1.2.2 Autoregression and Machine Learning for Difficult Time Series Data:

Given that one has found a good fitting machine learning model for a time series data, i.e., the train and test fits have comparatively high accuracy/a low root mean squared error (RMSE) and the corresponding plots suggest that the model has learned the inherent behavior of the data. Then this model may not be able to predict the learned data autoregressivly, meaning multi-step ahead predictions where the output is used as input for the next prediction will not reconstruct either the correct curve or the correct behavior. Figure 1.1 illustrates this problem. Even though the neural network visibly learned the training and test data of the fractal interpolated data set, the autoregressive result, i.e., single step ahead predictions and then taking the output as the next input, is still far off and does not at all reproduce the visible behavior of the original data. This means the yellow and the green lines, i.e., the train and test fits are both close to the original data and visibly reproduce the behavior of the learned data set. The red line, i.e., the autoregressive prediction, merely gives the mean.[4] Also, we consider the annual maize yields data set to be a challenging data set to be predicted correctly for three reasons: First, it is a very short time series, with only 57 data points and second, the annual maize yield is affected by various factors, such as the weather, genetic improvements of the plants, etc.

[4]Note that we will not discuss the interpolation of this time series here because this figure serves only to illustrate the problem of autoregression.

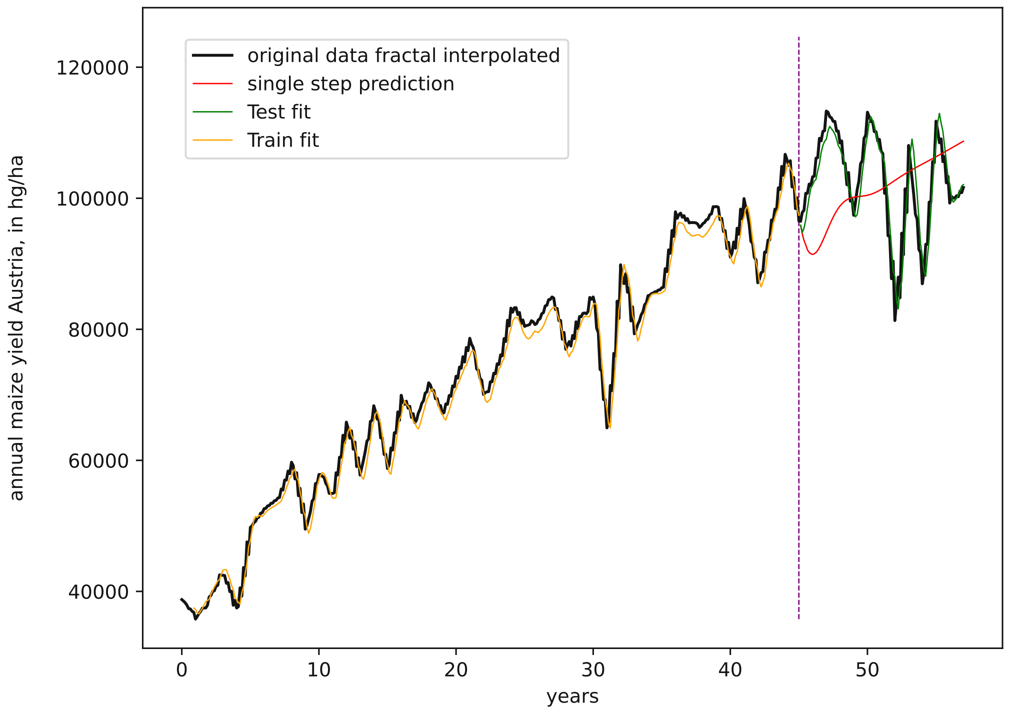


FIGURE 1.1: Plots for the fractal interpolated annual maize yields in Austria data set. This data set is discussed in Section 7.7. Here a long short term memory (LSTM) neural network with one hidden layer and 10 neurons in the hidden layer was used.

A solution to this is given by generating randomly parameterized neural networks. I.e., we do not observe the neural network's performance but instead have many such randomly parameterized neural networks produce autoregressive predictions of the data. Next, we take this multitude and filter the predictions according to the original data's complexity and reconstructed phase space characteristics. Thus, the following two approaches were developed:

- **Randomly Parameterized LSTM Neural Networks and Complexity-Based Prediction Filters**

  Instead of parameterizing a neural network to predict time series as precisely as possible autoregressively, the author developed a method using randomly parameterized neural networks to generate many predictions. In the next step, these predictions are then filtered using measures of signal complexity to build an ensemble consisting of forecasts that fit the complexity of the training data as close as possible. The results show that this method can drastically improve the accuracy of the so generated random ensemble.

The randomly parameterized Neural Networks are described and first applied, together with the complexity filters, in [32]. Chapter 6 describes all used neural network architectures, and Chapter 9 shows the experimental results and explains the whole approach.

- **Predictions and Filters Based on Phase Space Properties**

  The previously mentioned attractor-based stochastic interpolation technique (PhaSpaSto interpolation) is then tested with the (also previously mentioned) developed randomly parameterized neural network approach. Further, instead of filtering the predictions based on their inherent signal complexity, the predictions are then filtered based on the variance of second derivatives along a phase space trajectory. The results show that the variance of second derivatives along a phase space trajectory can effectively filter predictions. Further, the attractor-based interpolation technique can improve the predictability of univariate time series data. The applicability of the variance of second derivatives along a reconstructed phase space trajectory for filtering ensemble predictions is discussed and first applied in [31].

  Chapter 6 describes the used neural networks and Chapter 11 shows the experimental results and describes the whole approach. Further, these ideas are published in [31].

However, we need to point out another way of dealing with the problem of autoregression and neural networks. This would be to train a neural network not based on its data point fits but on the performance of an autoregressive prediction of both, training and test data set. This possible solution was discarded for two reasons. First, this research aims to entwine neural networks and measures of signal complexity and build on existing train and test frameworks to do so, but not to provide a completely new train and test framework for neural networks and machine learning to deal with time series data. Further, the latter would require restructuring existing machine learning frameworks such as *TensorFlow* or *scikit-learn* on a fundamental level of the provided infrastructure.

1.3 Structure of this Thesis

This thesis is split into four main parts:

Part I. Methodology and Related Work, which contains an exhaustive list of related work which contributed to the presented research (Chapter 2). Further, it contains the mathematical foundations of the employed measures of signal complexity (Chapter 3), the conceptual framework of reconstructed phase spaces (Chapter 4), and the employed and developed interpolation techniques (Chapter 5). Finally, this part contains all machine learning tools used for predicting and analyzing time series data with the corresponding error metrics (Chapter 6) and a discussion of all data sets used in this thesis (Chapter 7).

Part II. Applications and Results, contains the applications and the corresponding results of the developed techniques and ideas. This part is arranged to fit already published articles thematically. Here, the first presented applications and results are on the applicability of the employed fractal interpolation for improving neural networks time series predictions (Chapter 8). The next chapter then presents the results of the developed random LSTM neural networks approach and the corresponding prediction filters based on measures of signal complexity (Chapter 9). This is followed by the experiments on the applicability of the developed PhaSpaSto interpolation (Chapter 10) and the corresponding phase-space-based prediction approach (Chapter 11).

Part III. Discussion and Conclusion is the last part of the actual thesis. We first collect our best prediction results and compare them to benchmark results from the literature (Chapter 12). Next we discuss and summarize all featured results and experiments (Chapter 13). At the end of the thesis, we provide a conclusion (Chapter 14) and a list of the author's publications that are part of or influential to the presented thesis in Chapter 15.

Part IV. Appendix collects additional results, tables and plots to the featured experiments to keep the main text focused.

1.4 Limitations

The author wants to address some limitations of this work. First of all, this thesis, as the TU Wien informatics Ph.D. college guidelines encourage students to do, is heavily based on the author's published articles and, wherever possible, provides additional results and, to this date, not published results. Further, the presented research solved some issues but opened up many more questions (As research tends to do), which could not be addressed in the duration of the Ph.D. program. Therefore, to further ensure the honesty of this research, the author wants the following points to be mentioned beforehand.

This list may serve as a collection of ideas for new research and as evidence that incoherences that the reader may come across are thought through but could not be addressed due to the already extensive length of this thesis and the corresponding publications.

- **Computational resources:**

One of the biggest issues to get this work done was the access to Computational resources. Though the author had access to the TU Wien GPU cluster throughout the whole duration of this project, which the author is very grateful for, many aspects and permutations could not be shown simply due to the fact that it would take more time to calculate all of the afterward mentioned ideas.

- **Chronologic order of the developments and publications:**

  As Part II. Applications and Results features many already published results and approaches, the reader has to take into account that Chapters 8 and 9 are preliminary results to Chapters 10 and 11 and thus the later Chapters feature improved ideas and analysis. Due to, as previously mentioned, limited computational resources, the preliminary results were not reconstructed in the fashion of the later results.

- **Incoherence in data set choices:**

  As mentioned before, as many ideas take a long time to be calculated, this research suffers from non-infinite computational resources. Thus the author could not perform all experiments with all featured data sets. Therefore the author sticks to the published articles and, wherever possible, provides additional results, discussions, and arguments for the choices of data sets.

- **A variety of interpolation techniques to choose from:**

  As mentioned in Section 1.2.1, this thesis deals with interpolation techniques for time series data. There exists a huge variety of different interpolation techniques to choose from. However, as mentioned earlier, due to limited computational resources, we could not test all possible interpolation techniques and different numbers of interpolation points.

- **Limitations on the discussed machine learning algorithms:**

  As mentioned in Section 1.2.2, this thesis deals with machine learning and, to be precise, with neural network time series predictions. Many different neural network cell architectures and/or comparable machine learning algorithms are capable of predicting time series data. However, as mentioned earlier, due to limited computational resources, we could not test all machine learning techniques and employ LSTM neural networks throughout this work. However, we still provide comparisons to basic implementations of simpler recurrent neural network architectures.

Part I. Methodology and Related Work

Chapter 2

Related Work

This chapter lists publications that are related to the presented work and thus influenced the presented thesis. Many of the following publications feature combined approaches of machine learning and measures of signal complexity. Another topic covered by the following publications is "Combined approaches of machine learning and phase space reconstructions." All publications not fitting in the two previously mentioned topics provide research that heavily influenced the presented thesis in one way or another and thus will be put into context in this chapter. In the later parts of this thesis, the reader will encounter dozens of other publications that, though influencing this thesis, are a more technical influence as they provide the employed techniques. In contrast, the listing in this chapter mostly deals with articles that provide approaches of combined techniques and/or rather niche ideas.

This thesis aims to provide combinations of machine learning, measures of signal complexity, reconstructed phase spaces, and interpolation techniques to improve time series prediction and analysis. Thus the following paragraphs will address aspects of these concepts and provide ideas and justify hybrid approaches.

Data augmentation techniques, in general, can improve machine learning and neural network time series predictions. An increased amount of data available for training increases the accuracy of the employed algorithm on many data sets. Here the work by Semonoglou et al. [26] is exemplary. The researchers list a variety of data augmentation techniques such as time series flipping, random noise injection, time series combinations, magnitude warping, and **time series interpolation**. These techniques can reportedly improve the accuracy of neural network (in this case, a multi-layer perceptron) forecasts on univariate time series data because of the increased amount of training data. This is related to our work. We also employ neural networks to forecast univariate time series data and use data augmentation and interpolation to increase the amount of training data.

Regarding our choices of interpolation techniques, we employed a simple linear interpolation for reference for all of our experiments. Further, we employed a fractal interpolation similar to the one developed by Manousopoulos et al. [27]. In this article, the researchers present a fractal interpolation based on iterated function systems to interpolate arbitrary two-dimensional curves. We used the same iterated functions system, which can be tailored to match specific requirements by adjusting a vertical scaling factor, which significantly influences the y-coordinate of interpolated data points, and thus the Hurst exponent of fractal interpolated time series data, as discussed in [33]. Thus one can take into account the Hurst exponent when using fractal interpolation. The Hurst exponent is an indicator for the predictability of a time series data as it is directly related to the probability of fractional Brownian motions to change direction, [7, 34, 35].

However, if the Hurst exponent can be used to indicate predictability for machine learning models and/or identify regions of increased predictability is the subject of ongoing research, and in the author's opinion, the most popular combined approach of measure of signal complexity and machine learning for time series analysis in the scientific community. Yao et al., [36] thus use a neural network and ARIMA models to forecast the Kuala Lumpur composite index. The researchers also employ R/S analysis and thus the Hurst exponent to analyze the data under study for featuring long-term memory. Similar to this is the study by Yakuwa et al. [18], where fractal analysis, including the Hurst exponent, is performed for the Nikkei stock prices for 1500 days. Here the Hurst exponent suggests persistent behavior and can thus be forecast. Further, this knowledge is then used to tailor the input window for a neural network to predict the data under study. The results show that a neural network implementation can be improved using knowledge obtained from fractal analysis.

Literature also supports the claim that time series or parts of a time series with a larger Hurst exponent can be forecast more accurately than those with a Hurst exponent close to 0.5. This is tested in the work by Qian et al., [8] for neural networks, thus showing that the Hurst exponent can indicate time series and/or regions of increased neural network predictability. Further, the research done by Selvaratnam et al. [37] shows that neural networks can be improved by using R/S analysis to estimate a period of increased persistency, which is then used to tailor the input window of a neural network. A similar strategy is used in the work by Qian et al. from 2007, [38] where R/S analysis is used to estimate regions of increased predictability for machine learning approaches. Further, the researchers employ a phase space reconstruction to tailor the input nodes of a neural network. However, the research done by Diaconescu,[15], also discusses the predictability of data with increased Hurst exponents, but in this case, compares model data against real-life data. Though the Hurst exponent suggests that all of the time series can be forecast theoretically, the employed neural networks perform significantly worse on real-life stock market data. Thus suggesting a certain arbitrariness for the interpretability of R/S analysis with respect to indicating the predictability of machine learning approaches. However, it is common practice to do just that, i.e., indicating persistency in time series data and then predicting it to state that it can be predicted. This is done in the works by Ghosh et al. [14, 39], and in the work by De Mendonça Neto et al. [40].

We conclude from the previous discussion on connections between the Hurst exponent and the predictability of time series data that the Hurst exponent can characterize time series and indicate predictability in time series data to enhance machine learning approaches. However, given the previously discussed data augmentation and interpolation approaches to improve neural network time series predictions, a reasonable continuation of this might be to consider the Hurst exponent when interpolating time series data and/or performing data augmentation.

However, we can choose stochastic interpolation techniques where we can set a fixed Hurst exponent for a whole time series data, e.g., the multipoint fractional Brownian bridges discussed by Friedrich et al. [29]. Also, one can choose interpolation techniques based on genetic algorithms and a certain loss function, as done in the work by Chang et al. [41]. We used both of these ideas in this thesis for PhaSpaSto interpolation.

As already discussed regarding the applicability of the Hurst exponent for indicating the predictability of machine learning algorithms and choosing a fractal interpolation, we can employ complexity metrics to analyze time series data with respect to their applicability for machine learning and/or use complexity metrics to improve machine learning approaches. C continuation of these ideas is to exchange the Hurst for other measures of signal complexity.

Here the work by Karaca et al. is exemplary, [42, 43], where stock market data is analyzed and forecast using a variety of complexity metrics, e.g., the already discussed Hurst exponent, Shannon's entropy, Rényi entropy and wavelet entropy. These metrics can be used as additional features for the machine learning algorithm to improve the accuracy of the prediction. Also, one can use the fractal dimension of time series data, which is closely related to the Hurst exponent, to analyze time series and/or improve prediction approaches as done in the work by Ni et al. to select appropriate features to predict stock market data [44].

Finally, we mention an outstanding idea combining machine learning and Takens' theorem to reconstruct phase spaces from univariate time series data. The work of Gilpin [16] introduces a machine-learning model to reconstruct phase spaces based on autoencoders and a novel latent-space loss function. Further, this technique can create representations of stochastic systems with increased predictability.

Chapter 3

Measuring Complexity and Chaoticity

This chapter lists several measures for complexity and chaoticity of time series data. These measures have been used throughout the presented research, and the underlying ideas are discussed in this section.

3.1 Hurst Exponent, R/S Analysis, Hurst-Error

The Hurst exponent is a measure for the long-term memory of a time series data and is calculated by R/S Analysis, [7]. Following [45] and [7]:

R/S analysis (Rescaled range analysis) identifies long-run correlations in time series, yielding one parameter, the *Hurst exponent* "*H*".

Given a signal $[x_1, x_2, \ldots, x_n]$, one finds the average over a period τ (a sub-interval of the signal, i.e. $1 \leq \tau \leq n$), with k as $1 \leq k \leq n$ and elements i in this interval such that $k \leq i \leq k + \tau$:

$$\langle x \rangle_{\tau,k} = \frac{1}{\tau} \sum_{j=k}^{k+\tau} x_j \quad . \tag{3.1}$$

Next, an accumulated departure $\delta x\,(i, \tau, k)$ over a period $i \in 1, 2, \ldots, \tau$ is calculated as:

$$\delta x\,(i, \tau, k) = \sum_{j=k}^{i} \left(x_j - \langle x \rangle_{\tau,k} \right) \tag{3.2}$$

The range R, which is the the difference between maximal and minimal values for all x_i in $[k, k+\tau]$ is:

$$\begin{aligned} R(\tau, k) &= \max\left[\delta x(i, \tau, k)\right] - \min\left[\delta x(i, \tau, k)\right] , \\ &\text{satisfying } k \leq i \leq k + \tau \, . \end{aligned} \tag{3.3}$$

And finally, the standard deviation for each subinterval is:

$$S(\tau, k) = \sqrt{\frac{1}{\tau}\sum_{i=k}^{k+\tau}\left[x_i - \langle x\rangle_{\tau,k}\right]^2} \quad . \tag{3.4}$$

The range and the standard deviation are then averaged over all possible [1] k such that:

$$\begin{aligned} R(\tau) &= \frac{\sum_k R(\tau, k)}{\text{number of different } k\text{s}} \\ &\text{and} \\ S(\tau) &= \frac{\sum_k S(\tau, k)}{\text{number of different } k\text{s}} , \end{aligned} \tag{3.5}$$

where $1 \leq k \leq n$ and $k \leq i \leq k + \tau$. The Hurst exponent H is then given by the corresponding scaling properties:

$$\frac{R(\tau)}{S(\tau)} \propto \tau^H \quad , \tag{3.6}$$

We then find the asymptotic behavior for an independent random process with finite variance as:

$$\frac{R(\tau)}{S(\tau)} = \left(\frac{\pi}{2v}\tau\right)^{\frac{1}{2}} \quad , \tag{3.7}$$

thus gives the Hurst exponent as $H = \frac{1}{2}$ for random processes. For time series data which are not completely random, e.g., non-random real-life data, we expect $H \neq \frac{1}{2}$, as real-life processes usually feature long-term correlations.

The range of H is $0 < H < 1$. Here, a value $H < 0.5$ indicates anti-persistency, meaning that it is heavily fluctuating but not completely random. Values close to 0 indicate strong anti-persistency. Contrary to that, $H > 0.5$ indicates persistent processes. Further, it indicates strong persistency for values close to 1. Also, given these ranges, time series with $H \neq 0.5$ can theoretically be forecast, [37].

R/S Analysis is plotted in Figure 3.1, i.e. the ratio on a logarithmic scale against the intervals, also on a logarithmic scale. The Hurst exponent is the corresponding slope of the fit.

[1]The algorithms that perform R/S analysis find a subset of possible intervals and do perform the procedure on all possible intervals.

Another parameter related to R/S analysis characterizing a process's fractal and random behavior can be found by measuring the distance of the actual data points to the Hurst-fit, i.e., the residuals. This is measured using a root mean squared error. We refer to this as *Hurst-error*. This Hurst-error can differentiate between mono-fractal and multi-fractal time series data. Given two-time series with the same Hurst exponent but different Hurst errors, we state that the time series with the larger Hurst error is a more multi-fractal one, i.e., the range of the fluctuations differs for different scales. Contrary to that, a time series with a Hurst-error of zero is a perfectly mono-fractal time series, i.e., we find changes of the same range on all scales. The Hurst-fit and the corresponding Hurst-error can be seen in Figure 3.1[2].

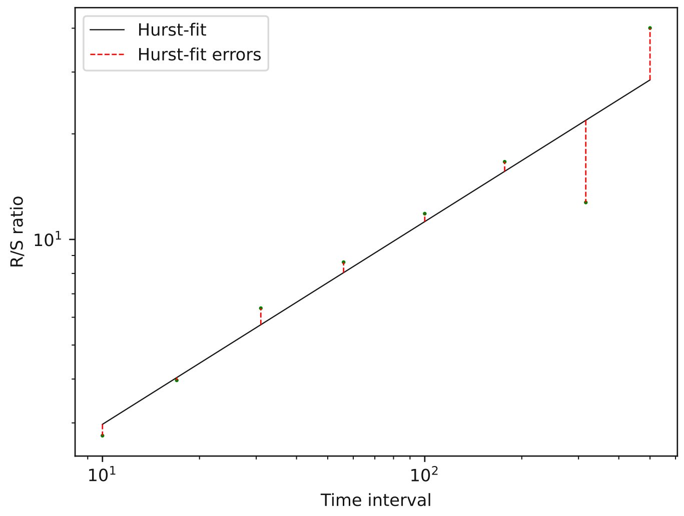


FIGURE 3.1: Double logarithmic plot for the fit of the Hurst exponent for a random walk with a probabilty of 0.5 and a length of 500 steps. The calculated Hurst exponent is $H = 0.57$, and the corresponding Hurst-error is $\text{RMSE}_{\text{Hurst}} = 5.229$. This results from different, i.e. larger, fluctuations for larger time intervals than for smaller time intervals. This inability to give the correct Hurst epxonent, i..e to under- or overestimate it is inherent to most algorithms and we discuss this issue in Section 13.4 and Appenidx E

3.2 Fractal Dimension

The basic idea of the fractal dimension of a time series is to consider the time series as a two-dimensional plot lying on a grid of equal spacing and then count the number of grid boxes covering the whole time series. This gives a ratio of the overall area and the grid area of the time signal. This process is referred to as *box-counting*. Thus the fractal dimension of a time series is a measure of the complexity of the signal, e.g., a straight

[2]Note that the test data for this fit was altered, such that the plot is explanatory and indicative. This was done because a fractional Brownian motion would usually not give such a big Hurst-error.

line would have a very low fractal dimensioan, i.e., 1, but this may vary depending on the employed algorithm. The fractal dimension can have a non-integer value, i.e., the fractal dimension D of a self-affine time series can have values $1 < D < 2$.

One can choose between several algorithms to calculate the fractal dimension of a time series. The following three concepts were employed in this research, i.e., the algorithm by Higuchi [46], the algorithm by Petrosian [47], and the algorithm by Katz [48].

3.3 The Spectrum of Lyapunov Exponents

The spectrum of Lyapunov exponents measures the system's predictability depending on initial conditions. For experimental time series data, as we cannot choose between different trajectories for different initial conditions, the spectrum of Lyapunov exponents still serves as a measure for the predictability of the system, [49]. The corresponding algorithm is too complex to be discussed here in length, and thus the interested reader is referred to [49] for an in-depth discussion of the topic.

The largest Lyapunov exponent is always the first one in the spectrum. In general, a positive Lyapunov exponent is a strong indicator for chaos [50]. In most cases, it is therefore sufficient to calculate only the first Lyapunov exponent of the spectrum, hence the largest Lyapunov exponent. Systems that possess more than one positive Lyapunov exponents are referred to as hyperchaotic, [51, 52].

3.4 Shannon's Entropy

Entropy is a measure of the unpredictability of a state or, equivalently, of its average information content. Shannon's entropy is the first of a family of entropy measures and a foundational concept of information theory, [53, 54].

Shannon's entropy of a signal can be understood as a measure of irregularity.

Note that the following entropy definition holds for any base b of the logarithm. Using $b = 2$ returns information in *bit*, $b = \mathrm{e}$ (Eulers' number) gives *nat* and for $b = 10$ the unit is called *digit*.

This thesis uses the implementation provided by [55]. Shannon's entropy is thought initially to describe discrete signals and does not well for continuous signals. One way to enhance the algorithm to deal with continuous data is to use binarization, where the signal is cut into a number of bins, then used to determine Shannon's entropy. Another option would be to *fuzzyfy* the signal and then apply Shannon's entropy to the fuzzy set, which is basically the idea of Fuzzy entropy, [56].

Still, we used the basic concept where one counts the frequency of reoccurring values. We are aware that this does not provide a meaningful interpretation of Shannon's entropy as in a continuous signal, values are hardly ever reoccurring. However, it is useful in differentiating between periodic and irregular/random signals. Thus Shannon's entropy is increased for, e.g., the Lorenz system and fractional Brownian motions compared to a cosine function, as discussed in Appendix E. Also, the applicability of Shannon's entropy for our purposes is further discussed in Section 13.2.

Given a signal $[x_1, x_2, \dots, x_n]$, we find the set of unique values of this signal as $\{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m\}$, where the total number of unique values is bound by $m \leq n$. For each unique value $\hat{x}_i$, we count the number of occurrences within the signal and refer to this count as $N_i \in \mathbb{N}$. Finally, we obtain the corresponding probabilities as $p_i = \frac{N_i}{n}$. This results in Shannon's entropy as:

$$H_{\text{Shannon}} = -p_i \sum_{i}^{m} \log_b (p_i) \tag{3.8}$$

3.5 SVD Entropy

SVD entropy is an entropy measure based on the ***S**ingular **V**alue **D**ecomposition* of a correlation or embedding matrix, thus the name. It is known to be applicable for analyzing univariate time series data such as stock market data, [57–59], or electroencephalography (EEG) signals, [60] . Furhter, the idea behind SVD entropy is that systems of decreasing complexity will yield decreasing entropy, [60]. It can be understood as indicating the number of eigenvectors[3] needed for explaining a dataset: I.e., the higher the SVD entropy, the more orthogonal vectors are required to explain the space state.

Following [60]: SVD entropy of a signal $[x_1, x_2, \dots, x_n]$ is obtained by constructing an embedding space for the signal with delay vectors as :

$$\vec{y}\,(i) = \left[x_i, x_{i+\tau}, \dots, x_{i+(d_{\mathrm{E}}-1)*\tau}\right] \quad , \tag{3.9}$$

where τ is the time delay and d_{E} is the embedding dimension. One constructs the embedding space in matrix form as:

$$Y = \left[\vec{y}\,(1), \vec{y}\,(2), \dots, \vec{y}\,(N - (d_E - 1)\,\tau)\right]^{\mathrm{T}} \quad , \tag{3.10}$$

[3]We will not dive into explaining the similarities and differences between singular values and eigenvalues here, instead, the interested reader is referred to [61].

which is a real $d_E \times n$ matrix. Singular value decomposition then leads to a factorization of the form:

$$Y = U\Sigma V^{\dagger} \tag{3.11}$$

Here U is a $d_E \times d_E$ real unitary matrix. Further, for real matrices, unitary is the same as orthogonal; thus, U is also orthogonal. V is an $n \times n$ unitary and real, thus orthogonal, matrix. The conjugate transpose $V^{\dagger}$ becomes V^{T}, i.e., just the transpose. Here Σ is a $d_E \times n$ rectangular diagonal matrix. Further, this matrix has a diagonal of non-negative real numbers. These diagonal entries of Σ, i.e., $\sigma_i = \Sigma_{ii}$, are referred to as the singular values of Y. We find r of these singular values for Y, where r is bound by the rank of Y such that $r \leq \min\{m, n\}$. A spectrum of normalized singular values is then obtained by:

$$\bar{\sigma}_i \; = \; \frac{\sigma_i}{\sum_{j=1}^{r} \sigma_j} \tag{3.12}$$

The basic concept of Shannon's entropy (See Equation 3.8) then gives SVD entropy as:

$$H_{\mathrm{SVD}} \; = \; -\sum_{i=1}^{r} \bar{\sigma}_i \log_2 \bar{\sigma}_i \tag{3.13}$$

Here, one might run into the problem of zero-valued singular values of a given matrix and, consequently, the issue of calculating the logarithm of 0, which is not defined. Here, Roberts et al., [60] argue that for real-life data, one will not run into this problem due to noise, including quantization noise. However, we still need to mention that a solution to this problem is to use the singular value decomposition for the symmetric covariance matrix $\hat{Y} = Y \times Y^{\mathrm{T}}$, thus $\hat{Y} = U\Sigma V^{\mathrm{T}}$.

One can employ the concept of average mutual information to estimate the time delay. One may use the false nearest neighbors algorithm to set the value for the embedding dimension. However, Roberts et al., [60] suggest an embedding dimension of $d_E = 20$ to characterize a given state. The concept of a phase space embedding and the corresponding algorithms are discussed in Chapter 4.

3.6 Fisher's Information

Fisher's information gives the amount of information extracted from a set of measurements, thus can be interpreted as the quality of the measurements, [62]. Similar to SVD entropy (Section 3.5), it indicates the number of singular values needed for explaining a dataset or a state, as Fisher's information, the way we used it here, is also based on a singular value decomposition. However, contrary to SVD entropy, Fisher's information depicts the difference between the individual singular values rather than employing

Shannon's entropy for analysis. An increased Fisher's information thus depicts increased variability of the singular values of a signal, i.e., the more random and noisy a signal, the higher the corresponding Fisher's information. We discuss these aspects in Appendix E.

One can find a discrete version of Fisher's information suitable for analyzing univariate time series data given as $[x_1, x_2, \ldots, x_n]$. Here we use the same construction based on a singular value decomposition as for SVD entropy in Section 3.5. Thus we end up with r singular values σ_i, which we normalize by:

$$\bar{\sigma}_i = \frac{\sigma_i}{\sum_{j=1}^{r} \sigma_j} \quad . \tag{3.14}$$

Fisher's information is then found by

$$I_{\text{Fisher}} = \sum_{i=1}^{r-1} \frac{[\bar{\sigma}_{i+1} - \bar{\sigma}_i]^2}{\bar{\sigma}_i} \tag{3.15}$$

.

The discussion on proper phase space embeddings for SVD entropy applies to Fisher's information as well.

Chapter 4

Reconstructed Phase Spaces

This chapter describes all used techniques for reconstructing phase spaces from univariate time series data.

To reconstruct a phase space from a univariate time series data, one essentially requires two parameters, the embedding dimension d_E and the time delay τ. The embedding dimension is the dimension of the reconstructed phase space, e.g., three dimensions. And the time delay is the delay between two consecutive time steps to constitute the embedding vectors. I.e., given a signal $[x_1, x_2, \dots, x_n]$, the corresponding phase space vectors are obtained as:

$$\vec{y}\,(i) = \left[x_i, x_{i+\tau}, \dots, x_{i+(d_E-1)\tau}\right] \quad . \tag{4.1}$$

Also known as Takens' theorem, developed by [17] and [63].

One can choose several methods to determine the parameters τ and d_E. We used the method of average mutual information, selecting a time delay based on autocorrelations, and the false nearest neighbors algorithm. One can find an in-depth discussion on the chosen time delay, the chosen embedding dimension, and the corresponding reconstructed phase space plots for each data set in Appendix A.

4.1 The Method of Average Mutual Information (AMI)

The following description is based on the work of [64, 65].

The concept of the average mutual information can be employed to determine a suitable time delay for the embedding.

Given a signal $[x_1, x_2, \ldots, x_{n-\tau}]$, we first introduce a binning for the signal according to [66]. We now denote the original time series data as signal A, and we find a-time-shifted signal B_τ as $[x_1 + \tau, x_2 + \tau, \ldots, x_n]$. We now suppose that the distributions of A and B_τ are approximated by histograms of $N_A = N_B$ elements. These elements uniformly divide the ranges (a_{min}, a_{max}) and (b_{min}, b_{max}), which are the same for time-shifted signals. The corresponding distributions are denoted as P_A and P_{B_τ}. Next, we find a_i and b_j, which are ith and jth elements of the uniform partitions of the original and the time-shifted signals A and B_τ respectively.

Next, the average mutual information between two signals A and B_τ is then defined as:

$$I_{AB_\tau} = \sum_{a_i\, b_j}^{N_A = N_B} P_{AB_\tau}\left(a_i, b_j\right) \log_2 \left[\frac{P_{AB_\tau}\left(a_i, b_j\right)}{P_A\left(a_i\right) P_{B_\tau}\left(b_j\right)}\right], \tag{4.2}$$

where $P_A\left(a_i\right)$ and $P_{B_\tau}\left(b_i\right)$ are the probabilities of occurencies of a_i and b_i in A and B_τ, respectively. $P_{AB_\tau}\left(a_i, b_j\right)$ is the probability of co-occurrence of a_i in B_τ and b_j in B_τ.

Then, to find a suitable time delay, one determines the first local minimum of this function and takes the corresponding τ to construct the phase space.

Here, a recommendation to choose the number of bins is to set $N_A = N_B = \lfloor\sqrt{\frac{n}{5}}\rfloor$.

4.2 Time Delay from the Autocorrelation Function

The autocorrelation of a time series can be used to identify periodicities in the time series. Also, the autocorrelation function can be used to identify a time delay for a phase space reconstruction. Here one chooses the value for which the autocorrelation function first passes through zero as this provides linear independence, [64].

Following [67]: The autocorrelation function is the average value of the product $x\left(t\right) \cdot x\left(t+\tau\right)$, with a varying time delay τ, for a given signal $x\left(t\right)$. The autocorrelation function $R_x\left(\tau\right)$ is formally defined as:

$$R_x\left(\tau\right) \equiv E\left[x\left(t\right) \cdot x\left(t+\tau\right)\right] = \lim_{T\to\infty} \int_0^T x\left(t+\tau\right) dt \tag{4.3}$$

4.3 The Method of False Nearest Neighbors

The following description is based on the work of [68, 69] and [70].

We employ the method of false nearest neighbors to estimate the embedding dimension of the data under study.

Given a signal $[x_1, x_2, \ldots, x_n]$, with corresponding phase space vectors:

$$\vec{y}\,(i) = \left[x_i, x_{i+\tau}, \ldots, x_{i+(d_E-1)\tau}\right] \quad . \tag{4.4}$$

Suppose the number of d_E time delay coordinates, i.e., the embedding dimension, is too small. In that case, one expects two phase space vectors to be false neighbors, if they are close to each other only because of the projection and not because of the data's/systems' inherent dynamics. Further, two false nearest neighbors have different time evolution and belong to different regions of the underlying attractor. To determine the right embedding dimension, one then analyzes the nearest neighbors of each vector. We denote the nearest neighbor of each vector $\vec{y}\,(i)$ as $\vec{y}\,(\mathcal{N}\,(i))$. The next step is to compare the distances of these points in d_E and d_E+1 dimensions. If the distance in d_E+1 dimensions is large, then the points are just neighbors by projection in d_E dimensions and not true neighbors, i.e., they will further separate with increasing embedding dimension d_E. Thus, if the distances $|x\,(i+d_E) - x\,(\mathcal{N}\,(k)+d_E)|$ are, and consequently, stay small, then only a small amount of the neighbors are false, and the found embedding dimension is sufficient.

Thus we find the following criterion to determine if a neighbor is a false neighbor:

$$\frac{|x\,(i+d_E) - x\,(\mathcal{N}\,(k)+d_E)|}{\|\vec{y}\,(i) - \vec{y}\,(\mathcal{N}\,(i))\|} > R_{tot} \tag{4.5}$$

or if

$$\frac{\|\vec{y}\,(i) - \vec{y}\,(\mathcal{N}\,(i))\|^2 + [x\,(i++d_E) - x\,(\mathcal{N}\,(i)+d_E)]^2}{R_A^2} > A_{tot}^2 \quad , \tag{4.6}$$

with the corresponding variance:

$$R_A^2 = \frac{1}{n}\sum_{k=i}^{n}\left[x\,(i) - \bar{x}\right]^2 \quad . \tag{4.7}$$

Here R_{tot} is a beforehand set parameter and, in most cases, is set to be between $10-20$. The criterion (4.6) compensates for noise and usually $A_{tot} \approx 2$.

By using both criteria (4.5) and (4.6), one then checks all d-dimensional vectors in the data set to compute the percentage of false nearest neighbors. This percentage should drop to zero or a considerably low number with increasing dimension. Given that the percentage of false nearest neighbors is zero or a low number, the corresponding embedding dimension is found sufficient to represent the underlying dynamics.

Chapter 5

Interpolation Techniques

This chapter discusses the two employed interpolation techniques that take into account the complexity and/or phase space characteristics of the data under study. First, a fractal interpolation based on iterated functions systems, and second, a stochastic interpolation taking into account the properties of a reconstructed phase space of the studied data set (PhaSpaSto interpolation).

5.1 Fractal Interpolation

The following discussion is based on the research presented in [27, 28, 71].

Traditional interpolation methods are based on elementary functions such as polynomials. In contrast, fractal interpolation methods employ iterated functions systems. An iterated functions system is a complete metric space X with a corresponding distance function h and a finite set of contractive mappings, $\{w_n : X \to X \text{ for } n = 1, 2, \ldots, N\}$ [72].

A time series is given as a set of M data points as $\{(u_m, v_m) \in \mathbb{R}^2\} : m = 0, 1, \ldots, M$. The interpolation is then applied to a subset of those data points, i.e., the interpolation points $\{(x_n, y_n) \in \mathbb{R}^2 : n = 0, 1, \ldots, N\}$. Both sets are linearly ordered with respect to their abscissa, i.e., $u_0 < u_1 < \ldots < u_M$ and $x_0 < x_1 < \ldots < x_N$. Therefore the set of interpolation points dissects the set of data points into intervals to be interpolated separately.

$\{\mathbb{R}^2; w_n, n = 1, 2, \ldots, N\}$ is an iterated functions system (IFS) with affine transformations

$$w_n \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_n & 0 \\ c_n & s_n \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_n \\ e_n \end{bmatrix} \tag{5.1}$$

which is constrained to satisfy

$$w_n \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \end{bmatrix} \quad \text{and} \quad w_n \begin{bmatrix} x_N \\ y_N \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} \tag{5.2}$$

for every $n = 1, 2, \ldots, N$. As done in [27], solving these equations yields

$$\begin{aligned} a_n &= \frac{x_n - x_{n-1}}{x_N - x_0} \quad , \\ d_n &= \frac{x_N x_{n-1} - x_0 x_n}{x_N - x_0} \quad , \\ c_n &= \frac{y_n - y_{n-1}}{x_N - x_0} - s_n \frac{y_N - y_0}{x_N - x_0} \quad , \\ e_n &= \frac{x_N y_{n-1} - x_0 y_n}{x_N - x_0} - s_n \frac{x_N y_0 - x_0 y_N}{x_N - x_0} \quad . \end{aligned} \tag{5.3}$$

The real numbers a_n, d_n, c_n, e_n are determined by the interpolation points and s_n is a free parameter, the *vertical scaling factor*. To be hyperbolic with respect to an appropriate metric, s_n is bound by $|s_n| < 1$. For an in-depth discussion on the subject, the reader is referred to [73].

As stated in [74], many records and time series do not exhibit a simple monofractal scaling behavior. E.g., different scaling exponents are required for different parts or at different scales of a time series. Thus we choose the fractal interpolation to reproduce local scaling behavior, i.e., we want the fractal interpolation to match a local Hurst exponent, e.g., for a chosen subinterval, as close as possible.

Here constructing an interpolation via fractal interpolation offers self-similarity in small scales. The scaling factors s_n link this self-similarity to the fractal dimension and consequently the Hurst exponent of the data under study, [75], as they directly influence the y-coordinate of an interpolation's data point. Tihs relation to the Hurst exponent is given by $D_f \approx 2 - H$, [76, 77].

5.1.1 Fractal Interpolation Applied

The following procedure was applied to find a suiting fractal interpolation of N_{int} additional data points $(\hat{x}_k, \hat{y}_k)$, $k = 1, 2, \ldots, N_{\text{int}}$ for a time series data of the original equidistant points: $\{(u_m, v_m) \in \mathbb{R}^2\} : m = 0, 1, \ldots, M$, where $u_0 < u_1 < \ldots < u_M$:

1. We pick a subset i of N data points from the original data points. In this subset are the interpolation points $\{(x_n, y_n) \in \mathbb{R}^2 : n = 0, 1, \ldots, N\}$.

   For each of this subsets we find a number of affine transformations
   $w_n, n = 1, 2, 3, \ldots, N$, according to Equation 5.1.

Further, for each of these subsets we find a Hurst exponent H_i

2. For each subset i, the following routine is performed:

   (a) For all pairs $(x_{n-1}, y_{n-1}), (x_n, y_n)$ of consecutive interpolation points, we find N_{int} equidistant initial data points (x_k, y_k) from a linear interpolation between $(x_{n-1}, y_{n-1}), (x_n, y_n)$, where $x_{n-1} < x_k < x_n$.
   Then for all pairs of consecutive interpolation points we perform the following procedure until N_{int} new unique equidistantly distributed data points $(\hat{x}_k, \hat{y}_k)$ are found[1], these are the new additional data points of the interpolant.

      (i) We choose a random vertical scaling factor $\hat{s}$, where $|\hat{s}| \leq 1$.

      (ii) Then for each k we randomly choose one of the transformations w_n and calculate a_n, d_n, c_n, e_n from Equations 5.3.

      (iii) We then apply w_n on (x_k, y_k) I times, such that: $x = x_k$, $y = y_k$ $y' = c_n x + s_n y + e_n$, $x' = a_n x + d_n$ and afterwards $x = x'$, $y = y'$. After each execution of w_n, for all k, if $|x' - x_k| < \epsilon_x$ we keep this data point such that $\hat{y}_k = y'$ and $\hat{x}_k = x_k$. and return to (i) .

   (b) Calculate the Hurst exponent $H_{i,\text{int,new}}$ for the interpolated time series.

   (c) If there was $H_{i,\text{int,old}}$ set beforehand, compare it to $H_{i,\text{int,new}}$. If $H_{i,\text{int,new}}$ is closer to H_i keep the corresponding fractal interpolation and $H_{i,\text{int,old}}$ is set to $H_{i,\text{int,new}}$.

   (d) Repeat this routine, starting with (a) R times.

Remarks:

- The length N of the subset i can be the whole data set. If the choice of the subset length does not fully cover the entire data set such that a full sub-length is not available at the end of the time series, we then choose the last remaining data points of length N, perform the whole procedure for this subset and keep only the missing part. This is depicted in Figure 5.1. For our experiments, we chose the length of the subsets, such that it divides the data set into ten subsets, where the excess data points were treated in the previously described way.
- The presented procedure simplifies the *fractal curve fitting* method presented in [27], such that our approach works only for time series data sorted with respect to their abscissa, and such that we further reduced the number of vertical scaling

[1]Note that for equidistantly distributed data points and a fixed number of additional interpolated data points $\hat{x}_k = x_k$.

factors as we set it constant for a whole subset i. Also, the above-presented procedure could be extended to arbitrary curves in two dimensions using the projections described in [27].

- The Hurst exponent is calculated using R/S Analysis, see Section 3.1.
- The randomly chosen vertical scaling factors s_n impact the Hurst exponent by influencing the y-component and thus the range and standard deviation in R/S analysis, as discussed in Section 3.1. This is apparent as $\hat{y}_k = c_n x + \underline{s_n} y + e_n$.
- The number of iterations R is set to 500 for each data set.
- (iii) describes a random iteration algorithm according to [78].
- The difference between the abscissa ϵ_x is set to 10^{-4}.
- The number of iterations I is set to 100.
- The larger the data set and the larger one chooses the subsets, the better this method works, as the estimation of the Hurst exponent does not work well for small data sets, as discussed in Appendix E.
- We provide a python implementation of this interpolation technique in [79].

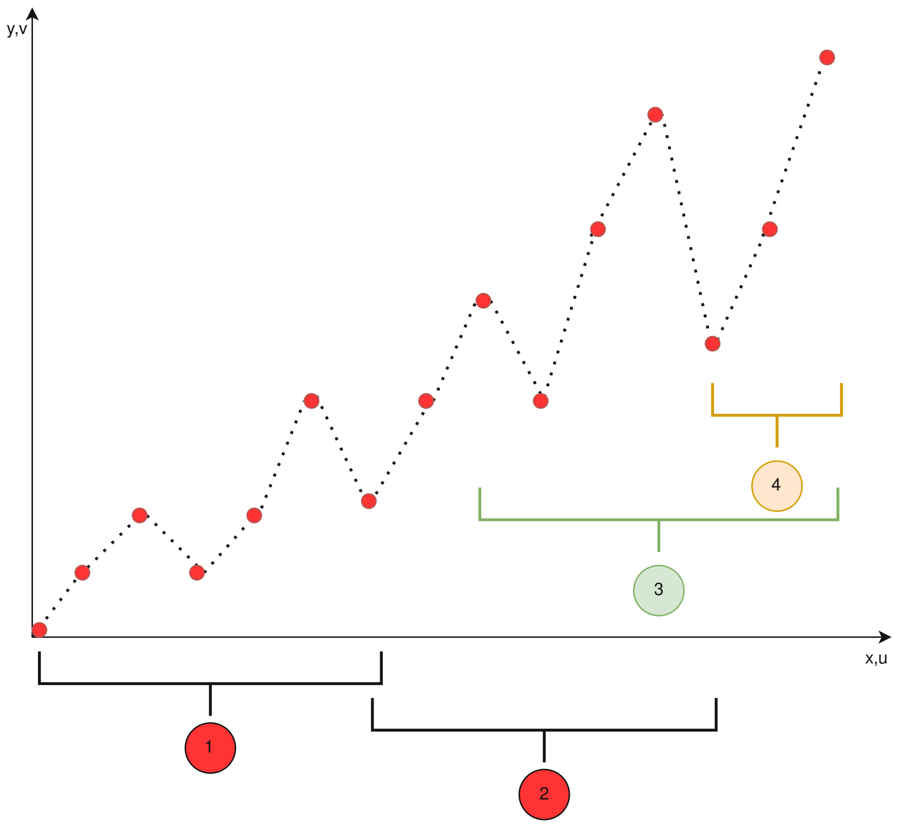


FIGURE 5.1: Schematic depiction of the slicing of the employed fractal interpolation. ①, and ② depict how the slicing is regularly performed throughout the whole time series. ③ shows the interval on which the fractal interpolation is performed for the left-over part. ④ is the left-over part that's actually taken from ③ to complete the interpolation for the whole data set.

5.2 PhaSpaSto Interpolation

The here described interpolation technique is based on the multipoint fractional Brownian bridges, developed in [29], and a basic genetic algorithm to find the best possible interpolation that guarantees a smooth reconstructed phase space trajectory. For simplicity, we refer to the developed method as *PhaSpaSto* interpolation, which is an abbreviation for **pha**se-**spa**ce-trajectory-smoothing **sto**chastic interpolation.

First, one generates a population of stochastically interpolated time series data, each featuring a randomly assigned Hurst exponent, and then performs an optimization using the presented genetic algorithm to improve the interpolation. The whole scheme is depicted in Figure 5.2. We provide the program code for this interpolation technique in [80].

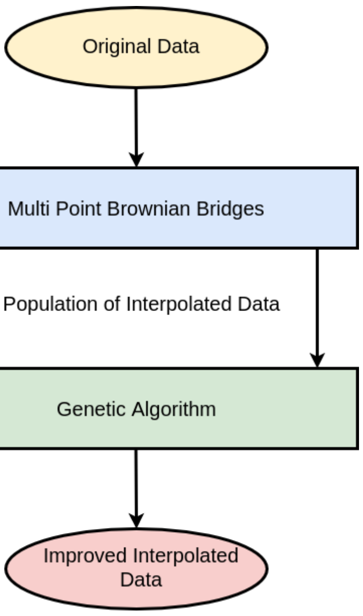

Original Data
Multi Point Brownian Bridges
Population of Interpolated Data
Genetic Algorithm
Improved Interpolated Data

FIGURE 5.2: Schematic depiction of the PhaSpaSto interpolation.

5.2.1 Multipoint Fractional Brownian Bridges

As depicted in Figure 5.2, the employed genetic algorithm is fueled by a population of stochastically-interpolated time series data, in our case multipoint fractional Brownian bridges, [29]. Thus we briefly summarize this approach.

We consider a Gaussian random process $X(t)$ whose covariance is defined as $C(t,t') = \langle X(t)X(t')\rangle$. In the following, we focus on fractional Browian motion where the covariance is given according to $\langle X(t)X(t')\rangle = \frac{1}{2}\left(t^{2H} + t'^{2H} - |t-t'|^{2H}\right)$, where H is the Hurst exponent. To better understand our interpolation strategy, we first define a fractional Brownian bridge, which is a construction of a fractional Brownian motion (fBm) starting from 0 at $t=0$ and ending at X_1 at $t=t_1$, i.e., a fractional Brownian motion that starts at 0 at $t=0$ and ends at X_1 at $t=t_1$, i.e.,

$$X^B(t) = X(t) - (X(t_1) - X_1)\frac{\langle X(t)X(t_1)\rangle}{\langle X(t_1)^2\rangle} \,. \tag{5.4}$$

This construction ensures that $X^B(t_1) = X_1$. This single bridge can now be generalized to an arbitrary number of (non-equidistant) prescribed points X_i at t_i by virtue of a multipoint fractional Brownian bridge

$$X^B(t) = X(t) - (X(t_i) - X_i)\sigma_{ij}^{-1}\langle X(t)X(t_j)\rangle \,, \tag{5.5}$$

where $\sigma_{ij} = \langle X(t_i)X(t_j)\rangle$ denotes the covariance matrix. Furthermore, we imply summation over identical indices.The bridge takes on exactly the values X_k at t_k, as shown by the last linear operation on the Gaussian random process $X^B(t_k) = X(t_k) - (X(t_i) - X_i)\sigma_{ij}^{-1}\sigma_{kj} = X(t_k) - (X(t_i) - X_i)\delta_{ik} = X_k$, where δ_{ik} denotes the Kronecker-delta. As a result, this technique enables the reconstruction of a sparse signal in which the selection of the Hurst exponent H determines the small-scale correlations. We built a simple genetic algorithm to find the best possible interpolation given the data's phase space reconstruction using Taken's theorem. We want our reconstructed phase space curve to be as smooth as possible and thus define the trajectory's fitness as follows

5.2.2 The Fitness of a Trajectory

The basic idea is to use a concept from image-processing, i.e., the blurriness of a picture, and apply it to phase space trajectories. This is because we want our trajectory as blurry, i.e., as smooth as possible. In image processing, the blurriness is determined via second-order derivatives of grey-scale images at each pixel, [81]. We employ this concept, but instead of using it at each pixel, we calculate the variance of second-order derivatives along our phase space trajectories. Similar to the concept from image processing, where the low variance of second-order derivatives implies more blurriness, curves with a low variance of second-order derivatives exhibit comparatively smooth trajectories. The reason here is intuitively clear. Whereas curves with a high variance of second-order derivatives have a range of straight and pointy sections, curves with a low variance of second-order derivatives have a similar curvature along the trajectory and thus are smoother. Hence, in order to guarantee smoothness along the trajectory, we want this variance to be as low as possible, which thus is our loss L. Concluding, our fitness is maximal when our loss L is minimal.

Again we start with the phase space vector and the corresponding embedding dimension d_E and time delay τ (See Chapter 4) of each signal as

$$\vec{y}\,(i) = \left[x_i, x_{i+\tau}, \ldots, x_{i+(d_E-1)*\tau}\right] \quad . \tag{5.6}$$

Thus we have one component for each dimension of the phase space. Consequently we can write the individual components as:

$$y_j\,(i) = x_{i+(j-1)*\tau} \quad , \tag{5.7}$$

where $j = 1, 2, \ldots, d_E$. We then take the second-order finite difference central derivative of a discrete function [82]:

$$u_j''\,(i) = x_{i+(j-1)*\tau+1} - 2x_{i+(j-1)*\tau} + x_{i+(j-1)*\tau-1} \quad , \tag{5.8}$$

at each point, and for each component. Next we add up all the components as:

$$u''(i) = \sqrt{\sum_{j=1}^{d_E} u''_j(i)^2} \quad . \tag{5.9}$$

And finally, we use the variance of the absolute values of second derivatives along the phase space curve as our loss L of a phase space trajectory:

$$L = \mathrm{Var}_i\left[u''(i)\right] \; . \tag{5.10}$$

5.2.3 Genetic Algorithm Architecture

The employed genetic algorithm consists of the following building blocks:

A candidate solution is an interpolated time series using a random Hurst exponent $H \in (0;1)$. The corresponding population of candidates is, e.g., 1000 of these stochastically interpolated time series data with random Hurst exponents. A population of interpolated time series data is generated using the multipoint Brownian bridges such that, for each member of the population, a random Hurst exponent with $H \in]0;1[$ is chosen, which then defines the interpolation of the member of the population. After generating the population, all members are sorted with respect to their fitness, i.e., the lower the loss L, the better an interpolation is. The mating is implemented such that only the best 50%, with respect to fitness, can mate to produce new offspring. The mating is done such that, for every gene, i.e., each interpolation between two data points, there is a 50:50 chance to inherit it from either one of the parents. The mutation was implemented that, in each generation, there is a 20% chance that a randomly chosen interpolated time series is replaced with a new interpolated time series within a corresponding randomly chosen new Hurst exponent. Also, we implemented a criterion for aborting the program, which was fulfilled if the population fitness mean did not change for ten generations. This described procedure is then performed for 1000 generations. But the 1000 generations were never reached, as the criterion for abortion always triggered around 200 generations, and the program was ended, thus yielding the best interpolation with respect to the fitness of the phase space trajectories before reaching the 1000th generation.

Chapter 6

Neural Network Time Series Prediction

This chapter lists all employed neural network architectures, such as long short-term memory (LSTM), and gated recurrent unit (GRU) neural networks. We further discuss the employed error metrics at the end of this chapter.

6.1 Artificial Neural Networks

The focus of this research and one of today's most common approaches to predicting time series are artificial neural networks (ANNs), [83]. Artificial neural networks (sometimes just neural networks) are data-based learning algorithms that emerged from the idea of human or animal brains. As such, they are constituted of connected neurons. Here, how they are connected is crucial for the task at hand. Each neural network consists of an input layer, hidden layers, and an output layer. We categorize the mentioned applications into two types of neural networks for our research. First, feedforward neural networks, i.e., neural networks connected in a non-cyclic way, and, second, recurrent neural networks, which are connected cyclically, i.e., the input and output or sub-parts of the network are connected such that the neurons form loops, [84].

Further, the neurons feature (non-linear) activation functions and are controlled by adjusting weights for each neuron. Thus the learning process consists of adjusting and readjusting the weights on each neuron, done via, e.g., backpropagation, [85].

6.1.1 Feedforward Artificial Neural Networks

Feedforward neural networks are neural networks that are connected in a non-cyclic manner, thus straightforward. The feedforward neural network and its prototype, the multi-layer perceptron (MLP), are the most simple neural networks. Information propagates in only one direction, i.e., from input to output nodes.

6.1.2 Recurrent Artificial Neural Networks:

Contrary to feedforward neural networks are recurrent neural networks (RNNs), where information can propagate in loops in the network. This architecture was invented to learn temporal dynamic behavior, e.g., speech recognition and time series prediction, [86].

RNNs are capable of using feedback or *recurrent* connections to cope with time series data. Though in principle designed for time series analysis and prediction, a standard RNN suffers the problem that the influence of a given input on the neural network either decays or blows up exponentially when passing through recurrent connections, known as exploding or vanishing gradients. LSTMs are designed to solve this problem and learn the inherent long-term dependencies.

LSTMs, [87], feature a component called *memory block* to enhance their capability to model long-term dependencies. This memory block is a recurrently connected subnet containing two functional modules, i.e., the memory cell and the corresponding gates. The task of the memory cell is to remember the temporal state of the neural network. On the other hand, the gates are responsible for controlling the information flow and consist of multiplicative units. There are three types of gates: Input gates, output gates, and forget gates. The input gates control the information flow into the cell, whereas the forget gates control how information remains in the memory cell. The output gate controls how much information is used for the output activation and decides what will be returned to the rest of the neural network.

The gated recurrent unit (GRU) is another addition to RNNs. As the name says, it is a gate mechanism, to be specific a *forget gate*, [88] [89]. Compared to the previously mentioned LSTM neural network, the GRU has a forget gate but lacks an output gate, thus having fewer parameters than LSTM neurons. In general, the architecture and applications of GRU and LSTM are similar, i.e., GRUs are also used for time series data. Compared to LSTMs, GRUs are shown to perform better on smaller and, thus, less frequent data sets, [90].

[91] gives an in-depth discussion on the different RNN architectures. An overview of the different neural network block architectures is given in Figure 6.1, which is also adapted from [91]. Here x_t is always the input for each neuron, whereas h_t is the output. As an exemplary activation function, and because of its default status, the presented scheme diagrams use a tangens hyperbolicus (tanh) activation function. The additional activation functions of the GRU and LSTM blocks use a sigmoid activation function, depicted as σ. All recurrent architectures feature a recurrent connection, depicted as yellow h_{t-1} and h_t inputs/outputs to the other recurrent cells. The LSTM cell features another connection to other cells, depicted as the blue c_{t-1} and c_t inputs/outputs.

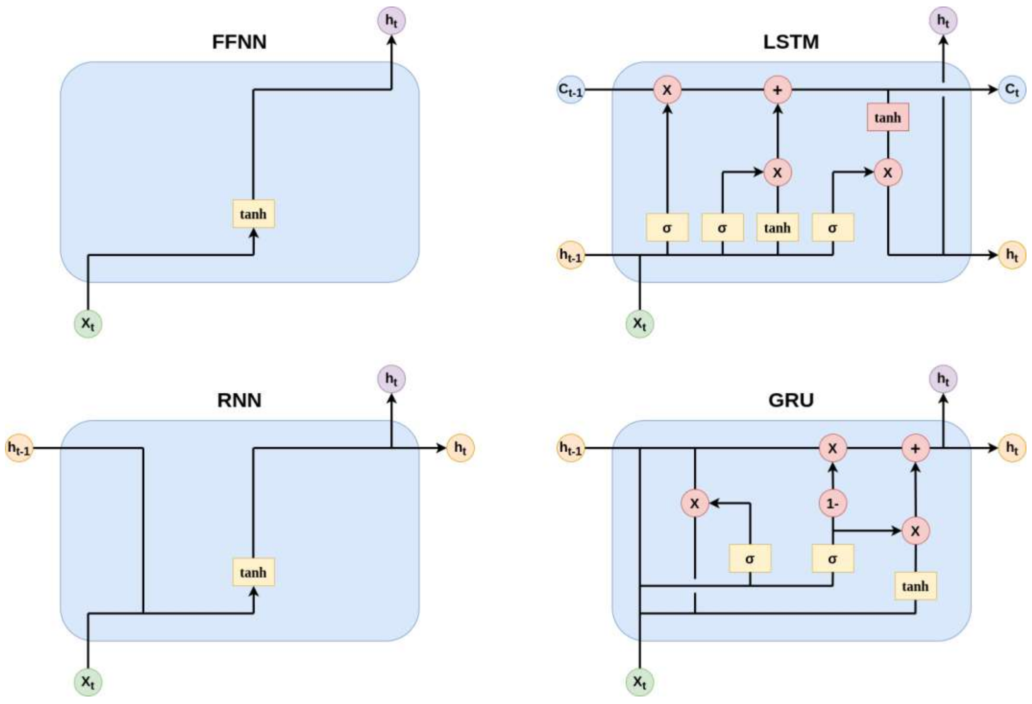


Figure 6.1: Different artificial neural network architectures.
FFNN: Feed forward neural nework
RNN: Recurrent neural network
LSTM: Long short term memory neural network
GRU: Gated recurrent unit neural network.

6.1.3 Randomly Parameterized Neural Networks

This section explains the randomly parameterized neural networks used for all experiments in Chapters 9 and 11. These randomly parameterized neural networks were used to cope with the problem of autoregressive neural network predictions. I.e. good train and test fits do not guarantee that the trained neural network can autoregressively predict time series data.

The idea is to generate many randomly parameterized neural networks to build ensemble predictions based on the inherent complexity or phase space properties of the autorgressively produced predictions. An autoregressive prediction is a one-step-at-a-time prediction, whereas old outputs are used as inputs for the next step.

These randomly parameterized neural networks feature one to five hidden LSTM layers, a hard sigmoid activation function in the hidden and input layers, and a rectified linear unit (ReLU) as the output activation functions. No dropout criteria or regularizations were used.

We used two different architectures for the randomly parameterized neural network approach:

1. Architecture:
   - Data scaled to $[0, 1]$
   - Number of input nodes: 1 $\rightarrow$ size of the training data -1
   - Number of neurons for each hidden layer: 1$\rightarrow$30
   - Batchsizes: 2 $\rightarrow$ 128
   - Epochs: 1$\rightarrow$30
2. Architecture:
   - Data scaled to $[0.1, 0.9]$
   - Number of input nodes: 1 $\rightarrow$ size of the training data -1
   - Number of neurons for each hidden layer: 1$\rightarrow$50
   - Batchsizes: 2 $\rightarrow$ 128
   - Epochs: 1$\rightarrow$50

Though we used LSTM cells for our experiments, one can use any neural network cell in the hidden layer.

6.2 Error Analysis

This section describes the error metrics used in the presented research. The error metric applied to all predictions is the root-mean-square error (RMSE).

We further use the mean-square-error (MSE) to make our results comparable to benchmark results from the literature, see Chapter 12.

We also present an adaption to the ensemble predictions of the randomly parameterized neural networks, i.e., many different predictions, explained in the previous Section 6.1.3.

6.3 Mean-Square and Root-Mean-Square Error

In the presented research we used two different errors for all experiments. First the mean-square-error:

$$E_{MSE} = \frac{1}{S}\sum_{t=1}^{S}\left[\hat{X}(t) - X(t)\right]^2 \quad , \tag{6.1}$$

And second the root-mean-square error (RMSE):

$$E_{RMSE} = \left(\frac{1}{S}\sum_{t=1}^{S}\left[\hat{X}(t) - X(t)\right]^2\right)^{\frac{1}{2}} \quad , \tag{6.2}$$

where $X(t)$ is the original data and $\hat{X}(t)$ is the prediction, S is the number of predicted data points.

6.4 Ensemble Error

For each ensemble prediction, i.e., consisting of N_p different predictions labeled with i for each time step t, we calculated the mean and the standard deviation as

$$\hat{X}(t) = \frac{1}{N_p}\sum_{i=1}^{N_p}\hat{X}_i(t) \; , \quad \sigma(t) = \sqrt{\frac{1}{N_p}\sum_{i=1}^{N_p}(\hat{X}_i(t) - \hat{X}(t))^2} \, , \tag{6.3}$$

where $\hat{X}_i(t)$ is a single observation, $\hat{X}(t)$ is the averaged observation for a single time step, $\sigma(t)$ is the corresponding standard deviation and N_p is the number of different predictions for each time step.

Next, to compare it to the validation dataset, we calculated the root-mean-square error (RMSE) as

$$E_{RMSE} = \left(\frac{1}{N_t}\sum_{t=1}^{N_t}\left[\hat{X}(t) - X(t)\right]^2\right)^{\frac{1}{2}} \quad , \tag{6.4}$$

where $X(t)$ are the data points of the validation dataset and N_t is the number of validation data points. Using error propagation, the corresponding error of the root-mean-square error was calculated as

$$\Delta E_{RMSE} = \sqrt{\left(\frac{\partial E_{RMSE}}{\partial \hat{X}(1)}\right)^2 \sigma^2(1) + \left(\frac{\partial E_{RMSE}}{\partial \hat{X}(2)}\right)^2 \sigma^2(2) + \cdots} \, , \tag{6.5}$$

thus yielding:

$$\Delta E_{RMSE} = \sqrt{\frac{\sum_{t=1}^{N_t}\left[\hat{X}(t) - X(t)\right]^2 \sigma^2(t)}{(N_t) * \sum_{t=1}^{N_t}\left[\hat{X}(t) - X(t)\right]^2}} \quad . \tag{6.6}$$

Chapter 7

Data Sets

For the conducted research, we used several univariate non-model time series data sets and one model data set, i.e., the Lorenz system.

Similar to Brunton et al. [92], we chose the Lorenz system and the measles outbreaks in NYC data sets, as these data sets feature a known attractor structure in reconstructed phase space. Also, we selected some data sets similar to the research done by Domingos et al. [93]. This was done to make our results comparable to state-of-the-art time series forecasts, i.e., hybrid ARIMA and machine learning approaches. We added agricultural relevant time series to our pool of data sets, as this research is part of DILAAG (Digitalization and Innovation Laboratory in **Agricultural** Sciences). The agricultural relevant data sets are the annual wheat and maize yields in Austria and the discharge of the River Krems.

Further, the selection of data sets was decided such that we used several short (around 100 data points) time series data from the time series data library [94] to develop the whole methodology, e.g., the monthly international airline passengers and the Perrin Freres champagne sales data set. To further validate our results, we chose some (in this context) data sets of intermediate length, around 200 to 300 data points. Also, we added the yield data sets as examples for sparsely sampled and challenging data sets. We added two financial time series, the Dow Jones Industrial Average and the USD/GBP exchange rate, to increase the variability of our experiments. And lastly, the Lorenz system, a model data set that we tailored to match our requirements for the task at hand.

We detrended several data sets by subtracting a linear fit to achieve a more confined phase space structure and a more stationary time series. The detrending was done for both the analysis and the actual predictions.

This chapter lists and references all used data sets and shows the corresponding time series plots. Further, the Lorenz system is briefly described at this chapter's end.

7.1 Monthly International Airline Passengers

This is a data set from the Time Series Data Library, [94]. It depicts monthly international airline passengers from January 1949 to December 1960, with an overall 144 data points, given in units of 1000. This data set was detrended for analysis and prediction tasks.

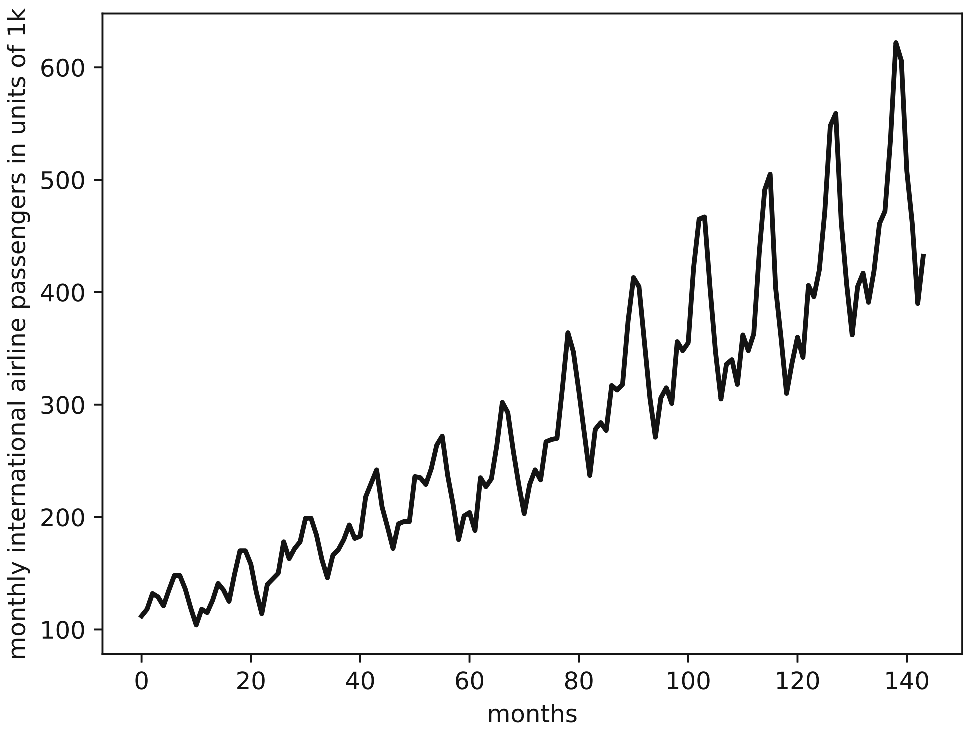


FIGURE 7.1: Plot for the monthly international airline passengers time series.

7.2 Monthly Mean Temperature in Nottingham Castle

This data set is from the Time Series Data Library, [94]. It depicts the mean monthly temperature in Nottingham castle from January 1920 to December 1939 in degrees Fahrenheit, with an overall 240 data points.

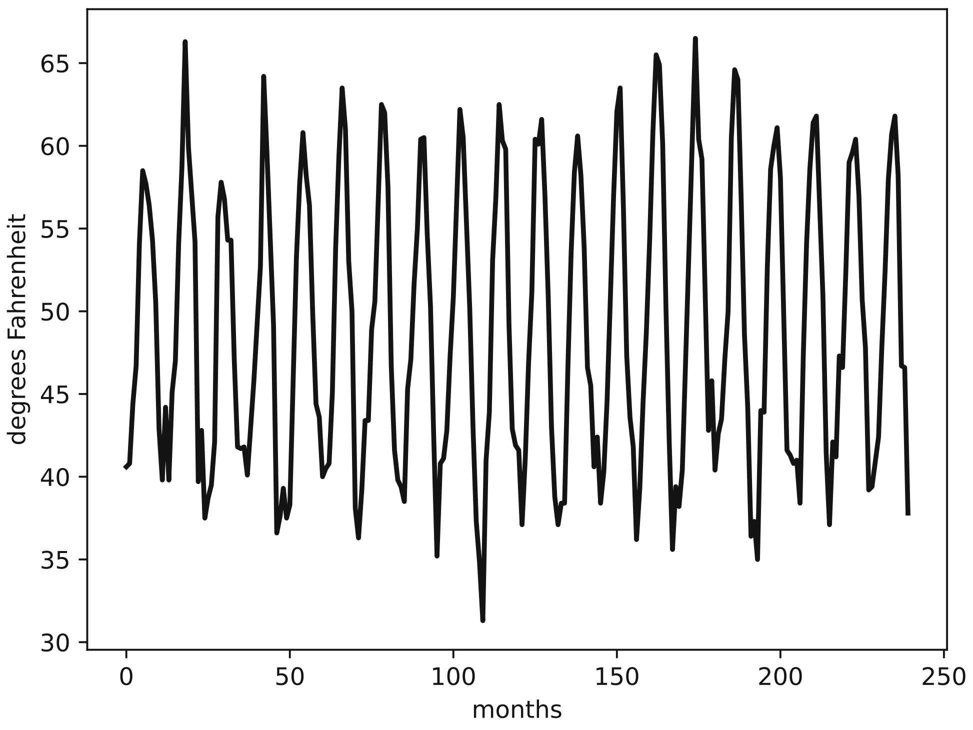


FIGURE 7.2: Plot for the monthly mean temperature in Nottingham castle time series.

.

7.3 Perrin Freres Champagne Sales

This is a data set from the Time Series Data Library, [94]. It depicts Perrin Freres champagne sales from January 1964 to September 1972, with an overall 105 data points. This data set was detrended for analysis and prediction tasks.

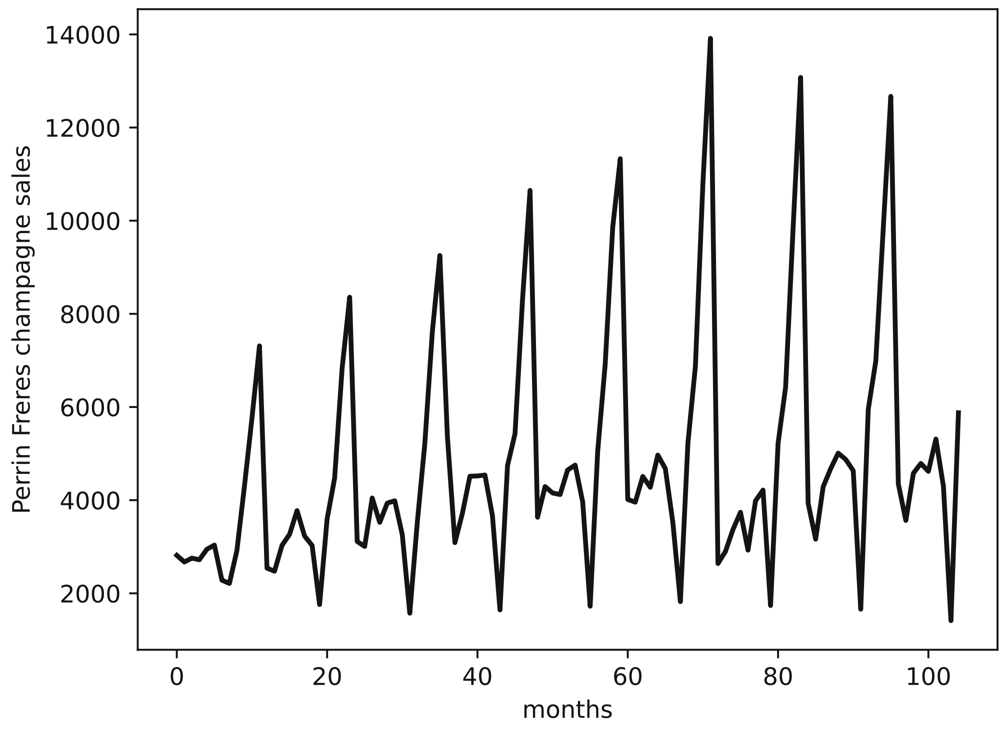


FIGURE 7.3: Plot for the Perrin Freres champagne sales time series.

7.4 Car Sales in Quebec

This is a data set from the Time Series Data Library, [94]. It depicts monthly car sales in Quebec from January 1960 to December 1968, with an overall 108 data points. This data set was detrended for analysis and prediction tasks.

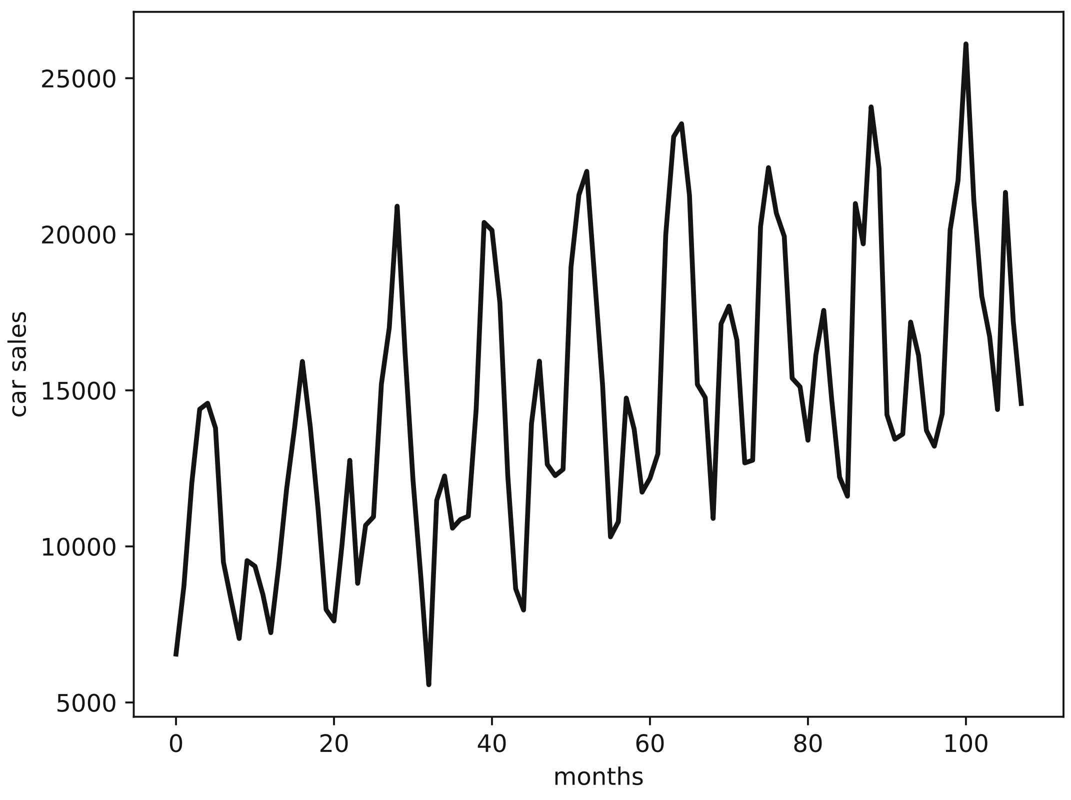


Figure 7.4: Plot for the monthly car sales in Quebec time series.

7.5 NYC Measles Outbreaks

This data set is obtained from [92], where it is discussed and shown to feature an attractor structure in the reconstructed phase space. The corresponding original source is [95]. It depicts Measles outbreaks in New York City (NYC) from 1928 to 1964, binned every two weeks, with an overall 432 data points.

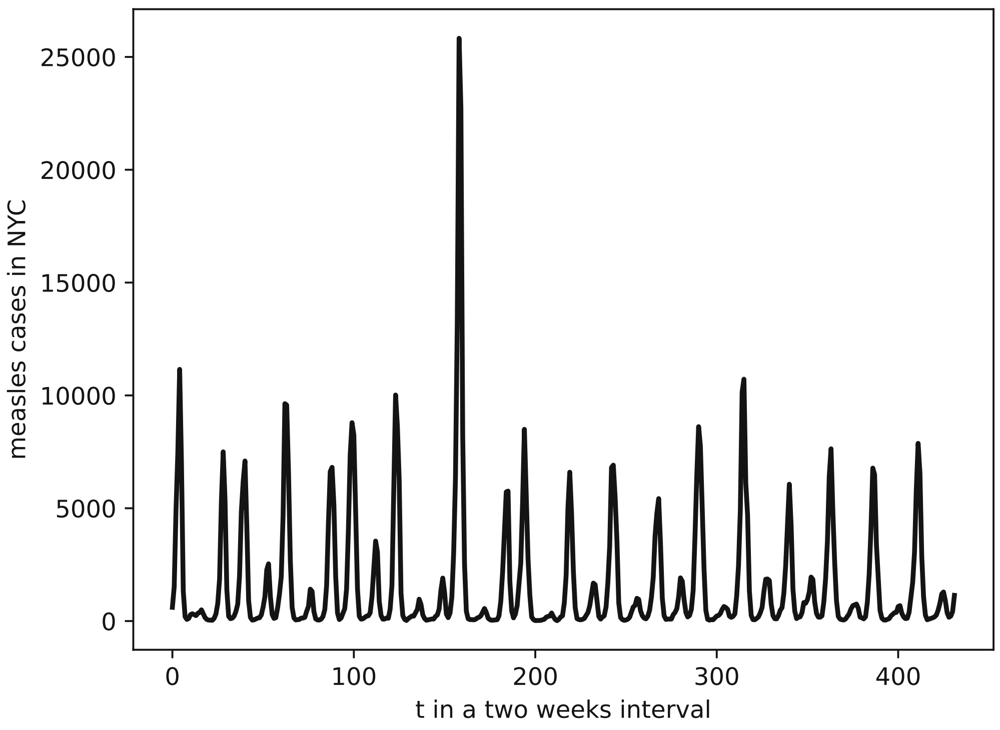


FIGURE 7.5: Plot for the NYC measles time series.

7.6 Annual Wheat Yields Austria

This is a data set of the annual wheat yields in Austria ranging from 1961 to 2017 with an overall of 56 data points. This data set was detrended for analysis and prediction tasks. The data set can be downloaded at http://www.fao.org/faostat/.

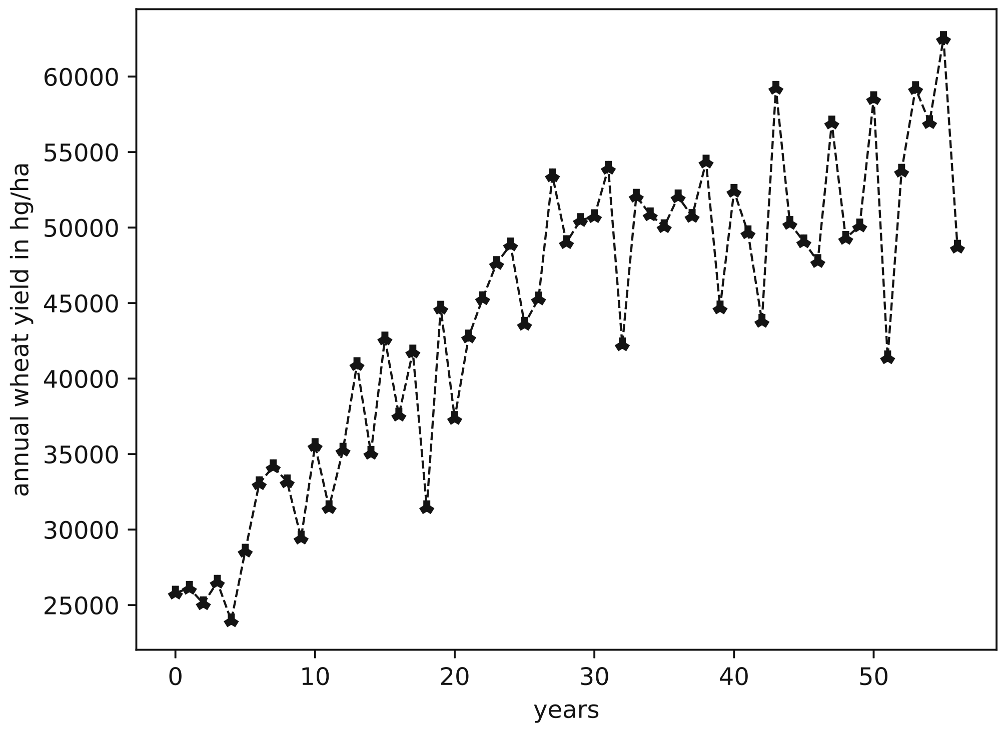


FIGURE 7.6: Plot for the annual wheat yields in Austria time series.

7.7 Annual Maize Yields Austria

This is a data set of the annual yields of maize in Austria ranging from 1961 to 2017 with an overall of 57 data points. This data set can be downloaded at http://www.fao.org/faostat/. This data set was detrended for analysis and prediction tasks.

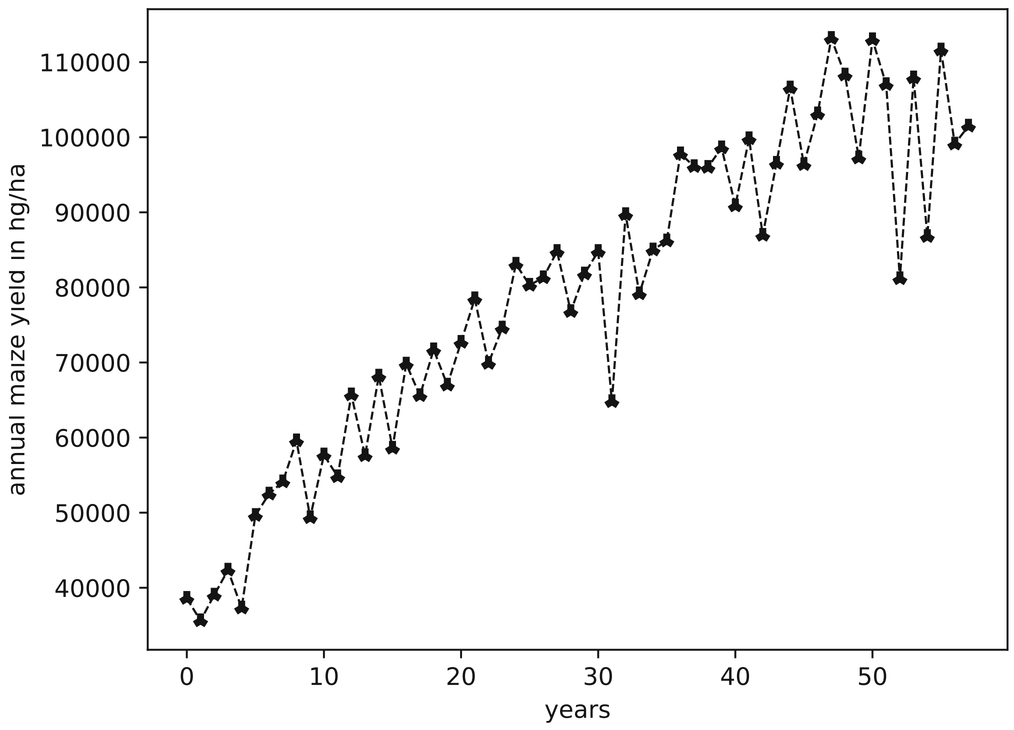


FIGURE 7.7: Plot for the annual maize yields in Austria time series.

7.8 CFE Specialty Monthly Writing Paper Sales

This data set spans 12 years and three months with an overall of 147 data points and describes the monthly CFE specialty writing paper sales. It is another data set from the Time Series Data Library, [94]. This data set was detrended for analysis and prediction tasks.

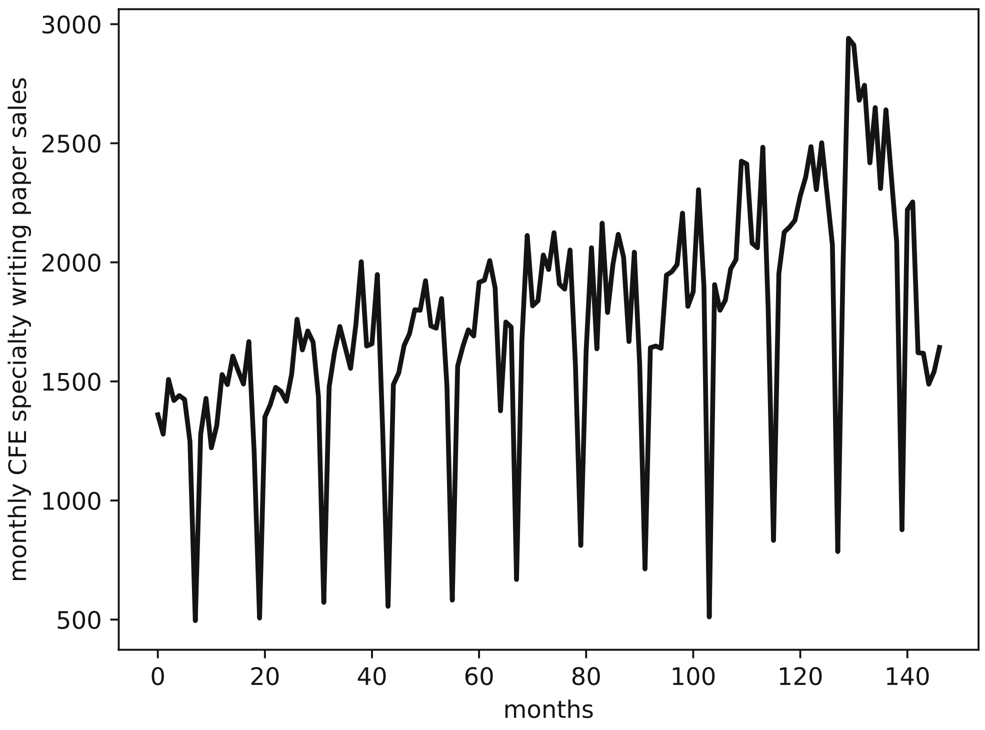


FIGURE 7.8: Plot for the monthly writing paper sales time series data.

7.9 Shampoo Sales

This data set describes monthly shampoo sales over three years, i.e., 36 observations, and is from [96]. This data set was detrended for analysis and prediction tasks.

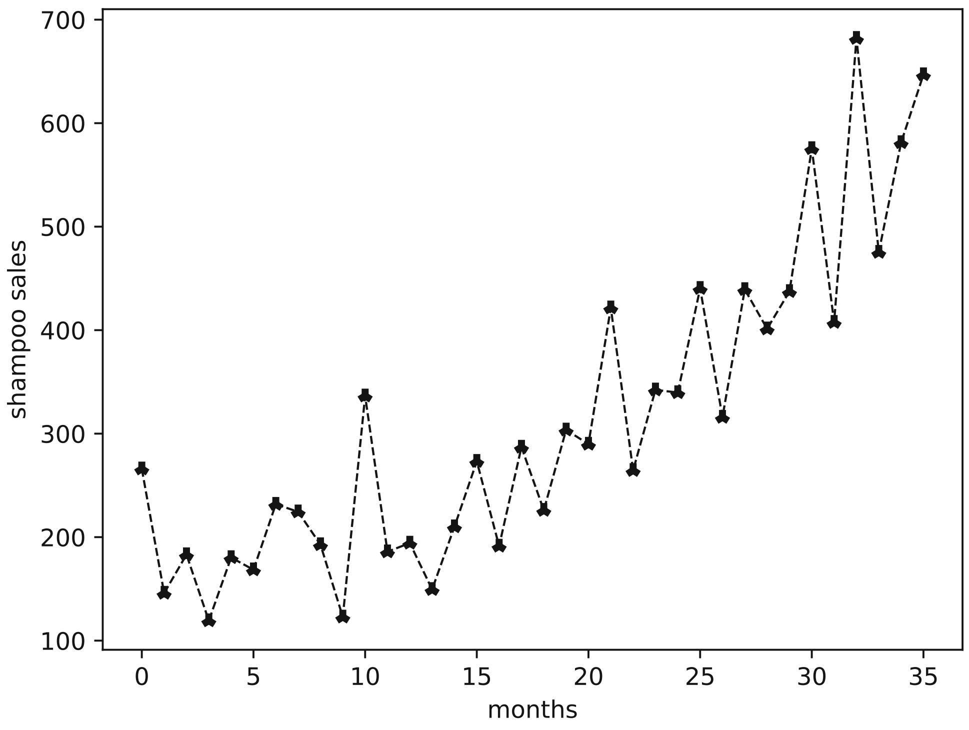


FIGURE 7.9: Plot for the monthly shampoo sales time series.

7.10 Canadian Lynx

This data set is the annual record of lynx trapped in the Mackenzie River district in North-West Canada from 1821 to 1934 and consists of 114 data points. This data set is part of the Time Series Data Library, [94].

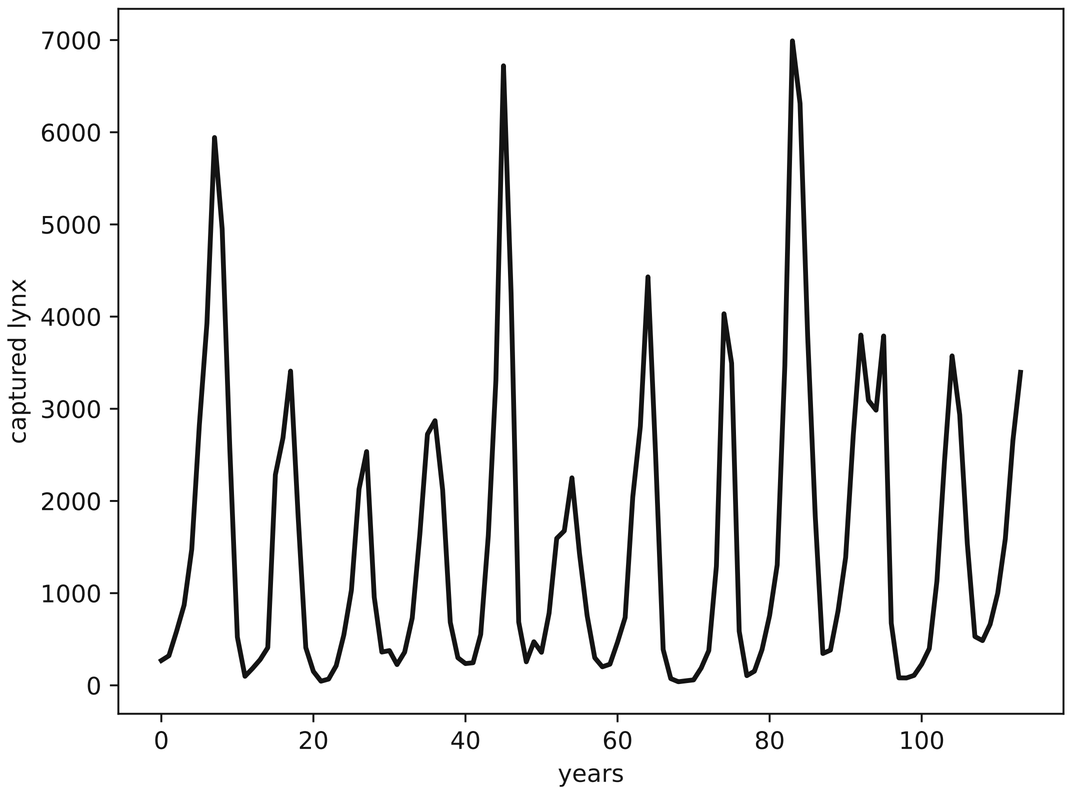


FIGURE 7.10: Plot for the Canadian lynx time series.

7.11 Dow Jones Industrial Average Daily Close in 2018

This data set describes the daily close value of the Dow Jones industrial average in 2018. This data set can be obtained from the website of the Federal Reserve Bank of St. Louis, [97].

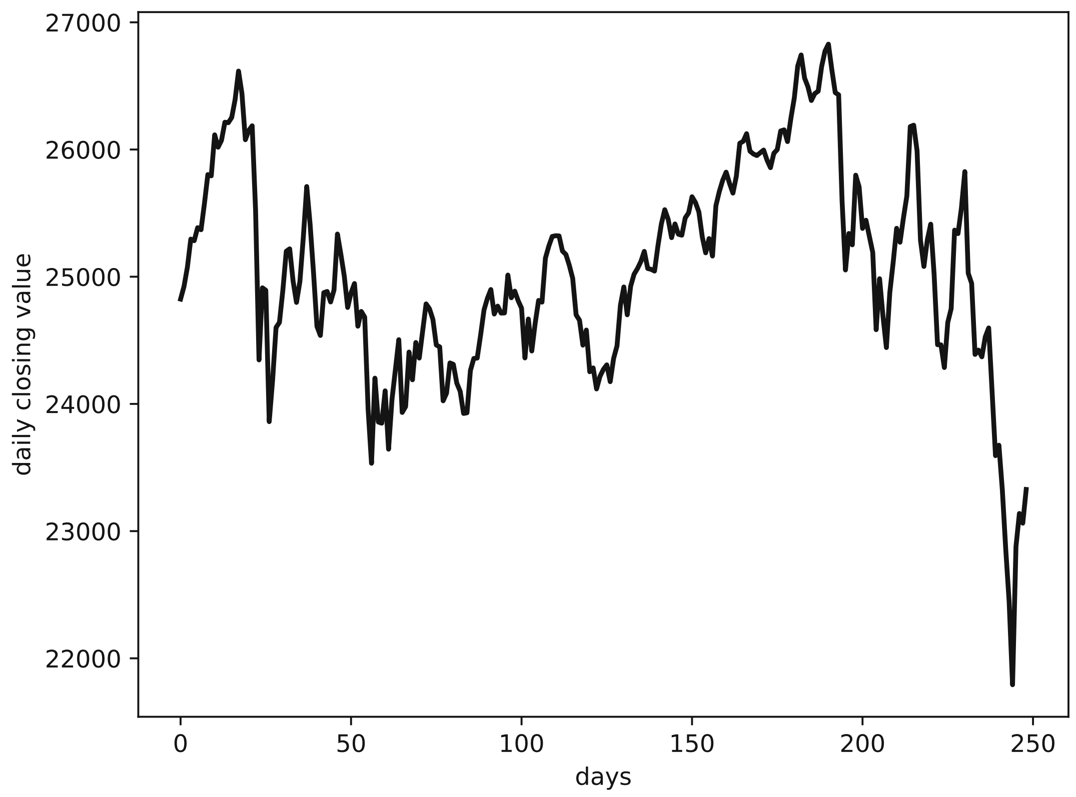


FIGURE 7.11: Plot for the Dow Jones daily close 2018 time series.

7.12 Krems River Discharge in the 1980s

This data set describes the monthly discharge of the river Krems in Imbach, Lower Austria, from January 1980 to December 1989, 120 data points in total. This data set can be obtained from the homepage of the Global Runoff Data Centre, [98].

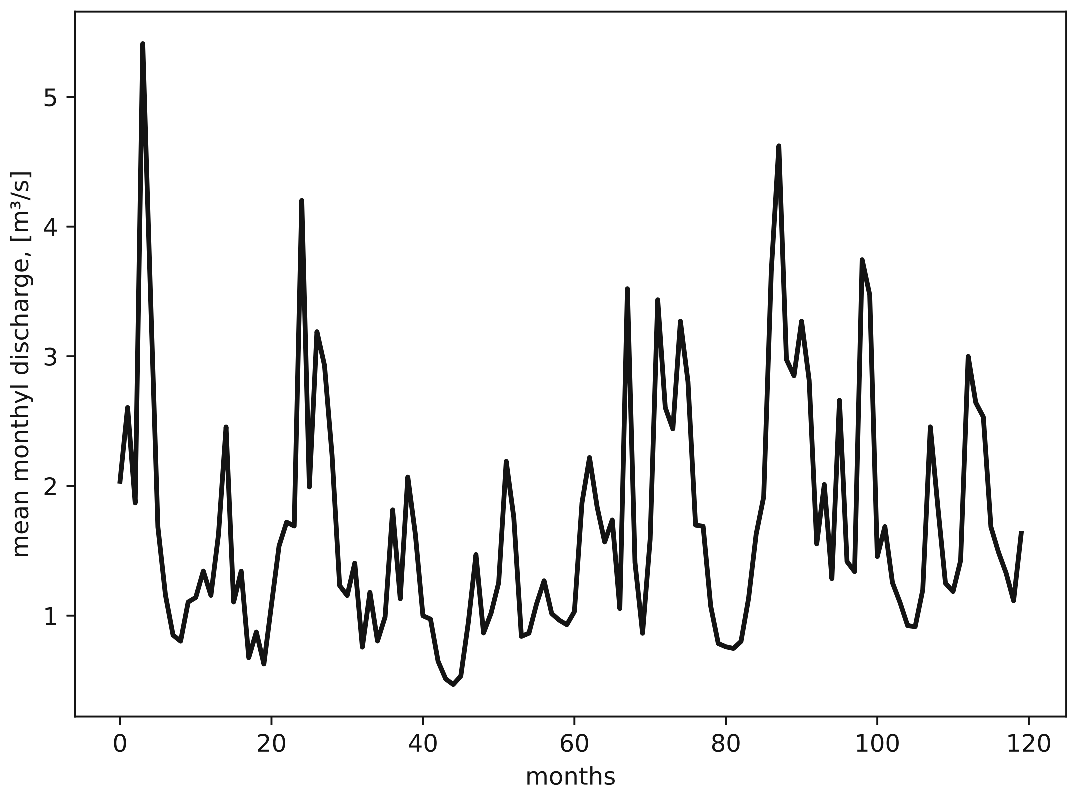


Figure 7.12: Plot for the river Krems discharge in the 1980s time series.

7.13 Sunspots

The sunspots data set are the annual records of spots on the sun's surface between 1700 and 1987, with 288 data points in total. This data set is part of the Time Series Data Library, [94].

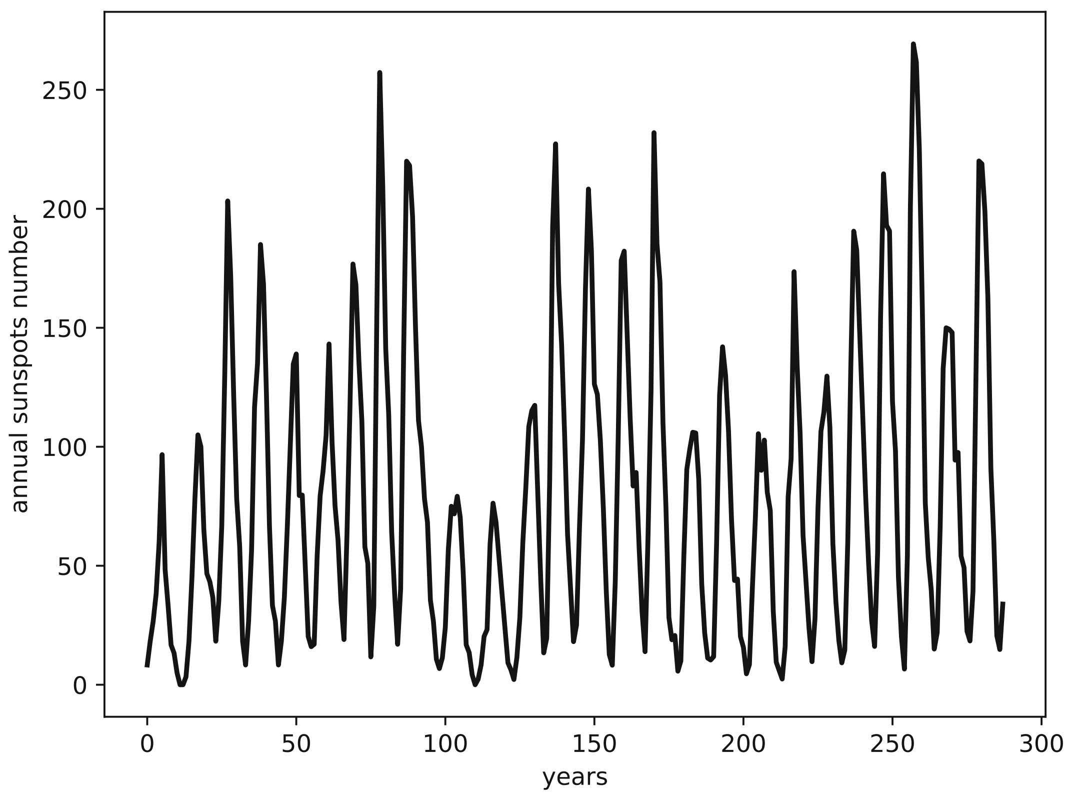


FIGURE 7.13: Plot for the sunspots time series.

7.14 British Pound/US Dollar Exchange Rate

This data set describes the weekly average of the British pound/US dollar exchange rate from 1980 to 1993, 731 data points in total. This data set can be obtained from the website of the Federal Reserve Bank of St. Louis, [97].

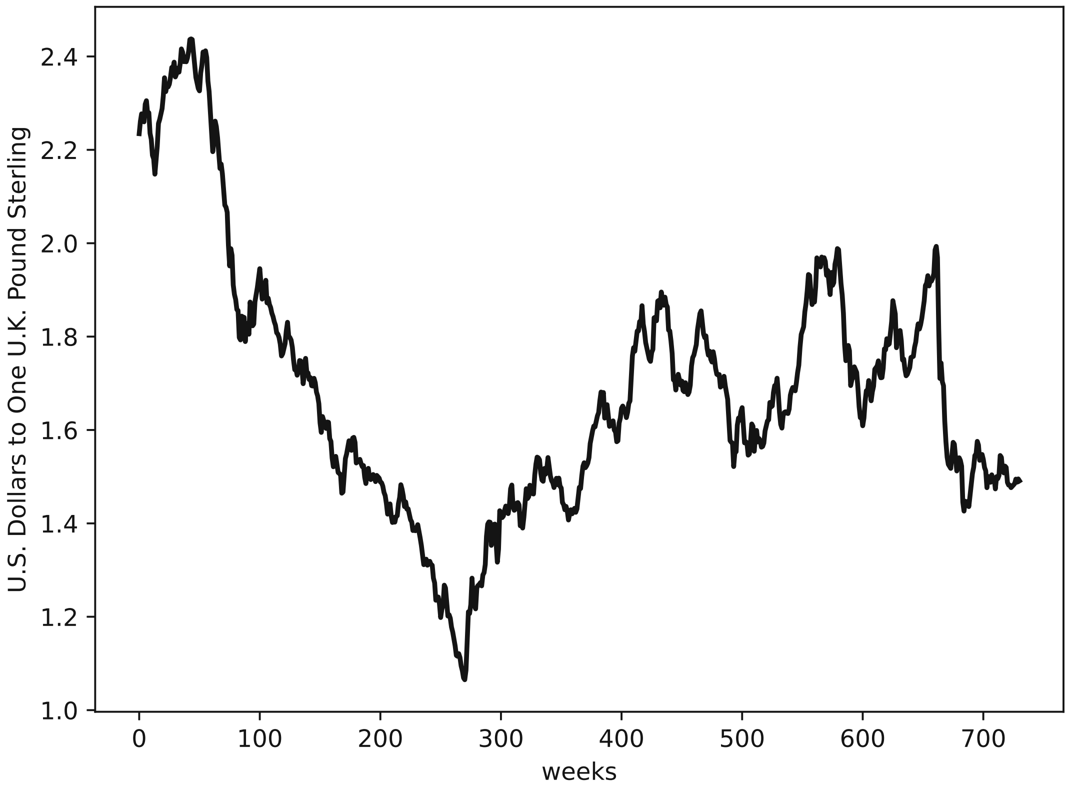


FIGURE 7.14: Plot for the British Pound/US Dollar Exchange Rate time series.

7.15 The Lorenz System

The model data set that was employed for this research is the Lorenz model, [99].

The Lorenz system is a set of three nonlinear equations:

$$\begin{aligned}\frac{dx}{dt} &= 10\,(-x+y)\\ \frac{dy}{dt} &= 28x - y - xz\\ \frac{dz}{dt} &= xy - \frac{8}{3}z\end{aligned} \tag{7.1}$$

We solved this system using a basic Runge-Kutta four approach, [100]. We chose the step size and length of the simulation with respect to the number of interpolation points, as we aim to show how good the system can be interpolated and predicted,

$$dt = \frac{0.1}{n_{int}+1}, \quad L = 200 \cdot (n_{int}+1) \quad , \tag{7.2}$$

where dt is the step size and L is the length of the simulation. The initial conditions for all experiments were chosen to be:

$$x = -8, \quad y = 8 \quad , z = 27 \tag{7.3}$$

Finally, as we need a univariate signal for the phase space reconstruction and to test our methods, we must choose one of the three variables. We chose $x(t)$ for all experiments. Figures 7.15 and 7.16 depict all coordinates of the Lorenz system and the corresponding phase space portrait.

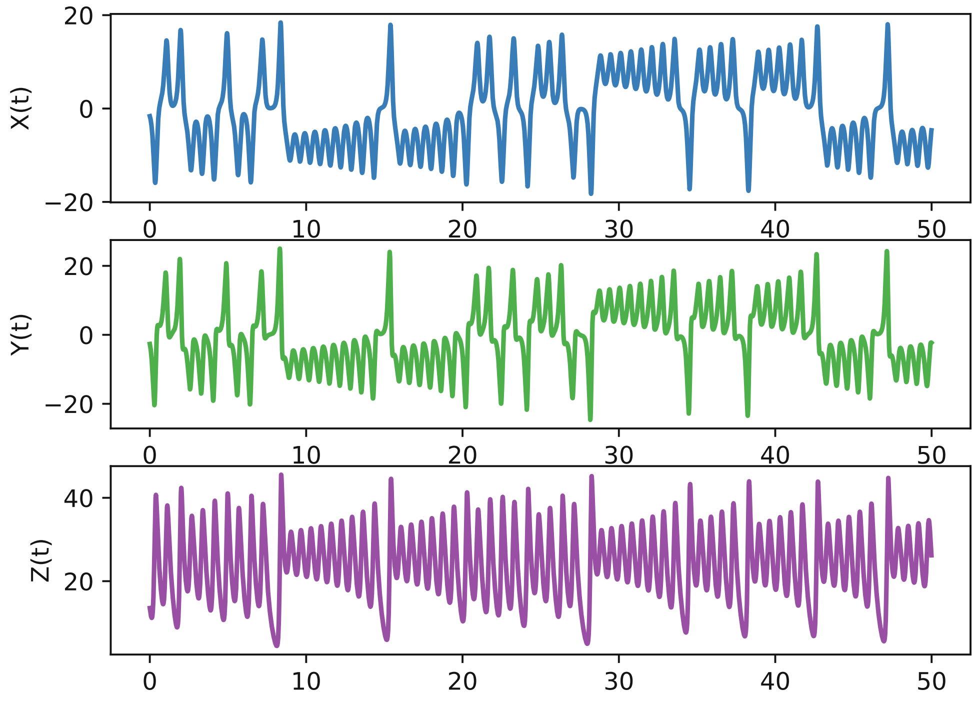


FIGURE 7.15: Time series plot for the separate coordinates of the Lorenz system, we generated 100000 time steps with $dt = 0.01$ and plotted the final 5000 data points.

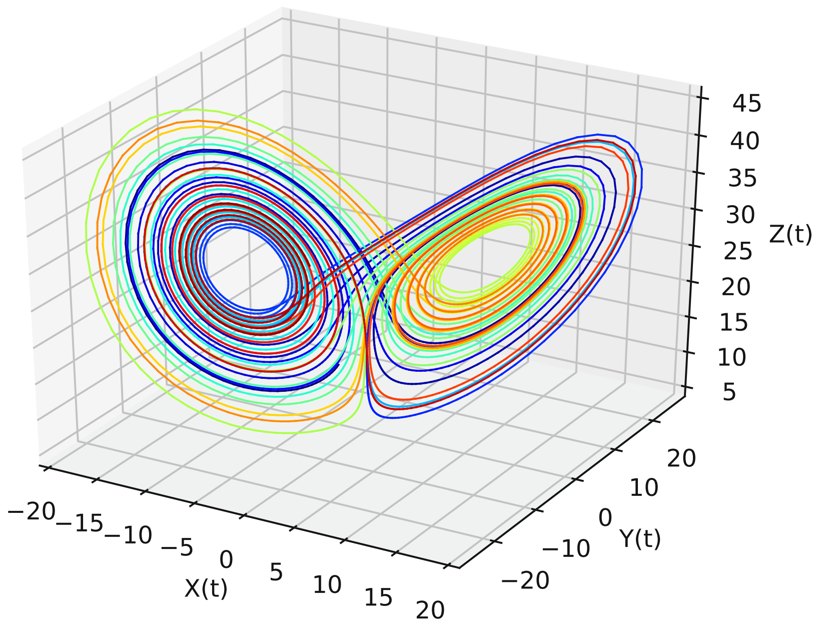


FIGURE 7.16: Attractor plot for the Lorenz system, we generated 100000 time steps with $dt = 0.01$ and plotted the final 5000 data points.

Part II. Applications and Results

Chapter 8

Fractal Interpolation and Neural Network Time Series Predictions

This chapter is a precursor to the later chapters, providing initial tests on a combined approach of fractal interpolation and neural networks. The contents of this chapter are published in [28]:

Sebastian Raubitzek and Thomas Neubauer. A fractal interpolation approach to improve neural network predictions for difficult time series data. Expert Systems with Applications, 169:114474, 2021. ISSN 0957-4174. doi: 10.1016/j.eswa.2020.114474. URL http://www.sciencedirect.com/science/article/pii/S0957417420311234. Visited on 2023-04-20.

The presented research discusses the applicability of fractal interpolation to improve long short term memory (LSTM) neural network time series predictions. We chose an LSTM neural network architecture because this type of neural network was invented to deal with time series data, [87]. For the fractal interpolation, we used the method described in Section 5.1.

We aim to show the applicability of fractal interpolation to neural network time series predictions and test it against linear interpolation, [101]. Linear interpolation was used in combination with measures of signal complexity and machine learning approaches for time series data in work done by Karaca et al. [42, 43].

First, the procedure is to interpolate the discussed time series using fractal and linear interpolation. Next, we analyze the data's signal complexity. I.e., we discuss the original, the fractal, and the linear interpolated data sets using R/S analysis, the fractal dimension of a time series, and the spectrum of Lyapunov exponents for experimental data. Next, we perform neural network train and test fits/predictions on the data.

We are using the following four data sets in this chapter:

- Monthly international airline passengers, Section 7.1;
- Monthly shampoo sales, Section 7.9;
- Annual wheat yields, Section 7.6;
- Annual maize yields, Section 7.7;

We chose the data sets such that one is a known test data set with visible seasonality, i.e., the monthly international airline passengers data set, and one is a test data set without a visible seasonality, i.e., the shampoo sales data set. Additionally, we chose two challenging data sets from Austrian agriculture, i.e., the annual maize and wheat yield data sets. We consider the latter two data sets challenging because of their yearly frequency and because these data sets are a whole country's average. Thus, different climate and regional differences are averaged out. Therefore we expect a somewhat random behavior, i.e., future data points do not depend on past ones, similar to a fractional Brownian motion.

The presented results show that fractal interpolation can effectively increase the accuracy of the employed long short term memory neural networks on a given data set compared to the non-interpolated case. Further, the results show that the discussed fractal interpolation can, in some cases, outperform a basic linear interpolation as the presented neural network predictions are slightly better for the fractal interpolated time series data. However, given the results of the following chapters, we cannot find profound evidence for fractal interpolation as the superior method to be used for neural network time series predictions. Still we need to mention that there is, as of the writing of this thesis, a very recent publication on data augmentation and neural network time series predictions for univariate time series data by Semenoglou et al. [26]. Also, as stated in [102], there are no comprehensive comparative studies on interpolation techniques. Further, as far as the author knows, there are no comprehensive comparative studies on the applicability of interpolation techniques for neural network time series predictions. However, to avoid confusing the reader, machine learning and neural network techniques are capable of interpolating data, [102], but we are explicitly talking about interpolation techniques to be used for improving machine learning and neural network approaches.

The idea of the employed fractal interpolation is to perform different interpolations, i.e., introduce fluctuations with different magnitudes, for different subintervals of a single interpolated time-series data. This means we want the interpolated time series data to have larger fluctuations on smaller scales for intervals with overall larger fluctuations and smaller fluctuations on smaller scales for intervals with overall smaller fluctuations, as we expect most non-model data sets to have multifractal characteristics, [74]. The number of additional data points was set to 17. A discussion on choosing the right number of interpolation points is given in Chapter 13.

The employed fractal interpolation approach is discussed in Section 5.1, all data sets under study are linked above and a full description is given in Chapter 7.

Section 8.1 discusses the signal complexities of all data sets under study. Section 8.2 describes the employed neural network architectures, how the predictions are performed, and shows the corresponding results. All findings are discussed in Section 8.3 and summed up in Section 8.4.

8.1 Complexity Measurements

All calculated complexity properties for the discussed time series can be found in Table 8.1. Here we need to mention that measures of signal complexity cannot give meaningful and/or interpretable results for such short time series as the non-interpolated time series in this chapter. Still, we present the values that the employed algorithms give and discuss these findings. The reader is referred to Section 13.4 and Appendix E, where we discuss the problem of estimating the signal complexity of short time series data and the corresponding findings in the context of the whole thesis.

When comparing the Hurst exponent and the fractal dimension of the original and the interpolated time series, all interpolated time series have a higher Hurst exponent and a lower fractal dimension. This results from the fact that both concepts can be linked using R/S analysis by $D_f \approx 2 - H$, [76, 77], where D_f is the fractal dimension, and H is the Hurst exponent. Since we use different algorithms to calculate the fractal dimension and the Hurst exponent, this identity is only approximately true. We use R/S analysis to calculate the Hurst exponent and Higuchi's algorithm to calculate the fractal dimension, [7], [46].

Still, we must discuss this relation between the fractal dimension and the Hurst exponent. Here we're closely following the work by [103]. First, the original relation is given by

$$D_f + H = n + 1 \quad , \tag{8.1}$$

for a self-affine surface in an n-dimensional space. However, as pointed out by [103] this relation does not hold for non-self-affine models/data. Thus, as the real-life data we discuss here is never truly self-affine, and because we're using different algorithms to calculate the fractal dimension and the Hurst exponent, this linear relation can only approximately be true. Thus we changed it to $D_f \approx 2 - H$. Where $n = 1$ for time series data. Also, as recommended in [103], one should calculate fractal dimension and Hurst exponent separately, using different algorithms, as the linear relation breaks down if self-affinity is violated. Which we did for comparison. Finally, given that Equation 8.1 approximately holds for some of the interpolated data under study, one could conclude that the interpolated data has some aspects of self-affinity to it. However, taking a closer look at this shows that for the shampoo sales data, this relation is violated for the non-interpolated time series data as $D_f + H = 2.3531 \not\approx 2$, for the fractal interpolated time series data, we observe $D_f + H = 2.1315 \not\approx$ and for the linear interpolated data $D_f + H = 1.9991 \approx 2$, which makes no sense because we consider the linear interpolated data to be not self-affine at all. We observe similar behavior for all data sets, i.e., that the linear interpolated time series best fulfills this relation. We conclude that, first, interpolation changes the nature of the signal. And second, either or both, the employed concepts and algorithms do not work for time series data of this length, and that fractal interpolation does not generate a self-affine time series.

Still, both measures, show that all interpolated time series are more persistent ones, or in terms of the Hurst exponent, data with more long term memory. It is expected that time series with these characteristics, i.e., a lower fractal dimension and a higher Hurst exponent, can be forecast with higher accuracy. However, for the non-interpolated time series data, the presented results cannot be interpreted as the scaling behavior of the time series, as one cannot identify a scaling behavior from time series data as short as the presented ones. Again, Section 13.4 provides an in-depth discussion on this topic.

Still, this increased Hurst exponent is to some degree the result of interpolating two consecutive data points. Persistent successive data points will likely hit a set endpoint given a fixed start. Thus we expect this behavior is inherent to most interpolations. This is shown for varying numbers of interpolation points for both the linear and the fractal interpolation in Appendix B.2. It is apparent that both the fractal and the linear interpolation increase the Hurst exponent with increasing numbers of interpolation points for all data sets discussed in Chapter 9, see Figures 9.2, B.1, B.3, B.5 and B.7.

Regarding the difference between the linear and the fractal interpolated time series data, one observes that the linear interpolated time series data have a lower fractal dimension and a lower Hurst exponent than the fractal interpolated data, which is contradictory. But, again, given that we calculated those measures with different algorithms, this again proves the previous statement regarding the link between Hurst exponent and fractal dimension. [15], [44].

Finally, we cannot give a meaningful interpretation of the spectrum of Lyapunov exponents on the topic as the here non-interpolated times series data need to be longer to do so. However, the interpolated time series data suggest that the fractal interpolation produced highly chaotic data with two positive Lyapunov exponents referred to as hyperchaos in the literature [52].

Table 8.1: Complexity measurements/properties

Data	**Hurst exponent**	**Fractal Dimension**	**Lyapunov spectrum**
Shampoo sales	0.5894	1.7637	-0.0262 -0.0545 -0.1633 -0.1738
Shampoo sales fractal interpolated	0.9252	1.2063	0.0971 0.0102 -0.0784 -0.2158
Shampoo sales linear interpolated	0.9009	1.0982	2.9902 0.9656 -0.2968 -2.0357
Airline passengers	0.3996	1.7658	0.0160 0.0103 -0.0242 -0.0750
Airline passengers fractal interpolated	0.9521	1.2401	0.0926 0.0162 -0.0746 -0.2317
Airline passengers linear interpolated	0.9298	1.033	2.5814 0.7693 -0.3465 -1.8434
Wheat yields	0.9063	1.7145	-0.0156 -0.0379 -0.0286 -0.2699
Wheat yields fractal interpolated	0.8982	1.2269	0.0886 0.0099 -0.0589 -0.2374
Wheat yields linear interpolated	0.8758	1.0921	2.5700 0.8839 -0.3784 -2.0337
Maize yields	0.8591	1.7556	-0.0260 -0.0263 -0.0969 -0.2620
Maize yields fractal interpolated	0.8983	1.2050	0.0868 0.0061 -0.0671 -0.2182
Maize yields linear interpolated	0.8809	1.0941	2.7797 0.9551 -0.3495 -2.0198

8.2 LSTM Neural Networks Predictions

A long short term memory (LSTM) neural network [87] was applied to analyze the four data sets. Each time series was split into two parts, one for training the data set and one to test the performance on previously unknown data, whereas the training part makes up two-thirds and the unknown part the last third of the data, chronologically. The accuracy of the predictions is then for the interpolated and the non-interpolated dataset, meaning that if a dataset was interpolated, we also take the interpolated data set for reference. We admit that this is problematic in terms of the comparison of predictions. However, the plots suggest that the neural networks do not only learn the interpolation but also seem to depict the behavior of the original data. The results of this chapter are to be understood as preliminary results for the advanced prediction techniques discussed in later chapters. In later chapters, however, we only evaluate non-interpolated data points.

8.2.1 Data Preprocessing

Since the performance of neural networks can be improved with proper data preparation, the following procedures were applied to all data sets.

First, the data $X(t)$ defined at discrete time intervals $t = v, 2v, 3v, ..., kv$, was scaled so that $X(t) \in [-1, 1]\,, \forall t$[1]. This was done for all 4 data sets. Second, the performance was increased when the data was stationary, i.e., a linear fit was subtracted at each time step from the data. This was done for all data sets except the wheat yield data since it worsened the results of the predictions.

8.2.2 LSTM Neural Networks

Fits on unknown data of the original and the fractal interpolated time series data was done using a long short term memory (LSTM) neural network, see Section 6.1.2.

The algorithm was optimized by observing the training loss curve and the overall performance. Further, we used a basic LSTM implementation with one hidden layer in accordance with [104]. Using `adam` as an optimizer, we observed decaying learning rates for all data sets. For the loss function we used `mean_squared_error`. The batch size was set to 1, and verbose was set to 2.

[1]In this chapter we used the interval $[-1, 1]$ because of the tanh activation function. The later chapters are using varying intervals because of varying activation functions.

An existing `Keras 2.3.1` implementation was used for this research. The following list describes in detail the architecture of the LSTM-layer:

- `activation="tanh"`
- `recurrent_activation="sigmoid"`
- `use_bias=True`
- `kernel_initializer="glorot_uniform"`
- `recurrent_initializer="orthogonal"`
- `bias_initializer="zeros"`
- `unit_forget_bias=True`
- `kernel_regularizer=None`
- `recurrent_regularizer=None`
- `bias_regularizer=None`
- `activity_regularizer=None`
- `kernel_constraint=None`
- `recurrent_constraint=None`
- `bias_constraint=None`
- `dropout=0.0`
- `recurrent_dropout=0.0`
- `implementation=2`
- `return_sequences=False`
- `return_state=False`
- `go_backwards=False`
- `stateful=False`
- `time_major=False`
- `unroll=False`

In Table 8.2 the varying parameters for the neural network for each data set can be found.

Table 8.2: LSTM architecture

Data	**Epochs**	**Hidden Layers**	**Number of input data points**
Shampoo sales	89	2	7
Shampoo sales fractal interpolated	13	25	7
Shampoo sales linear interpolated	13	25	7
Airline passengers	50	10	15
Airline passengers fractal interpolated	2	35	20
Airline passengers linear interpolated	2	35	20
Wheat yields	120	10	2
Wheat yields fractal interpolated	15	25	100
Wheat yields linear interpolated	15	25	100
Maize yields	100	10	2
Maize yields fractal interpolated	5	30	70
Maize yields linear interpolated	5	30	70

8.2.3 Error Analysis

For the error analysis all data was normalized so that all $X(t)$, $\hat{X}(t) \in [-1, 1]$ when the RMSE was calculted. The errors for all train fits and test fits can be found in Table 8.3.

8.3 Results/Discussion

The results show that the accuracy of an LSTM neural network on difficult time series data can significantly be improved using a fractal or a linear interpolation method. We observed this regarding the predictions' errors (Table 8.3). For all data sets, the interpolated approaches outperformed the regular ones. On the training data, the linear interpolated approach performed best for all data sets. The fractal interpolated approach performed best for the unknown data except for the airline passengers data set.

All fits on the training data, the test data for all data and the corresponding interpolated data are depicted in Figures 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7, 8.8, 8.9, 8.10, 8.11 and 8.12. The dashed purple line separates training data and test data.

Before further interpreting these results, we need to put them into perspective. First, the prediction on the interpolated datasets is validated for different time steps as for the non-interpolated time series data. This does not allow for a direct comparison of the predictability. Still, our results show that interpolated time series data provide better proximity fits than non-interpolated. This is primarily due to the fact that the majority of the predicted data points are interpolated ones. These interpolated data points do not provide meaningful results for the maize and wheat yield data sets. Still, data augmentation, and in this case, data interpolation, can increase the accuracy of neural networks on time series data as it increases the probability of predicting the correct data point in the near vicinity of the input. Thus these errors are not direct accuracy comparisons but compare different problems, and the reduced errors tell us that we shifted the problem away from predicting the next data point. However, we then need to find out if the data still depicts the information of the original time series, which is addressed in the later chapters of this thesis and summarized in Chapters 13.

The plots show that the non-interpolated airline passengers data set can be reproduced by the LSTM neural networks. However, The results for all other non-interpolated data sets are far off, and we conclude that the neural network cannot reproduce the inherent behavior of these data sets.

The fractal interpolated data set results are improved for all data sets. And for the shampoo, wheat, and maize yields, we consider the prediction to be drastically improved. However, to some degree, this results from increased persistency. We discussed this previously in Section 8.1 for the Hurst exponent. The increased persistency, in this case, means that consecutive interpolated data points are closer to each other. Thus the neural network has increased accuracy for denser and more persistent data sets as predicted data points for these data sets tend to be close to the previous data point. We

Chapter 8. Fractal Interpolation and Neural Network Time Series Predictions

see this behavior for all interpolated data sets. The test fits are improved, i.e., closer to the actual curve.

TABLE 8.3: RMSE for each data set

Data	Train data error	Test data error
Shampoo sales	0.1860	0.5839
Shampoo sales fractal interpolated	0.0236	0.0654
Shampoo sales linear interpolated	0.0132	0.0899
Airline passengers	0.0307	0.0566
Airline passengers fractal interpolated	0.0120	0.0237
Airline passengers linear interpolated	0.0058	0.0185
Wheat yields	0.1957	0.3298
Wheat yields fractal interpolated	0.0320	0.0849
Wheat yields linear interpolated	0.0212	0.0892
Maize yields	0.1382	0.3062
Maize yields fractal interpolated	0.0248	0.0558
Maize yields linear interpolated	0.0177	0.0756

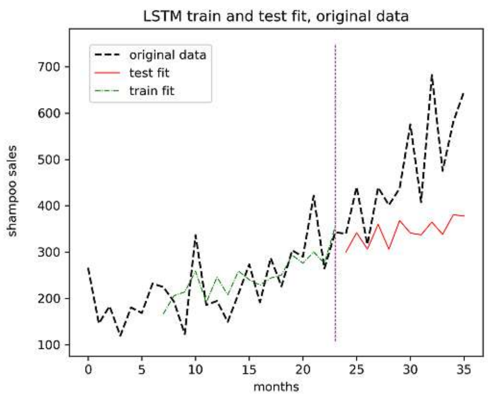


FIGURE 8.1: Shampoo sales

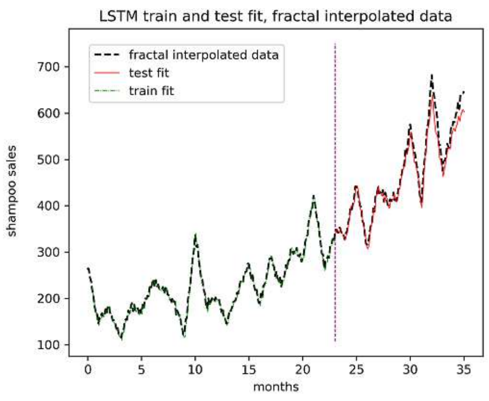


FIGURE 8.2: Shampoo sales, fractal interpolated

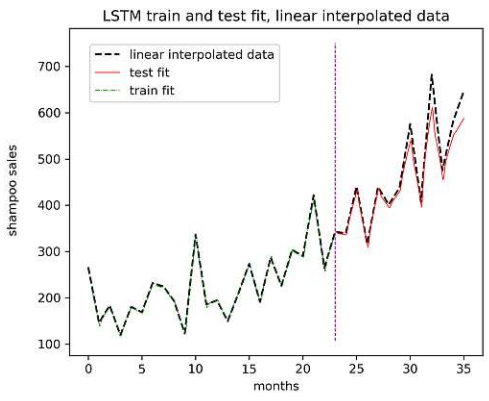


FIGURE 8.3: Shampoo sales, linear interpolated

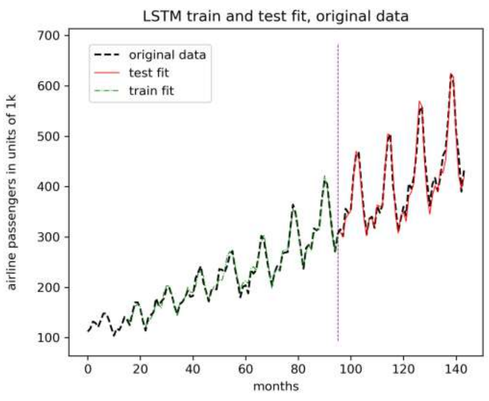


FIGURE 8.4: Airline passengers

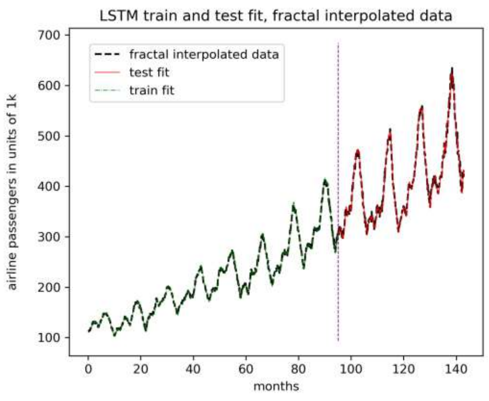


FIGURE 8.5: Airline passengers fractal interpolated

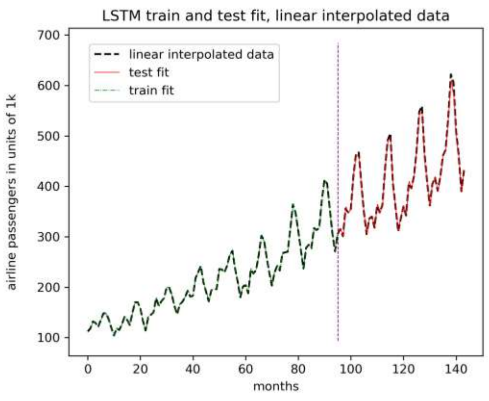


FIGURE 8.6: Airline passengers fractal interpolated

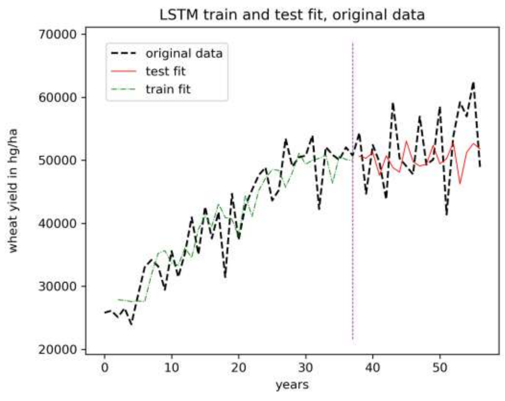


FIGURE 8.7: Austrian wheat yields

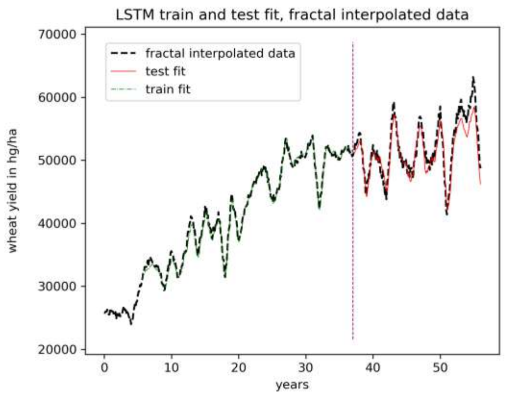


FIGURE 8.8: Austrian wheat yields, fractal interpolated

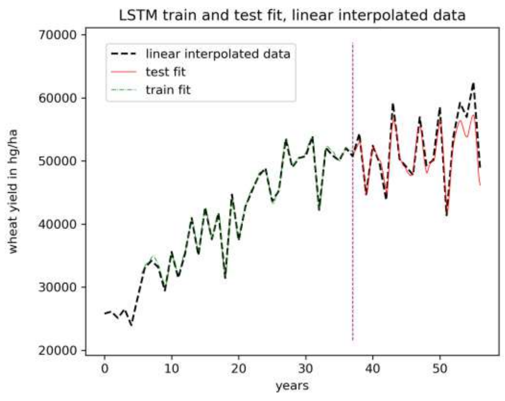


FIGURE 8.9: Austrian wheat yields, linear interpolated

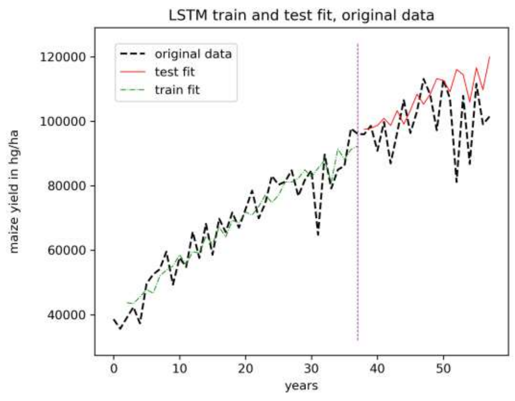


FIGURE 8.10: Austrian maize yields

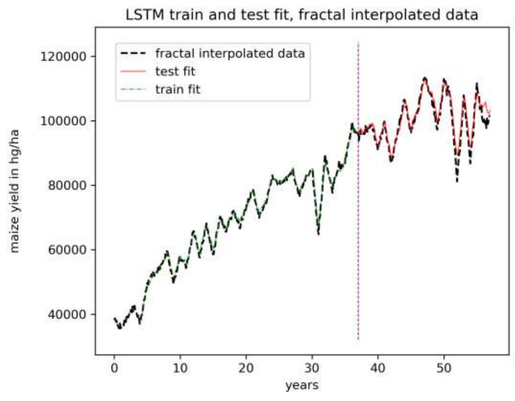


FIGURE 8.11: Austrian maize yields, fractal interpolated

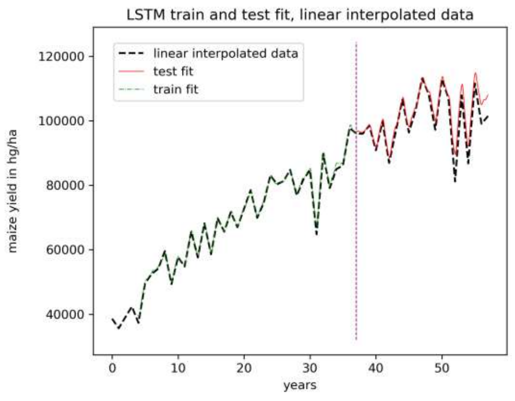


FIGURE 8.12: Austrian maize yields, fractal interpolated

8.4 Summary

This Section briefly summarizes all findings and results of this Chapter.

- The accuracy in terms of the RMSE of an LSTM neural network on short/sparsely sampled data sets can drastically be improved using fractal or linear interpolation to increase the overall amount of data. However, this shifts the problem to predicting interpolated data points. This direct comparison does not tell us if the neural network learned the information in the original data set. We address this problem in the following chapters by comparing only to actual data points using autoregressive predictions; see Chapters 9 and 11.
- The developed fractal interpolation approach slightly outperformed the linear interpolation approach in terms of a lower RMSE on unknown data. However, given the experiments at the end of the presented thesis, we cannot identify fractal interpolation as the overall better interpolation technique to improve the predictability of univariate time series data.
- The Hurst exponent (Table 8.1) for all time-series is increased when performing an interpolation. This supports the findings of increased predictability, as a Hurst exponent of ≈ 0.5 indicates random behavior. An increased Hurst exponent indicates a more persistent behavior of the underlying dynamics.
- The spectrum of Lyapunov exponents (Table 8.1) indicates chaotic behavior for all interpolated data, contrary to the observed increase in predictability. However we need to point out, that due to very low amounts of data points, the spectrum of Lyapunov exponents is not applicable to the non-interpoalted data (See Section 13.4).
- Ref. [77] gives a relation between the fractal dimension and the Hurst exponent as $D_f = 2 - H$ which is approximately true for the discussed data sets (Table 8.1). This is because the concepts of the fractal dimension and the Hurst exponent are similar for time series data, as both take into account the fluctuations present in the data. Thus the fractal dimension also indicates a less complicated behavior for the interpolated time series and thus increased predictability.
- Fractal and linear interpolation significantly change the nature of the interpolated signal with respect to the Hurst exponent, fractal dimension, and the spectrum of Lyapunov exponents.

Chapter 9

Randomly Parameterized Neural Networks and Complexity Prediction Filters

This chapter presents and sums up the findings of the author's publication [32]:

Sebastian Raubitzek and Thomas Neubauer. Taming the Chaos in Neural Network Time Series Predictions. Entropy, 23(11), 2021. ISSN 1099-4300. doi: 10.3390/e23111424. URL https://www.mdpi.com/1099-4300/23/11/1424. Visited on 2023-04-20.

The publication and, consequently, this chapter provide results and a discussion on the employed and developed randomly parameterized LSTM neural networks approach, which is discussed in Section 6.1.3. We continue the work done in the previous chapter (Chapter 8) such that we are using a fractal interpolation in combination with LSTM neural networks. In contrast to the previous one, this chapter focuses on autoregressive predictions. Thus we aim to show the applicability of fractal interpolation, the proposed randomly parameterized neural network approach, and the corresponding prediction filters. Also, similar to the previous chapter, we analyze the original and the interpolated data sets using five measures of signal complexity, i.e., The largest Lyapunov exponent, The Hurst exponent, Fisher's information, SVD entropy, and Shannon's entropy.

However, contrary to the previous chapter, we evaluate our results only on non-interpolated data points to provide comparable results.

We chose five test data sets from the *Time Series Data Library*, [94], for this experiment:

1. Monthly international airline passengers, see Section 7.1

2. Monthly car sales in Quebec, see Section 7.4

3. Monthly mean air temperature in Nottingham Castle, see Section 7.2

4. Perrin Freres monthly champagne sales, see Section 7.3

5. CFE specialty monthly writing paper sales, see Section 7.8

The chosen data sets differ in length and complexity. Each of these data sets has a reoccurring seasonal behavior and, as the baseline predictions show (Section B.1), can be reasonably predicted using a basic LSTM neural network with one hidden layer. Here, we want to point out that, compared to a regular neural network implementation, the randomly parameterized neural networks do not need to be parameterized to give a prediction, hence *randomly parameterized*, but instead are filtered after producing many forecasts.

The developed scheme consists of generating linear and fractal interpolated (Section 5.1) time series for each data set. Next, predicting these data sets using randomly parameterized LSTM neural networks. And finally, filtering this multitude of predictions based on their signal complexity to improve the overall accuracy.

For the fractal and linear interpolation, we use the following numbers of additional data points, i.e., between each two original data points, $N_I = \{1, 3, 5, 7, 9, 11, 13, 15, 17\}$. We chose the numbers of interpolation points to multiply each data set's data points to cover a range of different interpolation points for both the complexity and the predictability analysis. We use 17 as the upper threshold as the calculations start to get very expensive with increasing numbers of interpolation points. A discussion on choosing the right number of interpolation points is given in Section 13.1. Figure 9.1 depicts the whole procedure.

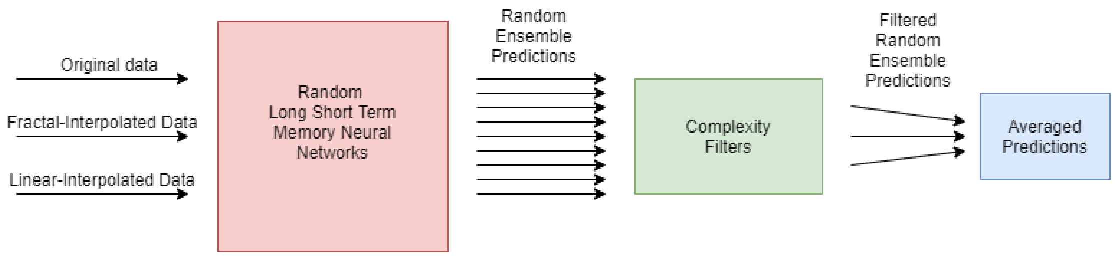

Original data
Fractal-Interpolated Data
Linear-Interpolated Data
Random Long Short Term Memory Neural Networks
Random Ensemble Predictions
Complexity Filters
Filtered Random Ensemble Predictions
Averaged Predictions

FIGURE 9.1: Schematic depiction of the filtering process. The whole pipeline is applied, first, to the original non-interpolated data, second, the fractal interpolated data, and third, the linear interpolated.

9.1 Signal Complexity

We applied five measures of signal complexity to the original data sets. The results can be found in Table 9.1. We briefly discuss the complexities for each measure separately, as an in-depth discussion with regards to their predictability can be found in Section 8.3:

- **The Hurst exponent (Section 3.1):** With a Hurst exponent of 0.7988, the most persistent data set is monthly car sales in Quebec. According to [37], we expected that time series with a Hurst exponent comparatively close to one can be predicted with higher accuracy than ones with a value close to 0.5, as the later ones are considered more random. The data sets under study are three persistent ones, i.e., with a Hurst exponent larger than 0.5. Contrary to that, two are anti-persistent ones with a Hurst exponent below 0.5.

- **The largest Lyapunov exponent (Section 3.3):** All largest Lyapunov exponents of all time series data under study are positive, just as we would expect from chaotic or complex real-life data. The data set with the highest value is monthly car sales in Quebec. As the Lyapunov exponents of experimental time series data serve as a measure for predictability, we suggest this data set, therefore, to be forecast with low accuracy qualitatively, which is contradictory to the previous discussion on the Hurst exponent.

- **Fisher's information (Section 3.6):** Fisher's information is expected to behave contrary to entropy measures since it is a measure for order/quality, which we observe when comparing Fisher's information to SVD entropy, [62]. Thus we expect data sets with a comparatively high value of Fisher's information to be forecast qualitatively more accurately. The data set with the highest value of Fisher's information is the monthly international airline passengers data set. The lowest value is found for the Perrin Freres monthly champagne sales data set. Thus Fisher's information has the largest value for the monthly international airline passengers data set, and the corresponding value for the SVD entropy is the lowest among all data sets. This is because, just like Fisher's information, SVD entropy is based on Single Value Decomposition (SVD) [105]. In contrast, Shannon's entropy, which is also an entropy measure but not one based on Single Value decomposition, differs in its behavior.

  As Fisher's information is based on SVD, we need to find two parameters before applying it; the time delay τ and the embedding dimension d_E. Here we used the values obtained by the method of average mutual information and the false nearest neighbors algorithm, which is discussed in Appendix A.

- **SVD entropy (Section 3.5):** The largest value of SVD entropy is shown by the Perrin Freres monthly champagne sales data set, which has, just as expected, the lowest value of Fisher's information. As it has a comparatively high SVD entropy value, we expect the Perrin Freres champagne sales data set not to be predictable very accurately qualitatively. Again, we chose the embedding dimension and time delay using the method of mutual information and the false nearest neighbors algorithm, as discussed in Appendix A.

- **Shannon's entropy (Section 3.4):** Shannon's entropy is based on the frequency of occurrence of a specific value. As we deal with non-integer-valued complex data sets, we expect Shannon's entropy not to be of much use for analysis. Shannon's entropy's highest value is found for the monthly mean temperature in Nottingham Castle data set. Since reoccurring temperature distributions possess a higher regularity than, e.g., airline passengers, this explains the corresponding value.

TABLE 9.1: Complexities of the original data sets.

	Hurst Exponent	**Largest Lyapunov Exponent**	**Fisher's Information**	**SVDEntropy**	**Shannon's Entropy**
Monthly international airline passengers	0.4233	0.0213	0.7854	0.3788	6.8036
Monthly car sales in Quebec	0.7988	0.0329	0.5965	0.5904	6.7549
Monthly mean air temperature in Nottingham Castle	0.4676	0.0069	0.6617	0.5235	7.0606
Perrin Freres monthly champagne sales	0.7063	0.0125	0.3377	0.8082	6.6762
CFE specialty monthly writing paper sales	0.6830	0.0111	0.5723	0.6138	7.0721

9.1.1 Complexity Analysis

We compared the complexities of all time series data under study, i.e., for different interpolation techniques and numbers of interpolation points.

The results are shown in Figures 9.2–9.4 for the monthly international airline passengers data set. The plots for the other data sets are collected in Appendix B.2. Note that, in each of the plots, the blue line, i.e., the complexity of the non-interpolated time series, is not a plot depending on the number of interpolation points but a constant since there is only one data set, the one with zero interpolation points, and plotted as a line for reference. In addition, those original complexities are contained in Table 9.1.

For this study, we observe the following behavior:

- The Hurst exponent of the fractal and the linear interpolated data sets behave very similarly for the monthly international airline passengers data set, see Figure 9.2. We observe similar behavior for the other data sets as well, see Appendix B.2. Though the Hurst exponent is initially lower for the fractal interpolated data for some data sets, the Hurst exponent does not differ significantly between fractal and linear interpolated time series data. In addition, adding more interpolation points increases the Hurst exponent and makes the data sets more persistent.

- The Largest Lyapunov exponents of the fractal interpolated data are much closer to the original data than those for the linear interpolated data; see Figure 9.3. We observe the same behavior for all data sets; see Appendix B.2;

- Fisher's information for the fractal interpolated data set is closer to that of the original data set (see Figure 9.2). We observe the same behavior for all data sets, as can be seen in Appendix B.2;

- Just as discussed in the previous section, SVD entropy behaves contrary to Fisher's information. In addition, SVD entropy of the fractal interpolated time series is closer to that of the non-interpolated time series; see Figure 9.4. The same behavior and, specifically, the behavior contrary to that of Fisher's information can be observed for all data sets under study; see Appendix B.2;

- Shannon's entropy increases with the number of interpolation points. This can be explained as follows: As more data points are added, the probability of hitting the same value increases. However, this is just what Shannon's entropy measures. For small numbers of interpolation points, Shannon's entropy of the fractal interpolated time series data is closer to the original complexity than the linear interpolated time series data. Shannon's entropy performs very similarly for large numbers of interpolation points, not to say overlaps, for the fractal and linear interpolated time series data. This behavior can be observed for all data sets, see Figure 9.3 and Appendix B.2.

Summing up our findings of the complexity analysis above, we find that:

- The fractal interpolation captures the original data complexity better than the linear interpolation. We observe a significant difference in their behavior when studying SVD entropy, Fisher's information, and the largest Lyapunov exponent. This is especially true for the largest Lyapunov exponent, where the behavior completely differs. The largest Lyapunov exponent of the fractal interpolated time series data stays mostly constant or behaves linearly. The largest Lyapunov exponent of the linear interpolated data behaves approximately like a sigmoid function and, for some data sets, even decreases again for large numbers of interpolation points.

- Both Shannon's entropy and the Hurst exponent seem not suitable for differentiating between fractal- and linear interpolated time series data.

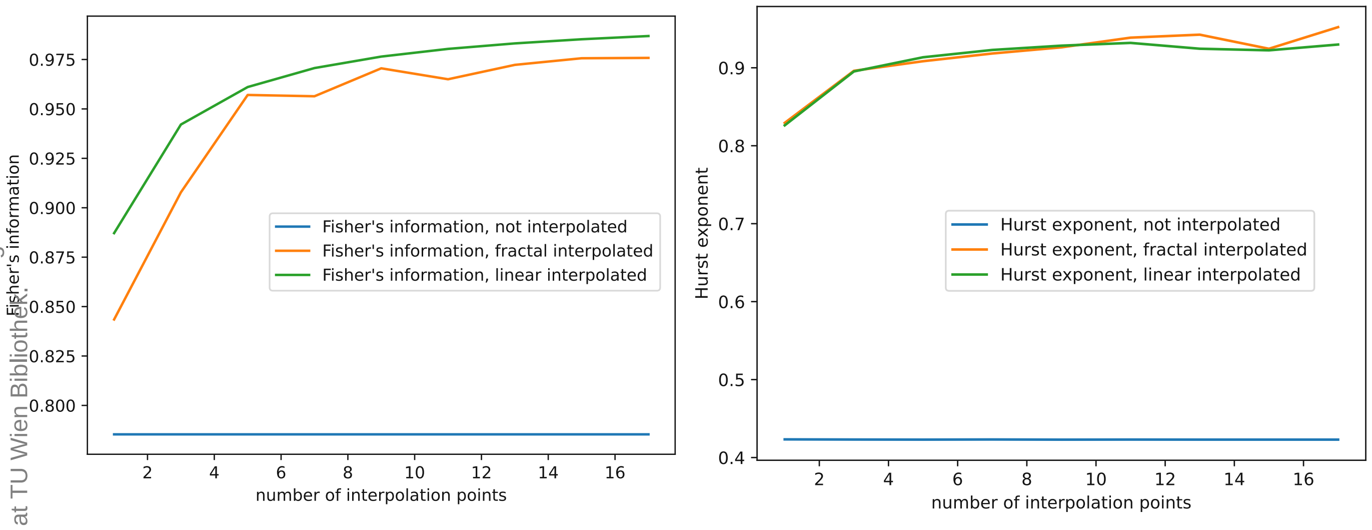


FIGURE 9.2: Plots for Fisher's information and the Hurst exponent depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear interpolated data. Monthly international airline passengers data set.

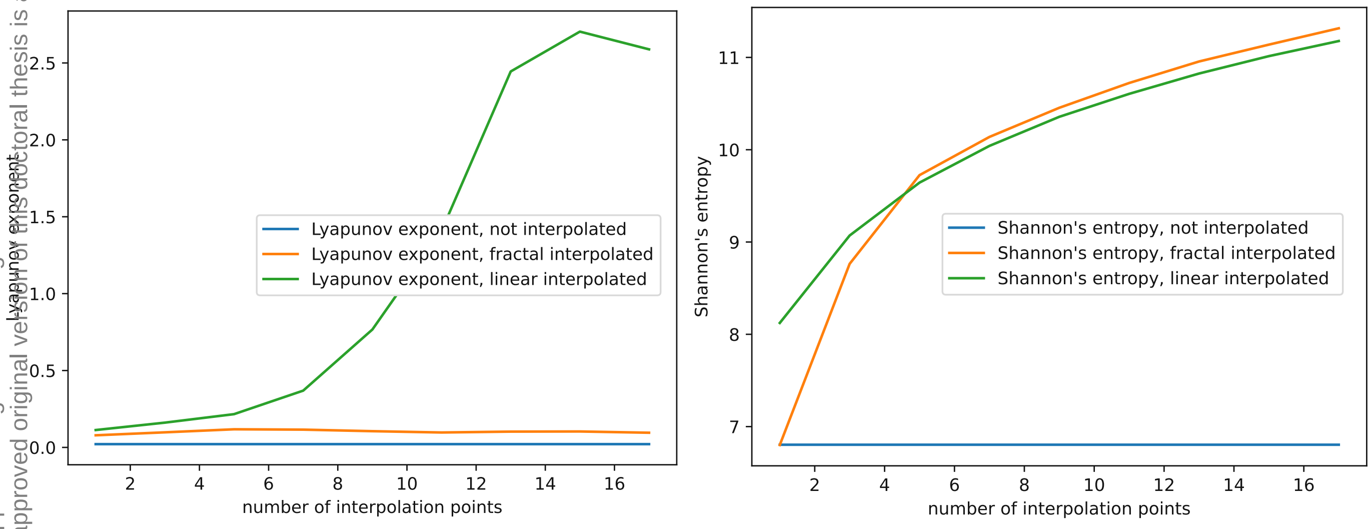


FIGURE 9.3: Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear interpolated data. Monthly international airline passengers data set.

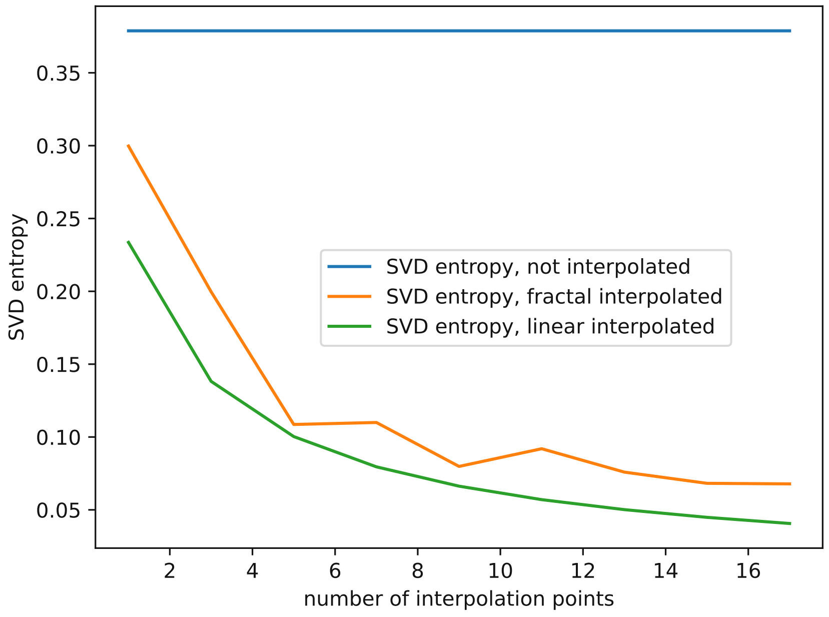


FIGURE 9.4: Plot for the SVD entropy depending on the number of interpolation points, for the non-interpolated, the fractal interpolated and the linear interpolated data. Monthly international airline passengers data set.

9.2 LSTM Ensemble Predictions

For each data set, we employed ensembles of randomly parameterized LSTM neural networks, see Sections 6.1.2 and 6.1.3. Here we use the first of the two described architectures. The idea is not to optimize the neural networks but to generate many different ones. I.e., train and predict the data under study using many different neural networks and then average them to obtain a final result/prediction. Five hundred of these networks were created, trained, and generated a prediction for the unknown data. In a later step, these predictions are filtered based on their complexity. This filtering process improves the predictions drastically.

9.2.1 Data Preprocessing

All data were preprocessed using the following two data manipulations:

First, all data were scaled to be within unit interval, i.e., $[0, 1]$. And some data sets were detrended by subtracting a linear fit. This is noted for each data set in Chapter 7.

After preprocessing, all data sets were split to use the first 70% as a training data set and the remaining 30% to validate/test the results for the monthly international airline passengers data set and 80% to 20% for the remaining data sets.

9.3 Complexity Filters/Data Postprocessing

As previously mentioned, the randomly parameterized LSTM neural networks produced 500 predictions for each time series data[1] The results of the averaged non-filtered predictions can be found in Tables B.9 and B.10.

These unfiltered predictions are still far off a good forecast because many bad predictions are among them. Here the hypothesis is that good predictions have a similar complexity as the original (fractal or linear interpolated, respectively) time series. Thus the predictions for each time series data are filtered regarding their complexity, i.e., only the top 1% with complexity close to that of the original data is kept. The filters reduced the number of predictions from 500 to 5 for each data set.

All complexity measures are discussed in detail in Chapter 3. The complexity filters for each measure of signal complexity are constructed as follows:

1. **Hurst exponent filter:** For each prediction, the Hurst exponent is calculated. Next, we compare the so obtained Hurst exponent to the Hurst exponent of the training data set such that:

$$\Delta H = \left| \hat{H} - H \right| \quad . \tag{9.1}$$

   Here H is the Hurst exponent of the training data set and $\hat{H}$ is the Hurst exponent of a prediction. To improve the averaged predictions, only the ensemble predictions with Hurst exponents close to that of to the training data set, i.e., with low ΔH are kept, and the others are discarded.

2. **Lyapunov exponents filter:** We calculate the first four Lyapunov exponents of each ensemble prediction and compare them with the Lyapunov exponents of the training data set,

$$\Delta L = \sum_{i=1}^{4} \left| \hat{L}_i - L_i \right| \quad . \tag{9.2}$$

   Here L_i is the ith Lyapunov exponent of the training data set, and $\hat{L}_i$ is the ith Lyapunov exponent of a prediction. Finally, we discard all predictions with high ΔL and keep only the ones with low ΔL.

3. **Fisher's information filter:** For each prediction, we compute Fisher's information. Next, we compare the outcome with the training data's Fisher information:

$$\Delta I_{\text{Fisher}} = \left| \hat{I}_{\text{Fisher}} - I_{\text{Fisher}} \right| \quad . \tag{9.3}$$

[1] I.e. For each number of different interpolation points and both the linear and fractal interpolated data sets.

Here I_{Fisher} is Fisher's information of the training data set, and $\hat{I}_{\text{Fisher}}$ is Fisher's information of a prediction. Finally, we discard all predictions with high ΔI_{Fisher} and keep the ones with the low ΔI_{Fisher}.

4. **SVD entropy filter:** For each prediction, the SVD entropy is calculated. Then, we compare the obtained entropy with the entropy of the training set:

$$\Delta H_{\text{SVD}} = \left| \hat{H}_{\text{SVD}} - H_{\text{SVD}} \right| \quad . \tag{9.4}$$

   Here H_{SVD} is the SVD entropy of the training data set, and $\hat{H}_{\text{SVD}}$ is the SVD entropy of a prediction. Finally, we discard predictions with high ΔH_{SVD}, and keep only the ones with low ΔH_{SVD}.

5. **Shannon's entropy filter:** Each prediction's Shannon's entropy is calculated, and it is then compared with the training data set's entropy:

$$\Delta H_{\text{Shannon}} = \left| \hat{H}_{\text{Shannon}} - H_{\text{Shannon}} \right| \quad , \tag{9.5}$$

   where H_{Shannon} is Shannon's entropy of the training data set, and H_{Shannon} is Shannon's entropy of a prediction. In the end, we discard all predictions with a comparatively large $\Delta H_{\text{Shannon}}$ and keep only the ones with low $\Delta H_{\text{Shannon}}$.

Additionally, we used all presented filters in combination with each other, e.g., first, a filter based on the Hurst exponent, and second, a filter based on Fisher's information. Again, this was performed such that the remaining predictions make up to 1% of all 500 ensemble predictions. Therefore, the first filter reduces the whole ensemble to only 10%, i.e., 50 predictions, and the second filter reduces the remaining predictions to 10%, i.e., five predictions, thus 1%.

Figure 9.5 explains the idea and shows a result for applied complexity filters. The left plot is the whole ensemble without any filtering, and the right side is the filtered ensemble prediction. Here a filter combining SVD entropy and Lyapunov exponents is applied to the ensemble predictions.

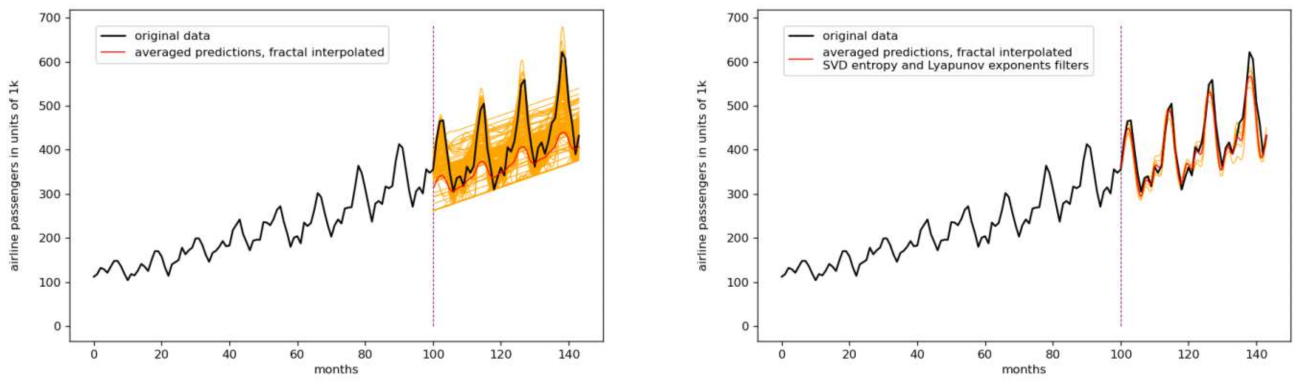


Figure 9.5: Plots for the unfiltered ensemble predictions (left plot) and the filtered ensemble predictions (right plot). Here, first, an SVD-entropy-based filter and second, a filter based on Lyapunov exponents were employed to improve the predictions. Also, the training data was a fractal interpolated data set with six additional interpolation points. The orange lines are different predictions constituting the ensemble. The red lines are the averaged predictions.

9.4 Results/Discussion

First, all five discussed data sets are linear and fractal interpolated. Next, five hundred predictions using randomly parameterized LSTM neural networks are made for each data set and subcategory, i.e., non-interpolated, linear interpolated, and fractal interpolated. The results are presented in Appendix B.6, in Tables B.9 and B.10.

Next, we employed filters based on the complexity of the data under study to reduce the number of ensemble predictions from 500 to 5, i.e., to 1%.

The best five results for all data are listed in Table 9.2 for the monthly international airline passengers data set. The results for the remaining data sets are collected in Appendix B.3, in Tables B.5–B.8. The overall best three results are highlighted as bold entries.

The results show that the interpolated approaches consistently outperformed the non-interpolated ones regarding the lowest RMSEs. Further, the presented filter methods can significantly improve the prediction, i.e., lower the RMSE.

9.4.1 Interpolation Techniques

Of all three best results for data sets, i.e., 15 best results overall, all best results are interpolated in one way or another. Further, nine are fractal interpolated, and six are linear interpolated predictions. In some cases, the linear interpolated predictions performed better than the fractal interpolated ones. Still, the conclusion is that fractal interpolation provides a slight improvement. The reasons for this conclusion are:

The results in Figure 9.6 and Table 9.2 show that the RMSE of the linear interpolated prediction is lower (best result, lowest RMSE) than that of the following two best results, which are fractal interpolated ones. But the corresponding error of the RMSE, i.e., the uncertainty of the prediction, is much higher for the linear interpolated case. A closer look at Figure 9.6 supports this claim, i.e., the quality of the linear interpolated single predictions is low in terms of how close the ensemble predictions resemble the actual curve. Thus the author guesses that the advantages of the linear interpolated results will vanish for increased statistics. This behavior is present for the monthly international airline passenger data set, the monthly car sales in Quebec data set, and the CFE specialty monthly writing paper sales data set.

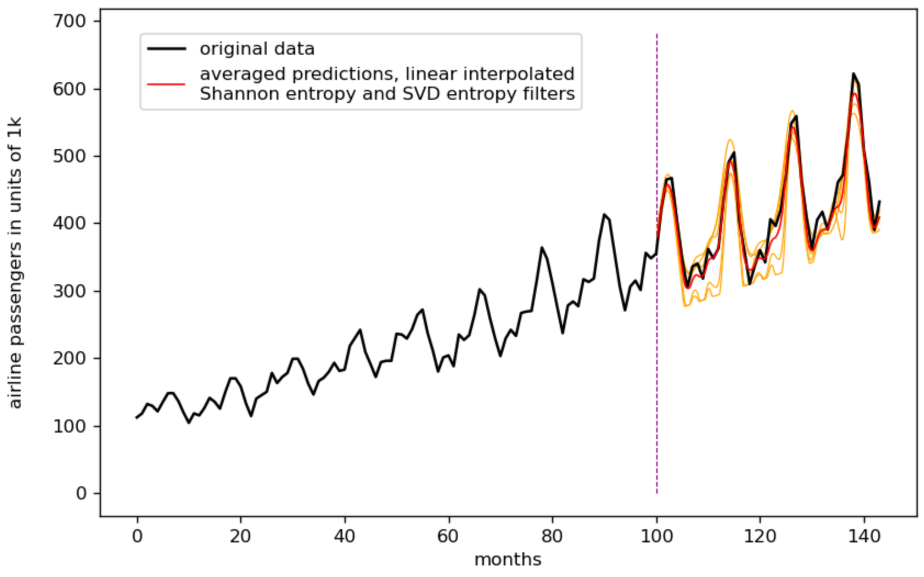


FIGURE 9.6: Best result for the monthly airline passengers data set. The orange lines are the remaining ensemble predictions after filtering. The red line is the averaged ensemble prediction. Linear interpolated, three interpolation points, Shannon entropy and SVD entropy filter, error: 0.03542 ± 0.00625

TABLE 9.2: Error table for the monthly airline passengers data set. The bold results are the three best ones for this data set.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	fisher svd	0.04122 ± 0.00349
non-interpolated	-	svd	0.04122 ± 0.00349
non-interpolated	-	svd shannon	0.04122 ± 0.00349
non-interpolated	-	fisher	0.04166 ± 0.00271
non-interpolated	-	fisher shannon	0.04166 ± 0.00271
fractal interpolated	**1**	**fisher hurst**	**0.03597 ± 0.00429**
fractal interpolated	**1**	**svd hurst**	**0.03597 ± 0.00429**
fractal interpolated	5	hurst fisher	0.03980 ± 0.00465
fractal interpolated	5	hurst svd	0.03980 ± 0.00465
fractal interpolated	5	shannon	0.04050 ± 0.00633
linear interpolated	**3**	**shannon svd**	**0.03542 ± 0.00625**
linear interpolated	3	shannon fisher	0.03804 ± 0.00672
linear interpolated	5	fisher	0.04002 ± 0.00357
linear interpolated	5	fisher shannon	0.04002 ± 0.00357
linear interpolated	5	svd fisher	0.04002 ± 0.00357

9.4.2 Complexity Filters

Given the 75 best results regarding all interpolation techniques and different data sets, a significant 62 are double-filtered predictions, meaning that we employed two consecutive complexity filters. 13 are single-filtered, and not a single unfiltered prediction made it into the top 75 predictions. Thus, we suggest using two consecutive complexity filters to improve the presented ensemble approach.

When pinning down which filters perform best, a precise answer cannot be given for this study. The reason is that within the best 15 results, only the combined filters consisting of SVD entropy & Hurst exponent and Lyapunov exponents & Hurst exponent occur more than once, i.e., each of them only two times.

Within the best 75 results, the situation looks different. The combined filters Shannon's entropy & Fisher's information occur seven times, followed by six occurrences of Shannon's entropy & SVD entropy. Here it's interesting to note that Fisher's information and SVD entropy are complexity measures based on single value decomposition (SVD). Thus a total of 57 of the best 75 results contain at least one SVD-based complexity measure. Therefore, employing an SVD-based complexity measure in combination with the Hurst exponent or Shannon's entropy is suggested. The author recommends using a combination of SVD-entropy and the Hurst exponent.

9.5 Remarks and Summary

The research presented in this chapter is a combined approach of two interpolation techniques, randomly parameterized LSTM neural network ensembles and complexity filters.

Finally, we give a summary and some remarks on the presented experiments:

- The presented randomly parameterized ensemble predictions can significantly be improved by employing fractal and linear interpolation techniques. The author recommends a fractal interpolation approach as the fractal interpolated predictions appear more stable with respect to the deviations of the errors.

- The presented randomly parameterized ensemble predictions can significantly be improved using filters based on complexity measures. The basic idea here is to only keep predictions with a signal complexity close to that of the training data.

  The unfiltered and non-interpolated results from Tables B.9 and B.10 when compared to the best results, shown in Tables 9.2 and B.5–B.8, show that the RMSE was reduced by a factor of ≈ 10 on average.

- Given a baseline of single step-by-step predictions, Tables B.2–B.4, and Appendix B.5, the filtered best ensemble predictions always outperformed the baseline predictions. One can probably find slightly better baseline predictions by further optimizing the baseline neural networks. However, the presented baseline results are still very reasonable and, in comparison, show the quality of the ensemble predictions.

- On first grasp, the unfiltered results, Tables B.9 and B.10, suggest a trend for the errors depending on the number of interpolation points. But when applying complexity filters, this trend vanishes. Thus no recommendation can be given on the optimal number of interpolation points.

- The Hurst exponent is increased for all interpolated data sets compared to the non-interpolated case. The same is true for the largest Lyapunov exponent. But whereas an increased Hurst exponent indicates increased predictability, an increased largest Lyapunov exponent suggests a more chaotic and unpredictable behavior. This is somewhat contradictory, and given that the interpolated approaches outperformed the non-interpolated ones, we conclude that the Hurst exponent is a better tool to assess predictability for non-model data sets. Still, both the spectrum of Lyapunov exponents and the Hurst exponent are capable of filtering and thus improving predictions.

- SVD entropy and Fisher's information qualitatively depict the same characteristics of a time series. We see this for all discussed data sets, which is discussed and shown in Section 9.1. This is apparent as both measures of signal complexity perform a single value decomposition for time series data. Single value decomposition requires two parameters, the time delay, and an embedding dimension, hence a phase space embedding. Both SVD entropy and Fisher's information show the same behavior for interpolated time series data differing in their numbers of interpolation points: Increasing the number of interpolation points increases Fisher's information and decreases SVD entropy. Thus both measures of signal complexity indicate increased regularity and predictability for interpolated time series data.

The presented approach used only univariate time series data. One can achieve an extension to arbitrary dimensional multivariate data by extending the neural networks to multivariate data/predictions. The author expects the multivariate prediction approach to benefit significantly from the presented ideas.

Different features may behave differently regarding their complexities when considering multivariate time series data. A tool to cope with varying complexities for different time series data is, e.g., transfer entropy, [106]. Transfer entropy and effective transfer entropy are complexity measures that deal with multivariate problems.

The presented approach is limited by the parameter range of the neural network implementation. Though one can use arbitrary ranges for the neural network parameters, one can significantly reduce computation costs if a good range is known.

Chapter 10

Interpolating Strange Attractors via Fractional Brownian Bridges

This chapter provides an experimental setup and results for the attractor-based stochastic interpolation approach, named PhaSpaSto interpolation. This method is described in Section 5.2. Further, this chapter contains and follows closely the work published in [30]:

Sebastian Raubitzek, Thomas Neubauer, Jan Friedrich, and Andreas Rauber. Interpolating strange attractors via fractional brownian bridges. Entropy, 24(5), 2022. ISSN 1099-4300. doi: 10.3390/e24050718. URL https://www.mdpi.com/1099-4300/24/5/718. Visited on 2023-04-20

This experiment aims to show the applicability of the developed PhaSpaSto interpolation. I.e., we want to show that the developed method can reconstruct missing data points better than comparable methods and that the proposed method gives a smoothly interpolated phase space portrait. First, this is done by testing the method on model data, i.e., the famous Lorenz system; second, by validating the presented approach on non-model data sets; and third, presenting and discussing interpolated non-model data sets.

To introduce this chapter we sum up PhaSpaSto interpolation, which is described in Section 5.2: Multipoint fractional Brownian Bridges are employed to generate a population of different interpolations with varying Hurst exponents of a single time series data. Next, a genetic algorithm is used to improve this population in terms of the smoothness of the reconstructed phase space trajectory. We achieve this by reducing the variance of second derivatives along the reconstructed phase space trajectory.

Interpolating Strange Attractors via Fractional Brownian Bridges

The idea behind this is the following: We expect real-life data sometimes to behave like dynamical systems but only partially like data from a system of ordinary differential equations. Real-life data usually has some noise or some stochastic aspects to it. To cope with both of these aspects, we developed PhaSpaSto interpolation. First, we stochastically interpolated a given time series, then searched for the smoothest trajectory in reconstructed phase space among these *random* interpolations. Thus we hope to combine the stochastic aspects of real-life data and a corresponding smoothed-out phase space trajectory.

We chose the Lorenz system as it is a well-known chaotic system that is known to have an attractor structure in the reconstructed phase space. The Lorenz system is discussed in Section 7.15. The corresponding results are shown in Section 10.1.1. We chose the other data sets to have a selection of data sets with visible oscillatory regularities, which yield an interpretable structure in reconstructed phase space. This means we see a somehow concentric structure in the reconstructed phase space, which can be enhanced using the developed technique. These data sets are the NYC measles outbreaks, the car sales in Quebec data set, the Perrin Freres champagne data set, the monthly international airline passengers data set, and the monthly mean temperature in Nottingham castle data set. Further, we chose two data sets that behave more randomly, similar to a fractional Brownian motion. These data sets are the shampoo sales data set and the annual maize yields in Austria data set. All used data sets are discussed in Chapter 7 and the corresponding results are discussed in Section 10.1.2.

Finally the findings are summarized in Section 10.2

10.1 Experimental Results

Here we provide the results of the genetic algorithm for all data sets, first for the Lorenz system and then for seven non-model data sets. For both cases, we validate the developed method such that we delete data points from the original time series and reconstruct the missing data points using PhaSpaSto interpolation. Further, we test PhaSpaSto interpolation against the best random interpolation of the population, against a linear interpolation, and a cubic spline interpolation [101]. Both the linear and spline interpolation were performed using the python package `scipy` [107]. The SciPy-spline-interpolation is a piecewise cubic polynomial which is twice continuously differentiable [108].

For the validation, we emphasize the Lorenz system, as the generated model data allows us to test arbitrary settings, i.e., using varying numbers of missing data points per the number of chosen interpolation points. Contrary to that, for the non-model data sets, we delete every second data point and reconstruct the missing data using PhaSpaSto interpolation. For the non-model data sets, we additionally present actually interpolated results, i.e., data sets with smoothed-out phase space trajectories.

10.1.1 Results for the Lorenz System

For more details on the Lorenz system, see Section 7.15. We performed our interpolation for a range of interpolation points $N_I = \{1, 2, 3, 4, \ldots, 20\}$.

Further, we then developed an experimental setup to test the method's applicability.

1. Generate a univariate time series from the Lorenz system.
2. Delete points from the data set to later be interpolated.
   Given a time series data of the Lorenz system, $[x_1, x_2, \ldots, x_n]$, we extracted data points according to the number of interpolation points $n_I \in N_I$ and then use the proposed interpolation technique such that:

   $$\begin{aligned} &[x_1, x_{n_I+2}, x_{2n_I+2}, \ldots, x_n] \quad \text{Original data points to be kept for interpolation,} \\ &[x_1, \hat{x}_1, \ldots, \hat{x}_{n_I}, x_{n_I+2}, \hat{x}_{n_I+1}, \ldots, \hat{x}_{2n_I+1}, x_{2n_I+2}, \hat{x}_{2n_I+2}, \ldots x_n] \quad \text{Interpolated data,} \end{aligned} \tag{10.1}$$

   where $\hat{x}_i$ are the interpolated data points.
3. Perform Interpolation according to Section 5.2.
4. Calculate the RMSE for the interpolated data with respect to the original data $[x_2, \ldots, x_{n_I+1}, x_{n_I+3}, \ldots, x_{2n_I+1}, \ldots]$. Do the same for the average interpolation of the whole population and each time series of the initial population.

Thus we get errors for the population mean, for all interpolated time series *in* the initial population and the improved interpolation using the discussed genetic algorithm. The corresponding RMSE is given as:

$$E_{RMSE} = \left(\frac{1}{n} \sum_{i=1}^{n} [\hat{x}_i - x_i]^2 \right)^{\frac{1}{2}} \quad . \tag{10.2}$$

Here x_i are the original data points, $\hat{x}_i$ are the newly interpolated values and n is the length of the time series.

The presented results for the Lorenz system show that the algorithm can identify/generate the best interpolation in terms of a low RMSE on missing data points out of the given initial population. This can be seen in Table 10.1, where we highlighted the

Interpolating Strange Attractors via Fractional Brownian Bridges

results where the genetic-algorithm-improved-interpolation outperformed every random interpolation of the population. Still, the spline interpolation outperforms the presented approach. This is also depicted in Figure 10.1, where we plotted the RMSE on missing data points for varying interpolation points. This graphic shows that the presented approach requires a certain amount of interpolation points, in this case, three, to be close to the best random interpolation of the population. We assume that the reason for this is that the variance of second derivatives along a phase space trajectory requires a certain *density* of phase space points to differ between smooth and edgy phase space trajectories. On the other hand, the spline interpolation performs well right from the start.

The corresponding reconstructed phase space plots (Figure 10.3) show that both, the best random interpolation (e) and the genetic-algorithm-improved interpolation (f), provide convincing phase space portraits. I.e., both are indeed close to the ground truth (a). The population mean (b), on the other hand, is far off and has a lot of pointed edges. The phase space picture produced by linear interpolation (c) is also quite angular, as one would expect from linear interpolation. In contrast, the phase space portrait for the spline interpolation (d) is the one that is the one closest to the original phase space portrait.

We plotted all obtained results for 13 interpolation points as time series in Figure 10.2. The results show that the population mean (a) is far off the ground truth and differs drastically at the high and low peaks, as it does not reach the actual data points. The genetic-algorithm-improved (b) and the best random interpolation of the initial population (c) capture most of the high and low peaks compared to the population mean. Further, when comparing the genetic-algorithm-improved and the population mean (d), one can see that the improved interpolation provides a smoother curve when depicted as a time series. In contrast, the population mean tends to produce sharp peaks. Finally, we compare the linear interpolation (e) and the spline interpolation (f) to the genetic algorithm improved interpolation. The linear interpolation here is far off, but the spline interpolation reproduces the Lorenz system almost perfectly, thus outperforming the genetic-algorithm-improved interpolation.

TABLE 10.1: Errors for the interpolated data on the Lorenz system depending on the number of interpolation points. The errors are shown for the mean interpolation of all populations and improved interpolation using the presented genetic algorithm. *Lowest RMSE in population* refers to the best randomly interpolated result, i.e., the one interpolation from the population that produced the lowest error by chance. We also featured the results for the linear and spline interpolation. We highlighted the interpolations where the genetic-algorithm-based interpolation outperformed the whole population of interpolations. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation. This table is depicted in Figure 10.1.

n_I	1	2	3	4	5	6	7	**8**	9	10
RMSE population mean	0.77419	0.88263	0.91026	0.89442	0.87013	0.86858	0.89120	**0.84220**	0.90323	0.88777
lowest RMSE in population	0.17939	0.18068	0.16757	0.19206	0.16126	0.18134	0.17782	**0.17211**	0.18216	0.19371
RMSE linear interpolated	0.42534	0.44752	0.44179	0.44185	0.41574	0.42406	0.42894	**0.40883**	0.43263	0.43353
RMSE spline interpolated	0.12263	0.11808	0.09968	0.12586	0.09678	0.12195	0.12280	**0.10862**	0.11554	0.13008
RMSE gen. alg. improved	1.03779	0.24381	0.17517	0.19488	0.16264	0.18182	0.17818	**0.17121**	0.18239	0.19374
below best %	74.4%	21.6%	4.3%	2.2%	1.4%	1.1%	0.7%	**0.1%**	0.8%	0.3%

n_I	11	**12**	**13**	**14**	**15**	16	17	18	19	20
RMSE population mean	0.90844	**0.92145**	**0.91509**	**0.90686**	**0.91750**	0.90326	0.90789	0.90080	0.88835	0.89651
lowest RMSE in population	0.18789	**0.19238**	**0.18693**	**0.18632**	**0.19943**	0.19640	0.19208	0.18449	0.18415	0.20291
RMSE linear interpolated	0.43649	**0.44211**	**0.43687**	**0.43423**	**0.44142**	0.43534	0.43720	0.43170	0.42685	0.43398
RMSE spline interpolated	0.12086	**0.12646**	**0.11530**	**0.11765**	**0.12532**	0.12873	0.13098	0.12581	0.11912	0.12581
RMSE gen. alg. improved	0.18816	**0.19141**	**0.18670**	**0.18626**	**0.19943**	0.19663	0.19215	0.18462	0.18441	0.20300
below best %	0.8%	**0.1%**	**0.1%**	**0.1%**	**0.1%**	0.8%	0.6%	0.6%	0.8%	0.6%

Chapter 10 Interpolating Strange Attractors via Fractional Brownian Bridges

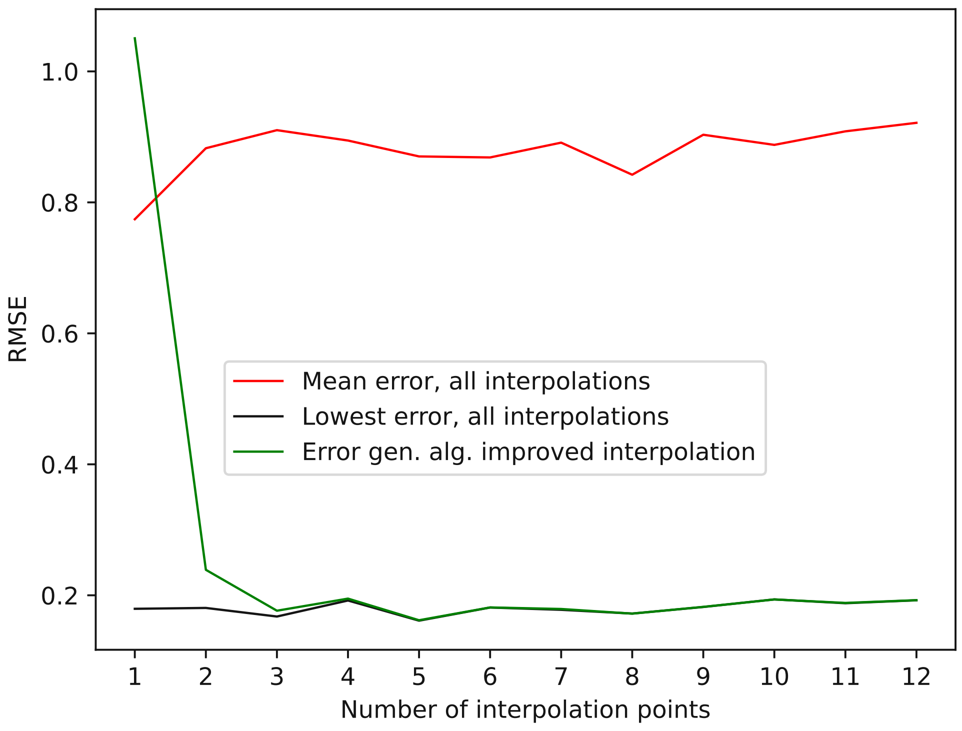
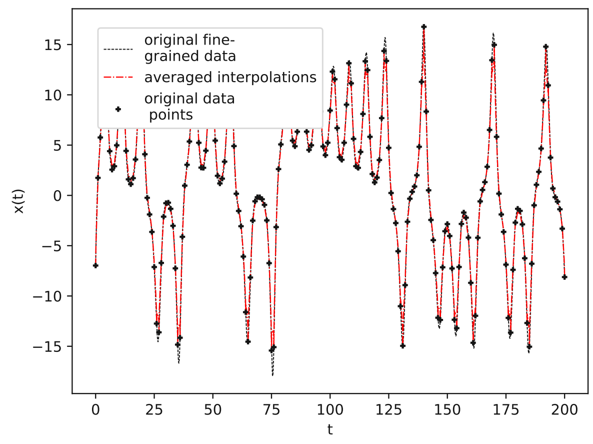
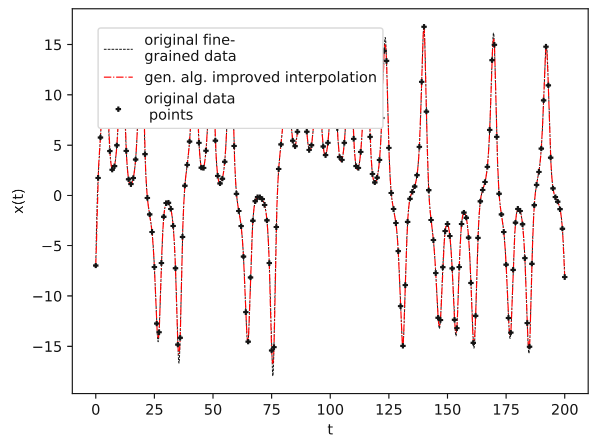


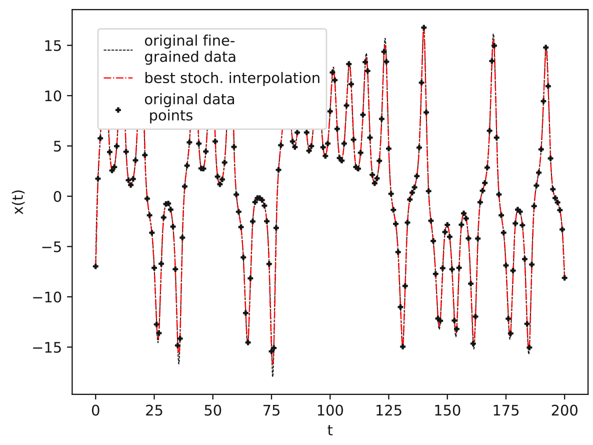
FIGURE 10.1: Shown in this Figure are the errors from Table 10.1 depending on the different numbers of interpolation points.



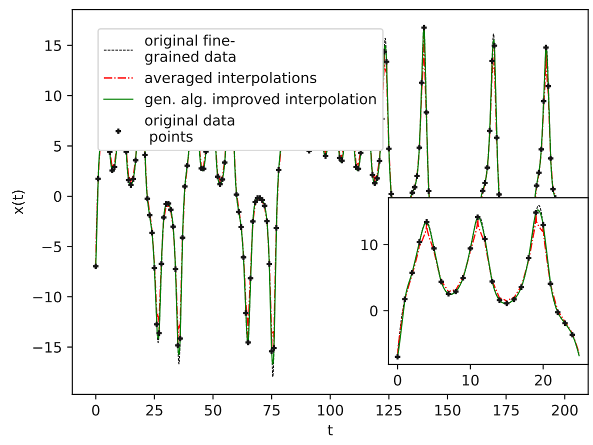
(a)



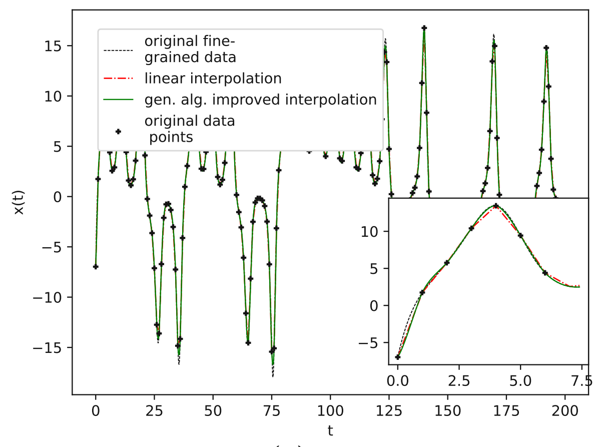
(b)



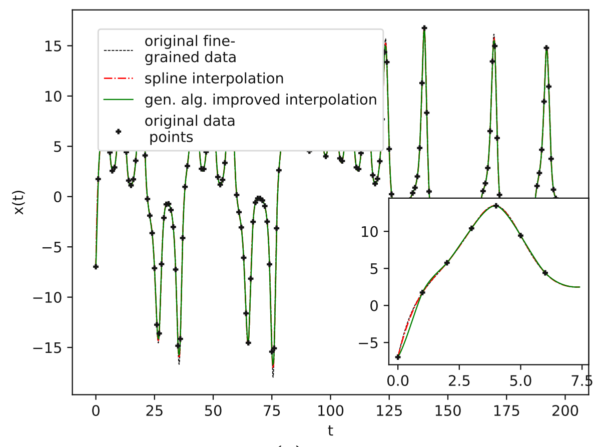
(c)



(d)

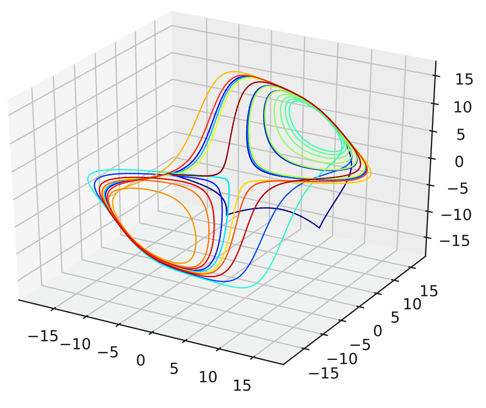


(e)

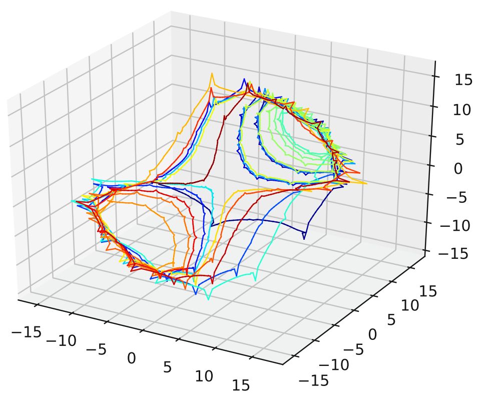


(f)

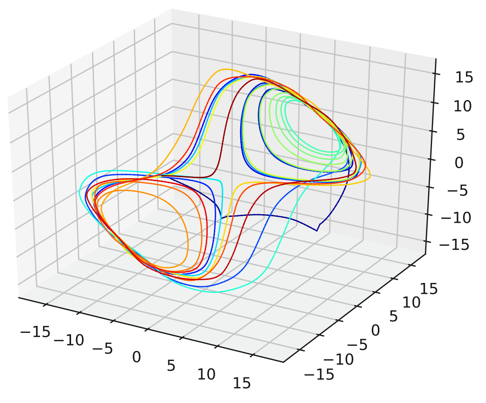
FIGURE 10.2: Original vs. interpolated time series data.
(a): Non-interpolated original data (i.e. the one the error's are calculated with) and population average;
(b): Genetic-algorithm-improved interpolation;
(c): The one interpolation of the population that has the lowest RMSE;
(d): Population average vs. genetic-algorithm-improved interpolation;
(e): Linear interpoaltion vs. genetic-algorithm-improved interpolation;
(f): Spline interpolation vs. genetic-algorithm-improved interpolation.



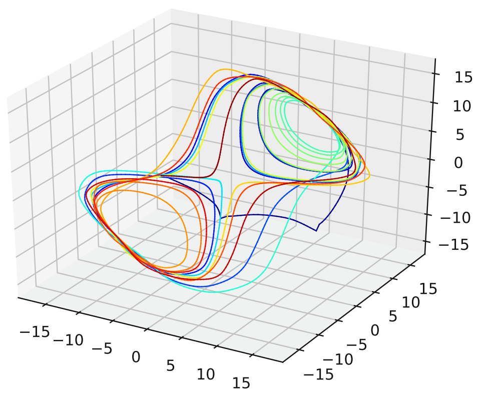
(a)



(b)



(c)



(d)

FIGURE 10.3: Reconstructed attractors for the interpolated Lorenz system.
(a): Non interpolated original data (i.e. the one the errors are calculated with);
(b): The average interpolation of the whole population;
(c): The one interpolation of the population that has the lowest RMSE;
(d): Interpolation imrpoved by the presented genetic algorithm approach.

10.1.2 Results for Non-Model Data Sets

This section provides tests for PhaSpaSto interpolation on real-life data sets with only a limited number of sampled data points. However, these data sets are the focus of the proposed method, i.e., to increase the fine-grainedness of short, sparsely-sampled time series data, e.g., agricultural data sets. Our approach is not restricted to equidistant time series: Due to the general form of the bridge construction (Equation 5.5), non-equidistant time series excerpts can be interpolated as well.

For this reason, we chose five data sets to demonstrate our method further, i.e., we validate the interpolation with missing data points and then present an actual interpolation and the improved phase space trajectories for the featured time series data. We consider a phase space trajectory to be improved if we achieve smoother trajectories, which exhibit fewer edgy points in reconstructed phase space. Further, for the reconstructed phase space plots, we rescaled every data set to the unit interval and subtracted a linear fit from the data set if a linear trend was visible, which is noted in Chapter 7 for each data set.

The validation on these non-model data sets is performed such that we delete every second data point of the original time series. Then, we interpolate all the gaps to reconstruct the missing data points. We present results for the average prediction of the population, the random interpolation with the lowest RMSE, a linear interpolation, a cubic spline interpolation, and the interpolation that was improved using the discussed genetic algorithm. This section features only the errors for the validation. All validation plots are collected in Appendix C.1 to keep the main text focused.

10.1.2.1 NYC Measles Outbreaks

This data set is discussed in Section 7.5 and we obtained it from [92], where it is discussed and shown to feature an attractor structure in reconstructed phase space. It depicts measles outbreaks in New York City (NYC) from 1928 to 1964, binned every two weeks, with a total of 432 data points. The data set depicts sharp, repetitive peaks, i.e., the increase and decrease of measles cases in NYC.

The results on how well the presented interpolation can reproduce missing data points of this data set are collected in Table 10.2 and depicted in Figure C.1 (a). These results show that, though the genetic-algorithm-improved interpolation drastically outperforms the average random interpolation, the algorithm did not once outperform the best interpolation of the population. Still, starting with seven interpolation points, the genetic-algorithm-improved interpolation performs well and is very close to the best of 1000 randomly interpolated results, i.e., always below or around the best 1% of the population. Further, PhaSpaSto interpolation does outperform the cubic spline interpolation starting with five interpolation points. We thus conclude that the presented interpolation technique captures the phase-space properties of this data set and effectively can be used to interpolate this time series. Also, compared to the cubic and linear interpolation, the proposed method takes at least seven interpolation points to reach peak performance for this data set. All validation plots, are collected in Appendix C.1.2.

An interpolation of the original data set is depicted in Figure 10.4. When comparing the reconstructed phase space of the original data set, the population mean (c), and the presented interpolation technique (d); we see that the phase space portrait of the latter features a smoothed-out phase space trajectory compared to the original time series (b) and the population mean (c), which are both pointy and have many sharp edges. Further, considering the actual time series graph (a), we see that the presented interpolation technique increases the major peaks, thus making extreme events more prominent.

Table 10.2: Errors for the interpolated data on the NYC measles data set depending on the number of interpolation points. We show the errors for the mean interpolation of all populations, the lowest error in the population, and the interpolation improved using the presented genetic algorithm. We highlighted the interpolation where the genetic-algorithm-based interpolation performed best. The corresponding plots for the best interpolations are collected in Appendix C.1.2. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation.

n_I	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
RMSE Population Mean	860.56140	860.56165	860.56098	860.56235	860.56124	860.56210	860.56145	860.56220
Lowest RMSE in population	594.27833	594.27832	594.27832	594.27832	594.27748	594.27831	594.27832	594.27833
RMSE linear interpolated	713.61079	713.61089	713.61089	713.61089	713.61089	713.61089	713.61089	713.61089
RMSE spline interpolated	607.03778	607.03778	607.03778	607.03778	607.03778	607.03778	607.03778	607.03778
RMSE gen. alg. improved	1138.28460	621.70136	602.03361	594.36367	594.33054	594.34891	594.34819	594.34132
Below Best %	75.3%	25.8%	13.4%	0.8%	0.8%	0.8%	0.8%	0.8%

n_I	**17**	**19**	**21**	**23**	**25**	**27**	**29**	**31**
RMSE Population Mean	860.56196	860.56132	860.56168	860.56090	**860.56153**	860.56287	860.56138	860.56192
Lowest RMSE in population	594.27901	594.27750	594.27934	594.27834	**594.28039**	594.27831	594.28069	594.27833
RMSE linear interpolated	713.61089	713.61089	713.61089	713.61089	**713.61089**	713.61089	713.61089	713.61089
RMSE spline interpolated	607.03778	607.03778	607.03778	607.03778	**607.03778**	607.03778	607.03778	607.03778
RMSE gen. alg. improved	594.33772	594.33837	594.33400	594.35508	**594.31806**	594.36050	594.42183	594.39145
Below Best %	0.8%	0.8%	0.8%	0.8%	**0.6%**	0.8%	1.1%	1.1%

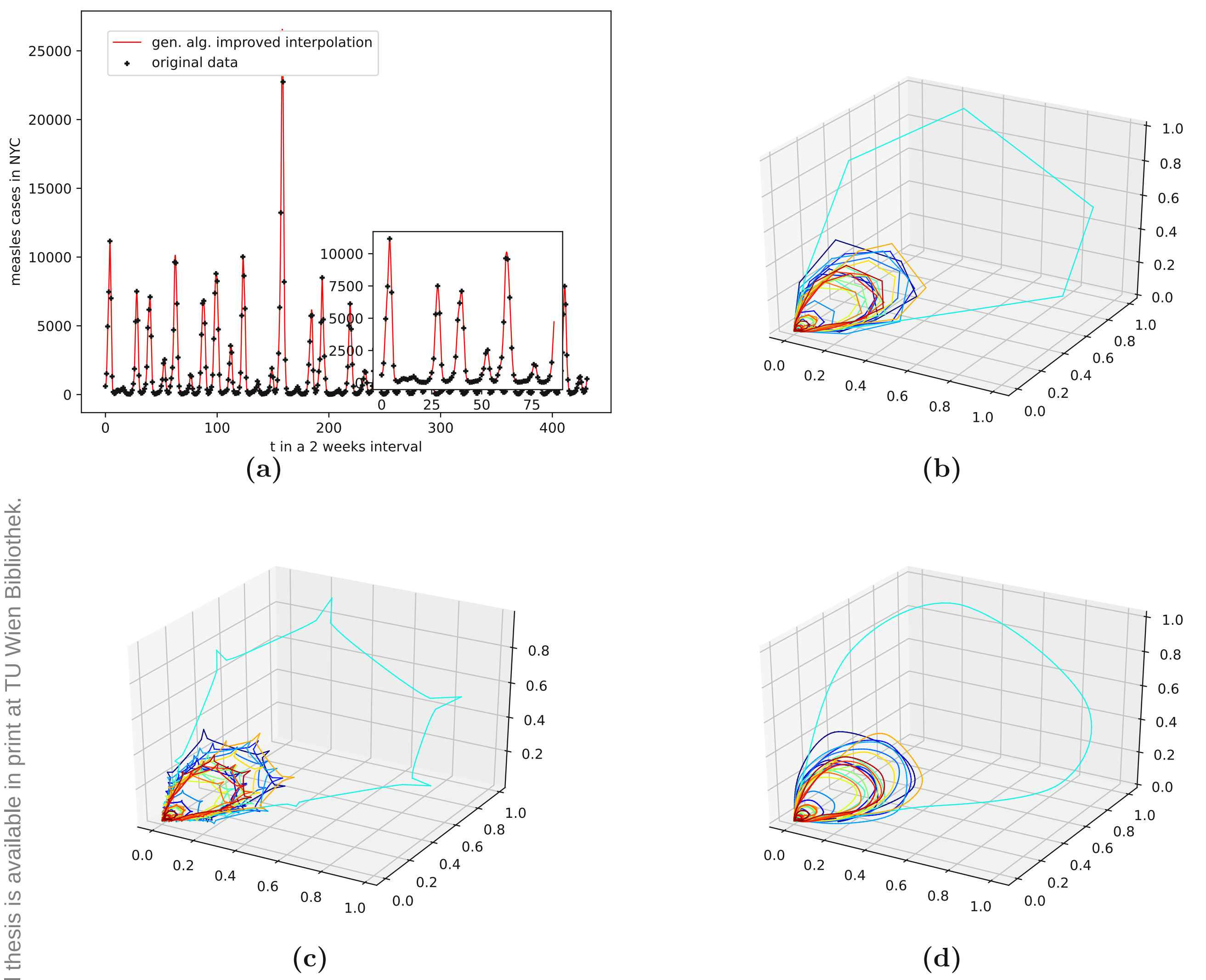


FIGURE 10.4: Interpolated data and reconstructed attractors for the NYC measles outbreaks data set.
(a): The original and interpolated time series data;
(b): Phase space reconstruction of the original data;
(c): Phase space reconstruction of the average population data;
(d): Phase space reconstruction of the genetic-algorithm-improved data.

10.1.2.2 Car Sales in Quebec

This data set is discussed in Section 7.4. This data set clearly shows an increasing linear trend and oscillatory regularities, i.e., seasonal behavior.

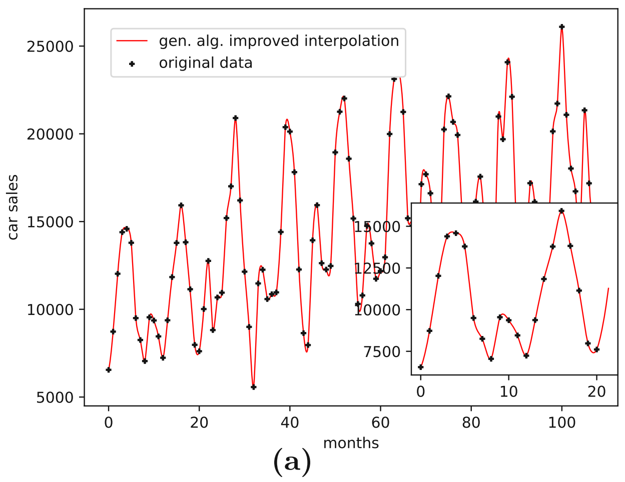
The results on the reproducibility of missing data points for all interpolation techniques are collected in Table 10.3, and depicted in Figure C.1 (b). The genetic-algorithm-improved interpolation drastically outperforms the average random interpolation. Further, PhaSpaSto interpolation always outperforms the cubic spline and linear interpolation. A random interpolation mostly achieves the overall best performance. Still, the PhaSpaSto interpolation performs best for one, three, and five interpolation points. Overall, the genetic-algorithm-improved interpolation performs well and is very close to the best of 1000 randomly interpolated results, i.e., for most cases below or around the best 1% of the population. Thus, we conclude that the presented interpolation technique effectively captures the phase-space properties of this data set and can be used to interpolate this time series data. We collect all additional plots for the validation in Appendix C.1.3, where one can find the reconstructed attractors for all interpolated validation sets and the corresponding time series plots.

An interpolation of the original data set is depicted in Figure 10.5. Here Figures 10.5 (c) and (d) present the population mean and the improved interpolation, respectively. When comparing them, one can see that the genetic algorithm improves the phase space portrait in terms of a smoothed-out phase space trajectory compared to the original time series (b) and the population mean (c), which are both pointy and have many sharp edges. When considering the actual time-series graph (a), the presented interpolation technique increases the major peaks, thus making extreme events more prominent. Further, it provides a rather smooth curve, i.e., no pointy edges, as depicted in the zoomed-in plot in (a).

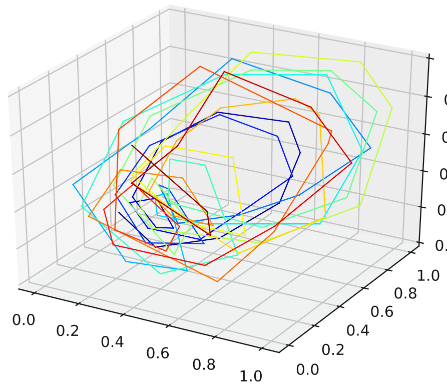
TABLE 10.3: Errors for the interpolated data on the car sales in Quebec data set depending on the number of interpolation points. The errors are shown for the mean interpolation of all populations, the linear interpolation, the cubic spline interpolation, as well as for the lowest error in the population, and for the interpolation that was improved using the presented genetic algorithm. We highlighted the interpolation where the genetic-algorithm-based interpolation performed best. We show the corresponding plots for the best interpolation in Appendix C.1.3. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation.

n_I	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
RMSE Population Mean	**2030.11005**	2030.11166	2030.11148	2030.11230	2030.11030	2030.11138	2030.11106	2030.11110
Lowest RMSE in population	**1954.95010**	1954.95013	1954.95016	1954.95013	1954.95005	1954.95009	1954.95020	1954.95015
RMSE linear interpolated	**2017.79949**	2017.79949	2017.79949	2017.79949	2017.79949	2017.79949	2017.79949	2017.79949
RMSE spline interpolated	**1971.23755**	1971.23755	1971.23755	1971.23755	1971.23755	1971.23755	1971.23755	1971.23755
RMSE gen. alg. improved	**1907.40084**	1960.21475	1954.94790	1954.94792	1954.95375	1954.97452	1958.57232	1954.97468
Below Best %	**0.1%**	17.2%	0.1%	0.1%	0.6%	1.01%	14.6%	1.01%

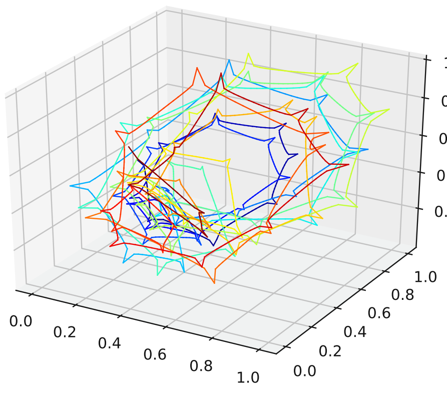
n_I	**17**	**19**	**21**	**23**	**25**	**27**	**29**	**31**
RMSE Population Mean	2030.11260	2030.11057	2030.11226	2030.11047	2030.11078	2030.11105	2030.11171	2030.11013
Lowest RMSE in population	1954.95010	1954.95007	1954.95011	1954.95014	1954.95007	1954.95010	1954.95003	1954.95021
RMSE linear interpolated	2017.79949	2017.79949	2017.79949	2017.79949	2017.79949	2017.79949	2017.79949	2017.79949
RMSE spline interpolated	1971.23755	1971.23755	1971.23755	1971.23755	1971.23755	1971.23755	1971.23755	1971.23755
RMSE gen. alg. improved	1954.97730	1954.99153	1955.00052	1954.99273	1955.02450	1955.02418	1955.01367	1954.98108
Below Best %	1.3%	1.4%	1.4%	1.4%	1.6%	1.6%	1.4%	1.4%



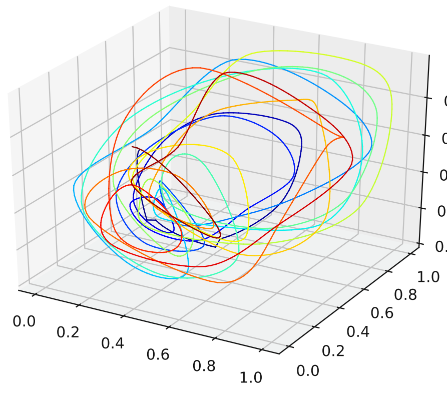
(a)



(b)



(c)



(d)

FIGURE 10.5: Interpolated data and reconstructed attractors for the car sales in Quebec data set.
(a): The original and interpolated time series data;
(b): Phase space reconstruction of the original data;
(c): Phase space reconstruction of the average population data;
(d): Phase space reconstruction of the genetic-algorithm-improved data.

Interpolating Strange Attractors via Fractional Brownian Bridges

10.1.2.3 Perrin Freres Champagne Sales

This data set is discussed in Section 7.3.

The validation results for this data set are collected in Table 10.4 and Figure C.1c.

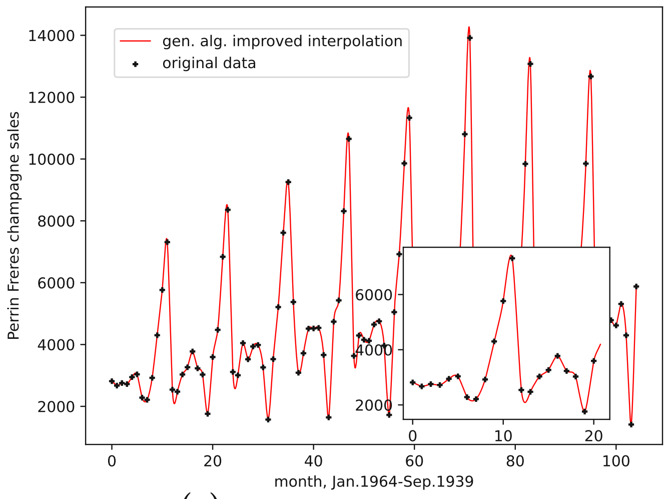
Though the genetic-algorithm-improved interpolation drastically outperforms the average random interpolation, the algorithm did not once outperform the best interpolation of the population. Still, starting with five interpolation points, the genetic-algorithm-improved interpolation performs well and is very close to the best of 1000 randomly interpolated results, i.e., consistently below or around the best 1% of the population. Overall the cubic spline interpolation performed best on this data set. Though the linear interpolation outperforms the population mean, it is still far off. We thus conclude that the presented interpolation technique does capture the phase-space properties of this data set from a given population and can be used to interpolate this time series data, but the cubic spline interpolation is the better choice.

An interpolation of the original data set is depicted in Figure 10.6. We again show the population mean (c) and the improved interpolation (d). The presented interpolation technique improves the phase space portrait in terms of a smoothed-out phase space trajectory (d) compared to the original time series (b) and the population mean (c), which are both pointy and contain many sharp edges. Here the population mean increased sharp edges drastically. Further, considering the graph of the actual time series (a), the presented interpolation technique again increases the major peaks. We consider the interpolated data set to be a rather smooth curve, as depicted in the zoom-in window in (a).

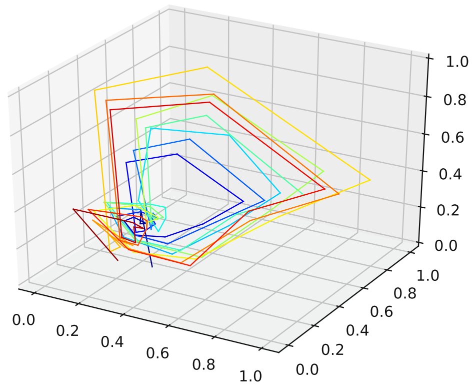
TABLE 10.4: Errors for the interpolated data on the Perrin Freres champagne sales data set depending on the number of interpolation points. The errors are shown for the mean interpolation of all populations, the lowest error in the population, linear interpolation, cubic spline interpolation, and improved interpolation using the presented genetic algorithm. We highlighted the interpolation where the genetic-algorithm-based interpolation performed best. The corresponding plots for the best interpolation are shown in Appendix C.1.4. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation.

n_I	1	3	5	7	9	11	13	15
RMSE Population Mean	2320.03501	2320.03532	2320.03403	**2320.03366**	2320.03524	2320.03333	2320.03195	2320.03301
Lowest RMSE in population	2144.04985	2144.05002	2144.04987	**2144.05007**	2144.04991	2144.04986	2144.04981	2144.04995
RMSE linear interpolated	2264.24606	2264.24606	2264.24606	**2264.24606**	2264.24606	2264.24606	2264.24606	2264.24606
RMSE spline interpolated	2099.58713	2099.58713	2099.58713	**2099.58713**	2099.58713	2099.58713	2099.58713	2099.58713
RMSE gen. alg. improved	2540.17542	2153.68153	2144.43642	**2144.07624**	2144.16502	2144.15580	2144.14292	2144.16357
Below Best %	82.5%	6.9%	1.01%	**0.3%**	0.6%	0.6%	0.6%	0.6%

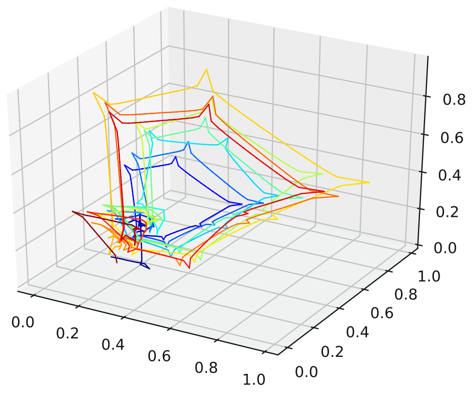
n_I	17	19	21	23	25	27	29	31
RMSE Population Mean	2320.03312	2320.03073	2320.03244	2320.03250	2320.03279	2320.03565	2320.03406	2320.03476
Lowest RMSE in population	2144.04987	2144.04986	2144.04967	2144.04982	2144.04997	2144.04985	2144.04976	2144.04999
RMSE linear interpolated	2264.24606	2264.24606	2264.24606	2264.24606	2264.24606	2264.24606	2264.24606	2264.24606
RMSE spline interpolated	2099.58713	2099.58713	2099.58713	2099.58713	2099.58713	2099.58713	2099.58713	2099.58713
RMSE gen. alg. improved	2144.09078	2144.13364	2144.17573	2144.13637	2144.16380	2144.10973	2144.10709	2144.13681
Below Best %	0.5%	0.6%	0.6%	0.6%	0.6%	0.6%	0.6%	0.6%



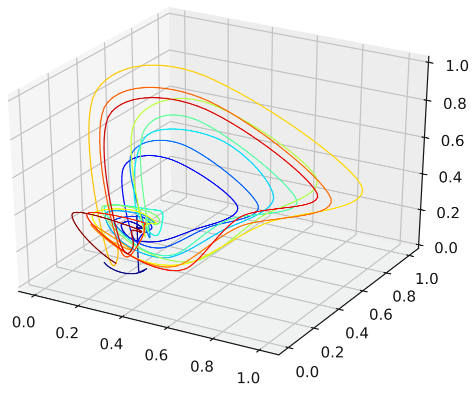
(a)



(b)



(c)



(d)

FIGURE 10.6: Interpolated data and reconstructed attractors for the Perrin Freres Champagne sales data set.
(a): The original and interpolated time series data;
(b): Phase space reconstruction of the original data;
(c): Phase space reconstruction of the average population data;
(d): Phase space reconstruction of the genetic-algorithm-improved data.

10.1.2.4 Monthly Airline Passengers

This data set is discussed in Section 7.1. Again this data set shows a visible linear trend and strong seasonal oscillatory regularities.

The results on how well the presented interpolation can reproduce missing data points of this data set are collected in Table 10.5 and depicted in Figure C.1 (d). The results show that, though the genetic-algorithm-improved interpolation drastically outperforms the average random interpolation, the algorithm did not once outperform the best interpolation of the population. Still, starting with three interpolation points, the algorithm outperformed the linear and the cubic spline interpolation. What's curious is that, for this data set, of all the non-model data sets, the linear interpolation outperforms the cubic spline interpolation.

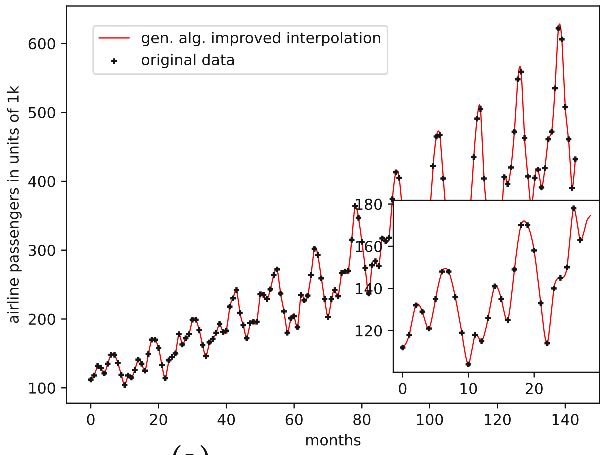
The genetic-algorithm-improved interpolation does not perform that well for this data set compared to a random interpolation of the time series. The improved interpolation is only around the best $\approx 40\%$ of the initial population for this data set. We thus conclude that the presented interpolation technique does not capture the phase-space properties of this data set very well. Still, the genetic algorithm does improve the initial population such that the population mean, the linear interpolation, and the cubic spline interpolation are outperformed, starting with three interpolation points. All time series and reconstructed attractor plots for this data set can be found in Appendix C.1.5.

An actual interpolation of the original data set is depicted in Figure 10.7. We again show the population mean (c) and the improved interpolation (d). The presented PhaSpaSto interpolation (d) improves the phase space portrait in terms of a smoothed-out phase space trajectory, compared to the original time series (b) and the population mean (c), which are both pointy and have many sharp edges. Further, considering the actual time series (a) graph, the PhaSpaSto interpolation technique slightly increases the major peaks. Also, compared to the other non-model data sets, the improved interpolation does provide a relatively smooth curve, but it looks way sharper than for, e.g., the car sales in Quebec data set (See Figure 10.5 (a)).

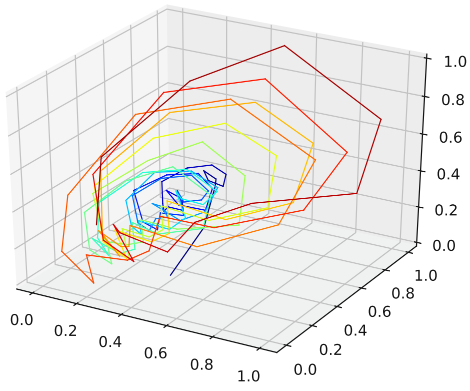
TABLE 10.5: Errors for the interpolated data on the monthly airline passengers data set depending on the number of interpolation points. The errors are shown for the mean interpolation of all populations, the linear interpolation, the cubic spline interpolation, as well as for the lowest error in the population, and for the interpolation that was improved using the presented genetic algorithm. We highlighted the interpolation where the genetic-algorithm-based interpolation performed best. The corresponding plots for the best interpolation are shown in Appendix C.1.5. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation.

n_I	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
RMSE Population Mean	19.93996	**19.93999**	19.93841	19.94072	19.93976	19.93873	19.94070	19.93889
Lowest RMSE in population	16.55624	**16.55779**	16.55732	16.55753	16.55558	16.55836	16.55719	16.55776
RMSE linear interpolated	17.39496	**17.39496**	17.39496	17.39496	17.39496	17.39496	17.39496	17.39496
RMSE spline interpolated	18.33872	**18.33872**	18.33872	18.33872	18.33872	18.33872	18.33872	18.33872
RMSE gen. alg. improved	18.65257	**16.81653**	16.84539	17.02728	16.84536	16.84545	16.84540	16.84539
Below Best %	59.4%	**35.6%**	38.0%	42.20%	38.1%	38.0%	38.1%	38.0%

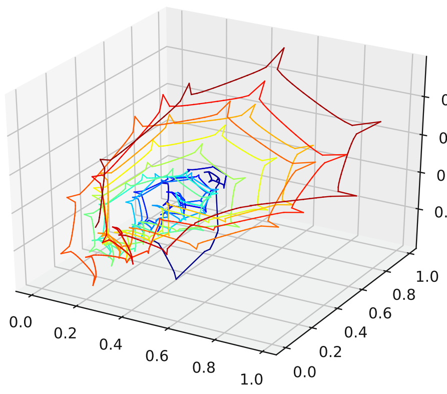
n_I	**17**	**19**	**21**	**23**	**25**	**27**	**29**	**31**
RMSE Population Mean	19.94029	19.94030	19.93985	19.93939	19.93659	19.94172	19.94023	19.93909
Lowest RMSE in population	16.55752	16.55730	16.55810	16.55715	16.55733	16.55603	16.55789	16.55741
RMSE linear interpolated	17.39496	17.39496	17.39496	17.39496	17.39496	17.39496	17.39496	17.39496
RMSE spline interpolated	18.33872	18.33872	18.33872	18.33872	18.33872	18.33872	18.33872	18.33872
RMSE gen. alg. improved	16.84546	16.84545	16.84548	16.84540	16.84535	16.84544	16.84544	16.84546
Below Best %	38.1%	38.1%	38.0%	38.1%	38.1%	38.1%	38.2%	38.1%



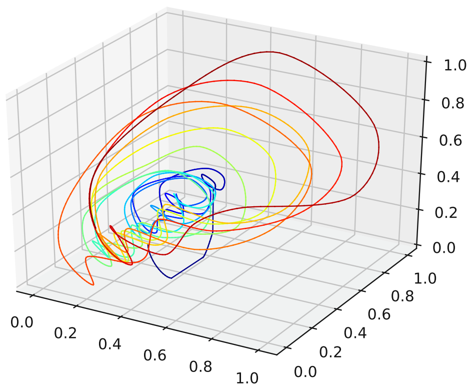
(a)



(b)



(c)



(d)

FIGURE 10.7: Interpolated data and reconstructed attractors for the monthly international airline passengers data set.
(a): The original and interpolated time series data;
(b): Phase space reconstruction of the original data;
(c): Phase space reconstruction of the average population data;
(d): Phase space reconstruction of the genetic-algorithm-improved data.

10.1.2.5 Monthly Mean Temperature in Nottingham Castle

This data set is discussed in Section 7.2. This time series shows strong seasonal regularities and behaves stationary, as no linear increasing or decreasing trend is visible.

The results on how well the presented interpolation can reproduce missing data points of this data set are collected in Table 10.6 and depicted in Figure C.1 (e). The results show that, though the genetic-algorithm-improved interpolation drastically outperforms the average random interpolation, the algorithm did not once outperform the best interpolation of the population, although outperforming the linear and the cubic spline interpolation. The genetic-algorithm-improved interpolation does not perform that well for this data set compared to a random interpolation of the time series. The improved interpolation is only around the best $\approx 34\%$ for this data set. We thus conclude that the presented interpolation technique does not capture the phase-space properties of this data set very well. The corresponding time series and reconstructed phase space plots are collected in Appendix C.1.6.

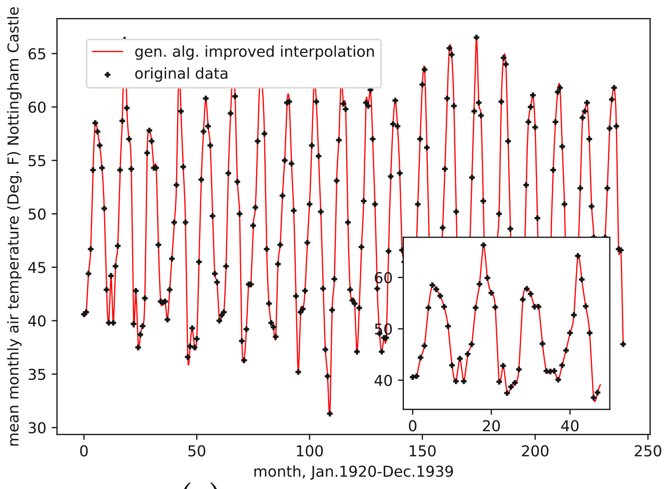
An interpolation of the original data set is depicted in Figure 10.8. We again show the population mean (c) and the improved interpolation (d). The presented interpolation technique improves the phase space portrait (d) in terms of a smoothed-out phase space trajectory compared to the original time series (b) and the population mean (c), which are both pointy and have many sharp edges. Also, given the time-series depiction of the PhaSpaSto interpolation (Figure 10.8 (a)), we see the same behavior as for all the other data sets; the major peaks are increased.

TABLE 10.6: Errors for the interpolated data on the monthly mean temperature in Nottingham castle data set depending on the number of interpolation points. The errors are shown for the mean interpolation of all populations, the lowest error in the population, and the interpolation improved using the presented genetic algorithm. We highlighted the interpolation where the genetic-algorithm-based interpolation performed best. The corresponding plots for the best interpolation are shown in Appendix C.1.6. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation.

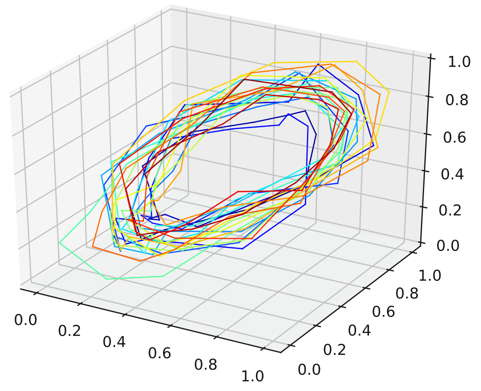
n_I	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
RMSE Population Mean	**3.09115**	3.09170	3.09167	3.09055	3.09088	3.09055	3.09166	3.09165
Lowest RMSE in population	**2.47879**	2.47858	2.47910	2.47890	2.47886	2.47901	2.47900	2.47875
RMSE linear interpolated	**2.61279**	2.61279	2.61279	2.61279	2.61279	2.61279	2.61279	2.61279
RMSE spline interpolated	**2.59028**	2.59028	2.59028	2.59028	2.59028	2.59028	2.59028	2.59028
RMSE gen. alg. improved	**2.48413**	2.50179	2.50279	2.50406	2.50420	2.50521	2.50512	2.505089
Below Best %	**12.6%**	31.3%	32.4%	33.5%	33.8%	34.4%	34.4%	34.1%

n_I	**17**	**19**	**21**	**23**	**25**	**27**	**29**	**31**
RMSE Population Mean	3.09095	3.09177	3.09115	3.09122	3.09146	3.09143	3.09179	3.09023
Lowest RMSE in population	2.47887	2.47920	2.47925	2.47899	2.47867	2.47941	2.47885	2.47892
RMSE linear interpolated	2.61279	2.61279	2.61279	2.61279	2.61279	2.61279	2.61279	2.61279
RMSE spline interpolated	2.59028	2.59028	2.59028	2.59028	2.59028	2.59028	2.59028	2.59028
RMSE gen. alg. improved	2.50494	2.50541	2.50529	2.50552	2.50505	2.50547	2.50550	2.50533
Below Best %	33.9%	34.6%	34.4%	35%	34.6%	34.6%	34.7%	34.7%

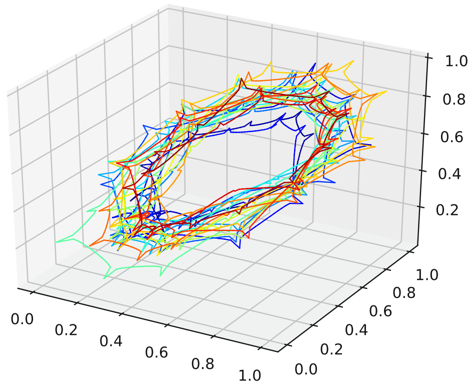
Interpolating Strange Attractors via Fractional Brownian Bridges



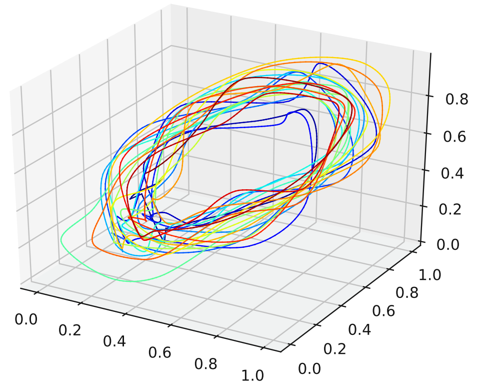
(a)



(b)



(c)



(d)

FIGURE 10.8: Interpolated data and reconstructed attractors for the monthly mean temperature in Nottingham castle data set.
(a): The original and interpolated time series data;
(b): Phase space reconstruction of the original data;
(c): Phase space reconstruction of the average population data;
(d): Phase space reconstruction of the genetic-algorithm-improved data.

10.1.2.6 Shampoo Sales

This data set and the original source are discussed in Section 7.9. Although this data set clearly shows a linear trend, no obvious regularities or seasonalities are apparent. Thus, we consider this data set to be more stochastical rather than oscillatory. We expect PhaSpaSto interpolation not to perform well on data sets like these.

Table 10.7 and Figure C.1 (f) both show the results on how well the employed interpolation techniques can reconstruct missing data points on this data set. For this data set, PhaSpaSto interpolation does not perform well at all. The best performance is achieved by the random interpolation with the lowest error, followed by the population mean and the linear interpolation. Spline interpolation performs worst on this data set. Because of its stochastic nature and no apparent seasonalities, PhaSpaSto interpolation is not a well-suited method for interpolating this data set. The corresponding time-series and reconstructed phase space plots are collected in Appendix C.1.7.

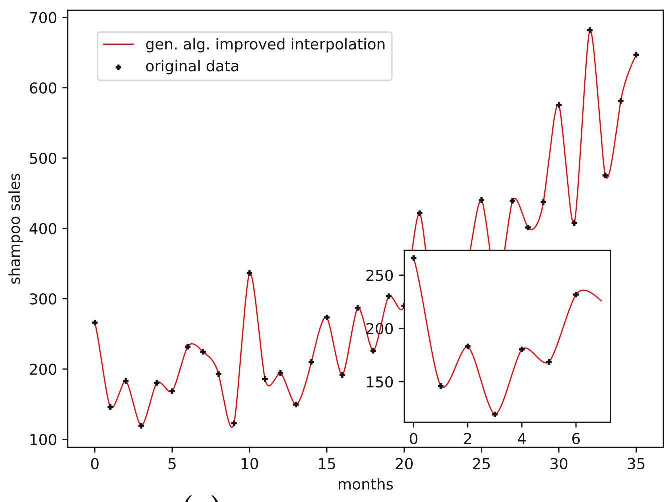
An interpolation of the original data set is depicted in Figure 10.9. We again show the population mean (c) and the improved interpolation (d). The presented interpolation technique improves the phase space portrait (d) in terms of a smoothed-out phase space trajectory compared to the original time series (b) and the population mean (c), which are both pointy and have many sharp edges. Also, given the time-series depiction of the PhaSpaSto interpolation (Figure 10.9 (a)), we see that PhaSpaSto interpolation slightly increases some of the major peaks, but overall gives an interpolation which is similar to what we would expect from a spline interpolation of the data set.

Interpolating Strange Attractors via Fractional Brownian Bridges

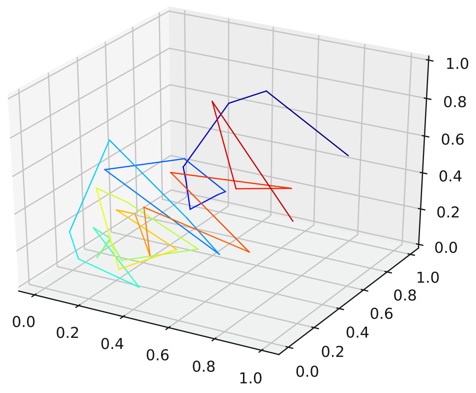
TABLE 10.7: Errors for the interpolated data on the shampoo sales data set depending on the number of interpolation points. The errors are shown for the mean interpolation of all populations, the lowest error in the population, and the interpolation improved using the presented genetic algorithm. We highlighted the interpolation where the genetic-algorithm-based interpolation performed best. The corresponding plots for the best interpolation are shown in Appendix C.1.7. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation.

n_I	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
RMSE population mean	93.77985	93.78096	93.77794	93.78474	93.78852	93.78078	93.78285	93.78171
Lowest RMSE in population	75.17553	75.01493	75.04488	75.13511	75.02854	75.20329	75.08445	75.12778
RMSE linear interpolated	100.56301	100.56301	100.56301	100.56301	100.56301	100.56301	100.56301	100.56301
RMSE spline inteprolated	108.02059	108.02059	108.02059	108.02059	108.02059	108.02059	108.02059	108.02059
RMSE gen. alg. improved	110.19816	105.25685	106.31509	105.51261	105.50765	103.98034	105.90568	105.86858
Below Best %	99.5%	99.4%	99.5%	99.5%	99.5%	99.5	99.5%	99.5%

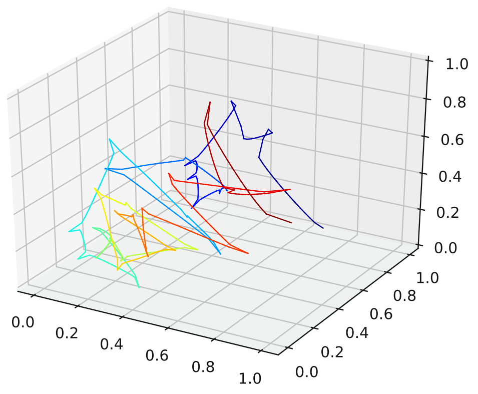
n_I	**17**	**19**	**21**	**23**	**25**	**27**	**29**	**31**
RMSE population mean	93.78587	93.78108	93.77988	**93.78166**	93.78444	93.77972	93.78210	93.78348
Lowest RMSE in population	75.12903	75.08101	75.26534	**75.15750**	75.12706	75.17626	75.03625	75.05900
RMSE linear interpolated	100.56301	100.56301	100.56301	**100.56301**	100.56301	100.56301	100.56301	100.56301
RMSE spline inteprolated	108.02059	108.02059	108.02059	**108.02059**	108.02059	108.02059	108.02059	108.02059
RMSE gen. alg. improved	105.44456	105.51313	105.51292	**103.69012**	104.71252	104.01555	103.73095	103.88873
Below Best %	99.5%	99.5%	99.5%	**99.5%**	99.5%	99.5	99.5%	99.5%



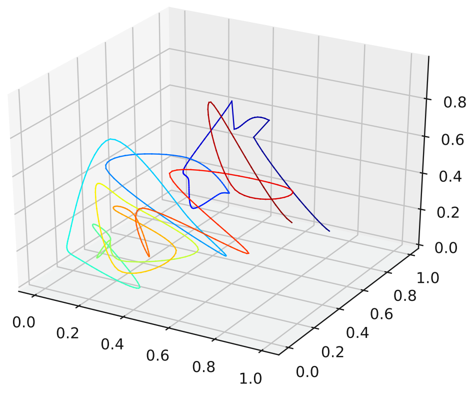
(a)



(b)



(c)



(d)

FIGURE 10.9: Interpolated data and reconstructed attractors for the monthly mean temperature in Nottingham castle data set.
(a): The original and interpolated time series data;
(b): Phase space reconstruction of the original data;
(c): Phase space reconstruction of the average population data;
(d): Phase space reconstruction of the genetic-algorithm-improved data.

Interpolating Strange Attractors via Fractional Brownian Bridges

10.1.2.7 Annual Maize Yields in Austria

The original source and a discussion on this data set can be found in Section 7.7. Like the shampoo sales data set, this data set does not provide us with visible seasonalities but an overall random behavior and a visible increasing linear trend.

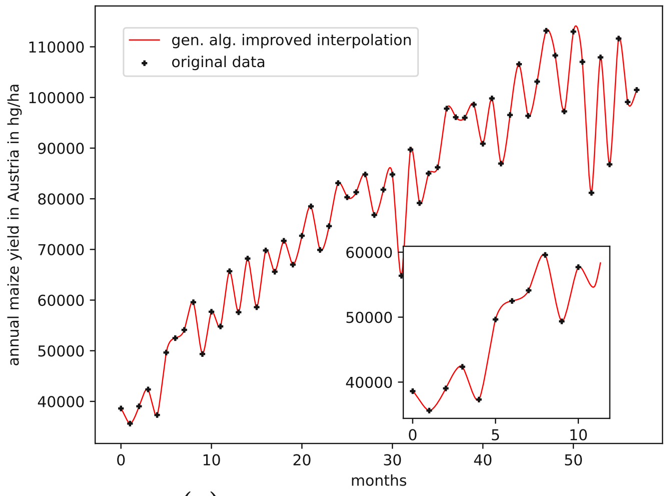
The results on how well the presented interpolation can reproduce missing data points of this data set are collected in Table 10.8 and depicted in Figure C.1 (g). PhaSpaSto interpolation performs second-worst on this data set. Spline interpolation performs worst, and the random interpolation with the lowest error performs best. The second-best is the average interpolation of all random interpolations. The third-best is the linear interpolation, thus concluding that a random or a linear interpolation is a better choice on data sets with no apparent trends. The corresponding time-series and reconstructed phase space plots are collected in Appendix C.1.8.

An interpolation of the original data set is depicted in Figure 10.10. We again show the population mean (c) and the improved interpolation (d). PhaSpaSto interpolation improves the phase space portrait (d) in terms of a smoothed-out phase space trajectory compared to the original time series (b) and the population mean (c), which are both pointy and have many sharp edges. Also, given the time-series depiction of the PhaSpaSto interpolation (Figure 10.10 (a)), we see similar behavior as for all the other data sets, some major peaks are increased, and overall the interpolation is very much how one would expect a spline interpolation to look like.

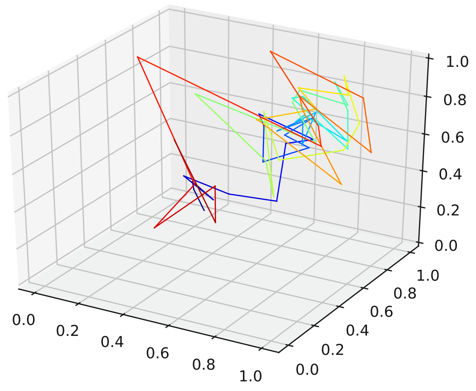
Table 10.8: Errors for the interpolated data on the annual maize yields in Austria data set depending on the number of interpolation points. The errors are shown for the mean interpolation of all populations, the lowest error in the population, and the interpolation improved using the presented genetic algorithm. We highlighted the interpolation where the genetic-algorithm-based interpolation performed best. The corresponding plots for the best interpolation are shown in Appendix C.1.8. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation.

n_I	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
RMSE population mean	**9467.34946**	9467.34909	9467.34932	9467.35117	9467.34672	9467.349182	9467.35129	9467.35127
Lowest RMSE in population	**8552.33575**	8552.40507	8552.33623	8552.26190	8552.26676	8552.30054	8552.37868	8552.27861
RMSE linear interpolated	**9641.12086**	9641.12086	9641.12086	9641.12086	9641.12086	9641.12086	9641.12086	9641.12086
RMSE spline inteprolated	**10655.09616**	10655.09616	10655.09616	10655.09616	10655.09616	10655.09616	10655.09616	10655.09616
RMSE gen. alg. improved	**10204.70770**	10400.17016	10404.69095	10401.94629	10401.12925	10401.11260	10400.83561	10401.01686
Below Best %	**96.0%**	96.0%	96.0%	96.0%	96.0%	96.0%	96.0%	96.0%

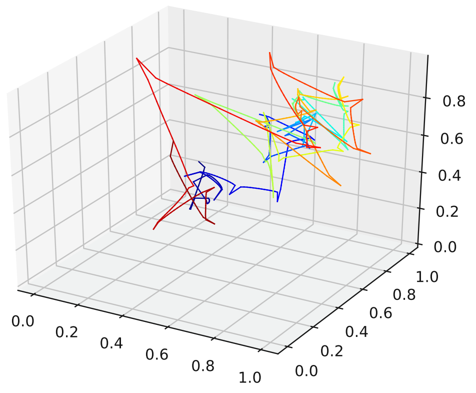
n_I	**17**	**19**	**21**	**23**	**25**	**27**	**29**	**31**
RMSE population mean	9467.35137	9467.35207	9467.34967	9467.34998	9467.35055	9467.35120	9467.35351	9467.35240
Lowest RMSE in population	8552.36502	8552.32328	8552.21861	8552.15975	8552.32538	8552.24323	8552.28914	8552.36142
RMSE linear interpolated	9641.12086	9641.12086	9641.12086	9641.12086	9641.12086	9641.12086	9641.12086	9641.12086
RMSE spline inteprolated	10655.09616	10655.09616	10655.09616	10655.09616	10655.09616	10655.09616	10655.09616	10655.09616
RMSE gen. alg. improved	10400.94068	10401.32128	10401.59039	10402.33613	10401.94342	10401.55254	10401.35126	10401.15196
Below Best %	96.0%	96.0%	96.0%	96.0%	96.0%	96.0%	96.0%	96.0%



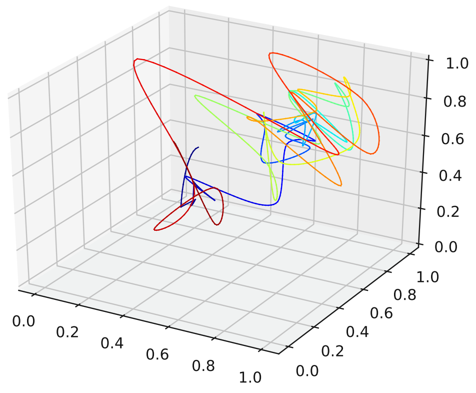
(a)



(b)



(c)



(d)

FIGURE 10.10: Interpolated data and reconstructed attractors for the annual maize yields in Austria data set.
(a): The original and interpolated time series data;
(b): Phase space reconstruction of the original data;
(c): Phase space reconstruction of the average population data;
(d): Phase space reconstruction of the genetic-algorithm-improved data.

10.2 Summary

We briefly summarize this chapter and point out the main findings:

- We presented a genetic algorithm to improve a stochastic interpolation, i.e., multi-point fractional Brownian bridges, to enhance the reconstructed phase space of any given time series. For simplicity, we named this method PhaSpaSto interpolation.

- We presented a novel approach to measure the quality of a phase space reconstruction according to Takens' theorem. Here we used an idea from image processing, i.e., to identify blurry images via the variance of second derivatives. These second derivatives are calculated along the reconstructed phase space curve for any given reconstructed phase space. We use the variance of these second derivatives to measure the quality of our phase space reconstruction. Given two interpolated phase space curves of the same time series, the one with the lower variance of second derivatives along the curve is the better phase space reconstruction, as it is the smoother one.

  We further need to mention that the applied smoothness criterion provides a smooth trajectory and curve for arbitrary embedding dimensions, which means that, however we choose the embedding dimension, the resulting variability in the actual 1-dimensional interpolated curve is minor. Thus, to the best of our knowledge, these minor differences in the interpolations won't matter for all applications the author can think of. However, given the strong dependence on minor changes in a chaotic system, we cannot exclude that one might come up with a scenario where these differences matter. We discuss this issue in detail in Appendix C.4.

- We showed that the developed technique performed well in the case of a model data set, i.e., one variable of the Lorenz system. Here we deleted data points from the original time series data and were able to outperform, in some cases, any best random interpolations of this time series data. Also, the presented method outperformed linear interpolation when finding the missing data points. Still, the proposed method did not outperform cubic spline interpolation on this task. This was done to validate our method and to show its applicability. Further, the presented reconstructed phase spaces plots show that the interpolated phase space reconstruction is similar to the original reconstructed phase space. The results for the Lorenz system are collected in Section 10.1.1.

  However, PhaSpaSto-interpolation is a much more expensive way to interpolate time series data than all other shown techniques. This issue is also discussed in Appendix C.5, where we show the difference between calculation times in detail.

Interpolating Strange Attractors via Fractional Brownian Bridges

however, we need to mention that it takes approximately 3.4×10^6 times longer than linear interpolation and 6.7×10^6 longer than the employed cubic spline interpolation to perform PhaSpaSto interpolation.

- We demonstrated the presented method using seven sparsely sampled non-model data sets. The validation was done by removing every second data point from the original time series and reconstructing these missing data points using the developed technique. For three out of seven data sets, the developed method effectively can identify the interpolations or parts of it with low errors, i.e., the result is around the best 1% of the population in terms of the RMSE for the reconstructed data points. PhaSpaSto interpolation outperformed the spline interpolation for six of seven non-model data sets and the linear interpolation on five non-model data sets. Also, the best random interpolation beat the cubic spline interpolation on six non-model data sets. For the monthly airline passengers data set, the PhaSpaSto interpolation does not perform very well as it is only around the best $30 - 40\%$ of all RMSEs of the population. And for the final two data sets, i.e., the shampoo sales and maize yields data sets, PhaSpaSto interpolation cannot find a meaningful interpolation and is outperformed by every other interpolation except the cubic spline interpolation. The interpolation performed well for the measles cases in NYC data set (Section 10.1.2.1), the car sales in Quebec data set (Section 10.1.2.2) and the Perrin Freres champagne sales data set (Section 10.1.2.3), which are data sets that show regularities and oscillatory behavior. The cases where the presented method did not perform well are the monthly international airline passengers data set (Section 10.1.2.4), the monthly mean temperature in Nottingham castle data set (Section 10.1.2.5), the shampoo sales data set (Section 10.1.2.6) and the annual maize yields in Austria data set (Section 10.1.2.7). We conclude that PhaSpaSto interpolation can retrieve missing data points for time series with seasonal behavior or oscillatory regularities better than for more stochastic/random data sets.

- We also used the seven non-model data sets to show the applicability of the developed technique as an actual interpolation technique, i.e., no deleted data points. The plots of the reconstructed phase spaces show that it softens the edges and provides a smoother and cleaner reconstructed phase space trajectory. Therefore the authors conclude that this technique applies to arbitrary univariate data sets. All of these plots are collected in Section 10.1.2. We recommend it when dealing with sparsely sampled seasonal time series or data sets that show oscillatory regularities.

Chapter 11

Reconstructed Phase Spaces and Neural Network Time Series Predictions

This chapter, in addition to the previous Chapter 9, presents another method to filter predictions, i.e., based on their phase space trajectories. This idea is based on the findings of Chapter 10. To be specific, the variance of second derivatives along a phase space trajectory can be used to identify good interpolation points, as shown in Chapter 10. We employ this idea to identify good predictions in an ensemble forecast. Further, this chapter is a continuation of the methods presented in Chapter 9 and applies the proposed ideas to new data sets, such as the Lorenz system or the annual yield data sets. This chapter contains work and results published in [31]:

Sebastian Raubitzek and Thomas Neubauer. Reconstructed phase spaces and lstm neural network ensemble predictions. Engineering Proceedings, 18(1), 2022. ISSN 2673-4591. doi: 10.3390/engproc2022018040. URL `https://www.mdpi.com/2673-4591/18/1/40`. Visited on 2023-04-20.

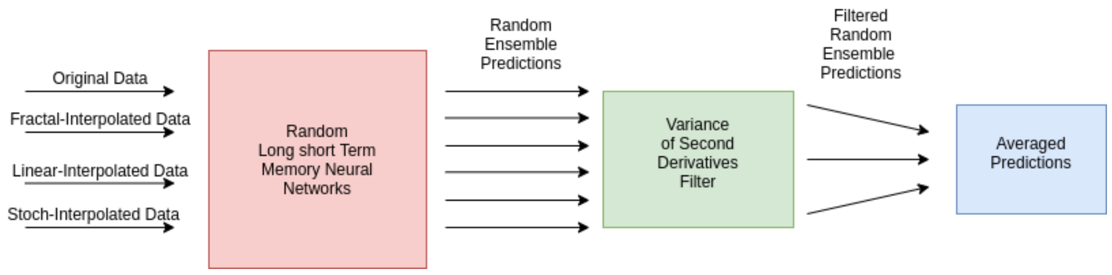

Original Data
Fractal-Interpolated Data
Linear-Interpolated Data
Stoch-Interpolated Data
Random Long short Term Memory Neural Networks
Random Ensemble Predictions
Variance of Second Derivatives Filter
Filtered Random Ensemble Predictions
Averaged Predictions

FIGURE 11.1: Schematic depiction of the filtering process. The whole pipeline is applied, first, to the original non-interpolated data, second, the fractal interpolated data, and third, the linear-interpolated.

The presented experiments aim to show the applicability of PhaSpaSto interpolation to improve neural network time series predictions and if the variance of second derivatives along a reconstructed phase space trajectory can be used to filter ensemble predictions. Further, the presented selection of time series data serves to understand for which kind of time series data the developed randomly parameterized neural networks approach works.

This chapter builds on the two previous Chapters 9 and 10, meaning that we are again employing the randomly parameterized long short-term memory neural networks (Section 6.1.3). And again, we are using different types of interpolated time series data, just as in Chapter 9. Contrary to Chapter 9 we are using one additional interpolation technique, i.e. the attractor-based stochastic interpolation method (the PhaSpaSto interpolation) from Section 5.2. To save computational resources and to take into account the results from Chapters 9[1] and 10[2] we chose the number of interpolation points to be $N_I = \{9, 11, 13, 15\}$. Further, this selection of interpolation points is motivated by the fact that PhaSpaSto interpolation requires a certain amount of interpolation points to increase the smoothness of a given phase space trajectory (Also discussed in Chapter 10). We discuss the right amount of interpolation points in Section 13.1. We filter the multitude of predictions based on their reconstructed phase space properties using the variance of second derivatives along a phase space trajectory, discussed in Section 5.2.2. The whole scheme is depicted in Figure 11.1. Additionally, we provide the framework from Chapter 9 as an additional baseline prediction, i.e., the multitude of autoregressive predictions is also analyzed using the prediction filters based on measures of signal complexity, discussed in Section 9.3.

11.1 Datasets

This prediction experiment uses a total of eleven different data sets, where ten of the discussed data sets are not analyzed in Chapter 9 and Ref. [32]. The remaining data set, i.e., the monthly international airline passengers data set (Section 7.1), which we already analyzed in Chapter 9, is shown here for comparison. The discussed data sets and the corrresponding train/test splits are:

1. The Lorenz system, see Section 7.15; Train/test-split: 80/20

2. Annual maize yields Austria, see Section 7.7; Train/test-split: 80/20

[1]This chapter did not find a trend for the best number of interpolation points to improve a neural network time series prediction.

[2]This chapter suggests that ≈ 13 interpolation points provide a good phase space reconstruction

3. Annual wheat yields Austria, see Section 7.6; Train/test-split: 80/20
4. Measles cases in NYC, see Section 7.5; Train/test-split: 80/20
5. Monthly international airline passengers, see Section 7.1; Train/test-split: 70/30
6. Canadian Lynx, see Section 7.10; Train/test-split: 88/12
7. River Krems discharge, see Section 7.12; Train/test-split: 80/20
8. USD/GBP exchange rate, see Section 11.4.9; Train/test-split: 93/07
9. Dow Jones Industrial Average daily close in 2018, see Section 7.11; Train/test-split: 80/20
10. Sunspots, see Section 7.13; Train/test-split: 77/23
11. Shampoo Sales, see Section 7.13; Train/test-split: 80/20

The data sets are chosen to cover a range of different problems. We chose the Lorenz system as it is an example of a deterministic chaotic system. We included the annual maize yields, the annual wheat yields, and the river Krems discharge data set to test the applicability of the proposed approach to data related to Austrian Agriculture. The Measles outbreaks in NYC data set was chosen as it is known to be a real-life data set with an attractor structure in reconstructed phase space, [92]. The Canadian Lynx, USD/GBP exchange rate, Monthly international airline passengers, and the sunspots data set were chosen to compare our results to state-of-the-art time series prediction approaches, [93]. We added the Dow Jones Industrial Average daily close in 2018 and the shampoo sales data set to increase the amount of non-seasonal data sets.

11.2 LSTM ensemble predictions

For each time-series data, we employ ensembles of randomly parameterized LSTM neural networks, see Sections 6.1.2 and 6.1.3. Here we use the second of the two described architectures. The idea here is not to optimize the neural networks but to generate many different ones. I.e., train and predict the data under study using many different neural networks and then average them to obtain a final result/prediction. One thousand of these networks are created and trained to predict the unknown data for the maize and wheat yields. Because of limited computational resources, we create five hundred of these neural networks and predictions for all other data sets. In a later step, these predictions are filtered based on their phase space properties.

11.2.1 Data preprocessing

The data was preprocessed using the following two data manipulations:

First, the data $X(t)$ defined at discrete time intervals $t = v, 2v, 3, ..., kv$, was scaled to $X(t) \in [0.1, 0.9]$, $\forall t$. Second, the maize yields, the wheat yields, the airline passengers, and the shampoo sales data sets are detrended by subtracting a linear fit, thus making the data stationary.

11.3 Ensemble Filters/Data Postprocessing

As previously mentioned, the randomly parameterized LSTM neural networks produced many predictions for each time-series data. The results of the averaged non-filtered predictions can be found in Appendix D.3.

These unfiltered predictions are still far off a good forecast because many wrong predictions are among them. Here the hypothesis is that good predictions have similar phase space properties as the original data. Thus the predictions for each time series data are filtered regarding their phase space properties, i.e., using the variance of second derivatives along the reconstructed phase space trajectory to discard wrong predictions. This is done in two ways, i.e., two filters are developed and employed:

- We randomly chose 1 to 10 predictions from the whole set of predictions. Next, these predictions are averaged to form an ensemble prediction. This ensemble prediction is merged with the training data. Then, we analyze the variance of second derivatives along the phase space trajectory. We repeat this process 1 million times. Then we keep the set of averaged predictions with the lowest variance of second derivatives. On all plots, this procedure is referred to as `loss_rand`[3].
- All predictions are merged with the training data. We calculate the variance of second derivatives along the reconstructed phase space trajectory. The ten predictions with the lowest variance are used to form an averaged ensemble prediction. This filter is referred to as `los2_rand`.

For the phase space embeddings in this chapter, we use $\tau = 1$ and $d_E = 3$. This choice of phase space embeddings is further discussed in Appendix A. We determined the time delay for the Lorenz system using the method based on the average mutual information and set $d_E = 3$ because this is known from the literature; [92, 99].

[3]This name refers to the name used in the implementation

11.4 Results

In this chapter, we test the applicability of the PhaSpaSto interpolation method and the corresponding loss function, i.e. the variance of second derivatives along the phase space trajectory, to the presented approach from Chapter 9. I.e. we generate randomly parameterized long short term memory neural networks, train them with the original, the corresponding linear, fractal and PhaSpaSto interpolated data sets and filter the autoregressive predictions based on their complexity (As in Chapter 9) and based on their phase space properties using the variance of second derivatives along the reconstructed phase space trajectories. Also, the discussed model data, i.e. the Lorenz system, features a fine-grained/original data set and a sparsely sampled data set. This sparsely sampled data set is based on the fine-grained data set, but many data points were deleted with resepect to the employed interpolation technique and interpolation points, as decscribed in Section 10.1.1. The results are presented in Tables 11.4, 11.2, 11.3, 11.1 and 11.5. Further, the plots for the best prediction for both, the phase-space-based and complexity-based filters, are shown in Figures 11.5,11.3,11.4, 11.2 and 11.6. The plots for all results not shown on this chapter but featured in the previously mentioned tables are collected in Appendix D.1

We list the baseline predictions for all data sets, except for the Lorenz system and the monthly international airline passengers, in Appendix D.2. The baseline predictions for the monthly international airline passengers data set are listed in Appendix B.1. We do not provide a baseline prediction for the Lorenz system. This is because of the chaotic nature of the system. Given that we're using different lengths of the data set with respect to their interpolation points, the data sets for each number of interpolation points slightly differ. Thus we cannot provide one baseline prediction for all data excerpts from the Lorenz system. Further, as suggested in [109], a neural network with a single hidden layer might not be able to predict the Lorenz system, as only sophisticated hybrid data assimilation approaches, e.g., Kalman filters in combination with neural networks or similar techniques seem to be capable of reproducing the Lorenz system.

11.4.1 The Lorenz System

The results for the Lorenz system (Section 7.15), Tables 11.1 and the corresponding Figures 11.2, D.1 and D.2 show that the presented approach can not predict the Lorenz system. The predictions are completely off and only partially capture some of the characteristic behavior of the Lorenz system. Still, the results where the prediction did capture some of the characteristics suggest that there may be a way to improve these predictions drastically. Two ways to do this would be to, first, improve the neural network architecture, and second, to train on a longer data set. Overall, we conclude that the Lorenz system can not be predicted using the presented ideas effectively.

TABLE 11.1: Best results for the Lorenz system.
Left: phase space filter predictions
Right: Signal complexity filter predictions

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
los2_rand	Fractal Interpolated	13	0.20360	±0.03985
loss_rand	Fine-Grained Model	13	0.23426	±0.01975
los2_rand	Not Interpolated	-	0.24299	±0.01864
los2_rand	Not Interpolated	-	0.25770	±0.01242
loss_rand	Stoch. Interpolated	15	0.26656	±0.02109

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
Hurst	Linear Interpolated	13	0.19044	±0.03529
Shannon	Linear Interpolated	13	0.19102	±0.01867
Hurst Fisher	Stoch. Interpolated	13	0.19122	±0.02710
Hurst SVD	Stoch. Interpolated	13	0.19122	±0.02710
Fisher Hurst	Stoch. Interpolated	13	0.19122	±0.02710

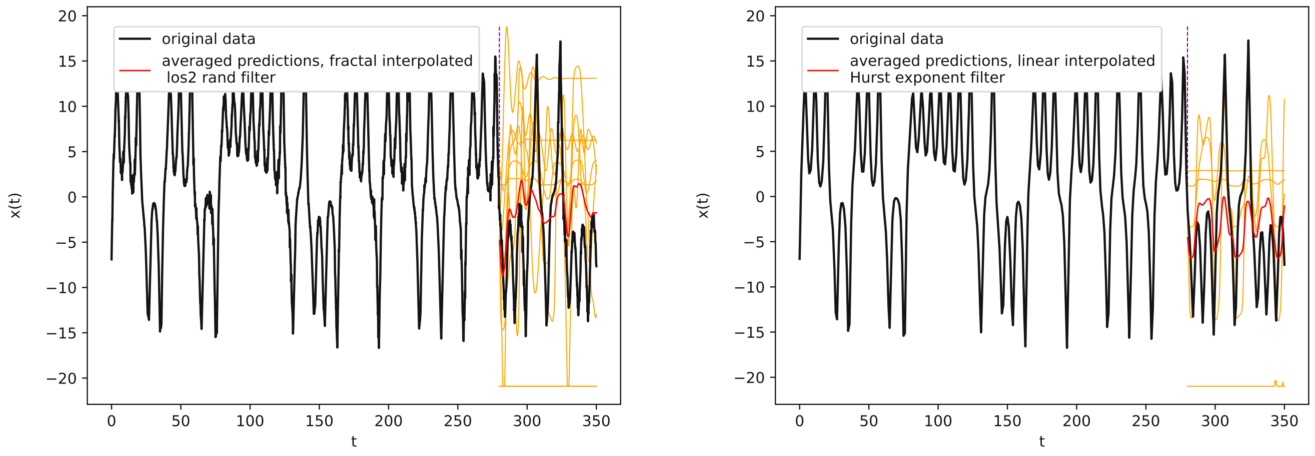


FIGURE 11.2: Best predictions for the Lorenz system.
Left: los2_rand-filter, fractal interpolated, 13 interpolation points, RMSE=0.20360±0.03985
Right: Hurst-filter, linear interpolated, 13 interpolation points, RMSE=0.19044±0.03529

11.4.2 Annual Maize Yields Austria

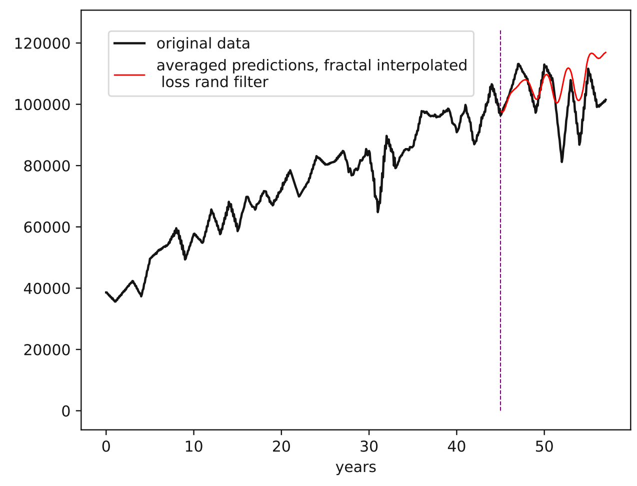
The results for the annual maize yields in Austria data set (Section 7.7) are shown in Tables 11.2 and the corresponding Figures 11.3, D.3 and D.4.

The filters based on measures of signal complexity outperform the filters based on the second derivatives of reconstructed phase space trajectories, as can be seen in Table 11.2. The left side shows the errors for phase-space-based filters, and the right side the complexity-based filters. The best result is achieved using a linear interpolated data set with 15 additional data points and filters based on the Hurst exponent and the spectrum of Lyapunov exponents. Though this was overall the best prediction for this data set, the error ΔRMSE indicates a higher variability than for the second-best results and the best result for the phase-space-based filters. This is especially true because the phase-space-based filter picked only one prediction out of 1000. Thus no variability is given for this prediction. The corresponding plots in Figure 11.3 depicts the best results. Here the left side shows the best result for the phase-space-based filters, whereas the right side presents the best result for the complexity-based filters. The complexity-based-filtered result also depicts the increased prediction variability in terms of the yellow lines contributing to the ensemble prediction. Also, as shown on the right side of Figure 11.3 the best results reproduce a mean prediction. Thus we conclude that this data set can effectively not be predicted. Still, given the left side of 11.3, we can see that the presented approach somehow reproduces the first two peaks, and results on this data sets might be improved in the future.

The two best predictions, i.e. the predictions depicted in Figure 11.3, outperformed all baseline predictions from Appendix D.2. The baseline RMSEs are 0.34020 for the LSTM, 0.35033 for the GRU, and 0.38684 for the recurrent neural network.

TABLE 11.2: Best results for the annual maize yields in Austria data set.
Left: Phase space filter predictions
Right: Signal complexity filters predictions

Filter	Interpolation Technique	N_I	RMSE	ΔRMSE	Filter	Interpolation Technique	N_I	RMSE	ΔRMSE
loss_rand	Fractal Interpolated	11	0.13939	±0.00000	Hurst Lyap	Linear Interpolated	15	0.13385	±0.04908
loss_rand	Stoch. Interpolated	9	0.13961	±0.01087	Fisher Hurst	Stoch. Interpolated	11	0.13511	±0.03642
loss_rand	Stoch. Interpolated	13	0.14022	±0.00567	Lyap Hurst	Stoch. Interpolated	15	0.13654	±0.03487
los2_rand	Stoch. Interpolated	9	0.14320	±0.00801	Fisher Lyap	Linear Interpolated	15	0.13759	±0.04690
los2_rand	Stoch. Interpolated	15	0.14414	±0.00651	SVD Lyap	Linear Interpolated	15	0.13759	±0.04690



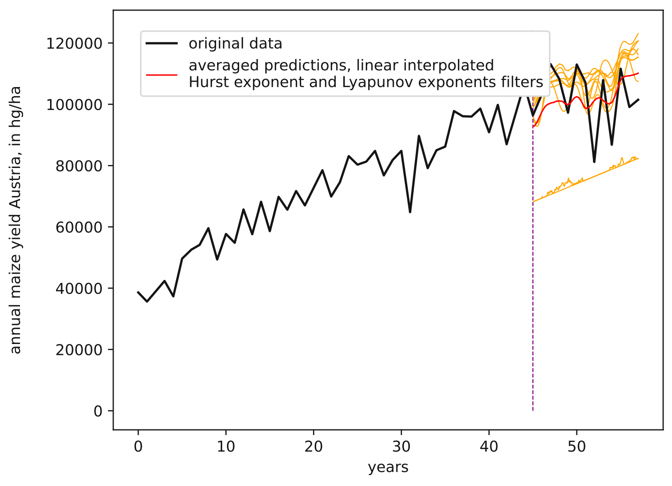


FIGURE 11.3: Best predictions for the annual maize yields in Austria data set.
Left: loss_rand-filter, fractal interpolated, 11 interpolation points, RMSE=0.13939±0.00000
Right: Hurst-filter, linear interpolated, 13 interpolation points, RMSE=0.13385± 0.04908

11.4.3 Annual Wheat Yields Austria

The results for the annual wheat yields in Austria data set (Section 7.6) are shown in Tables 11.3 and the corresponding Figures 11.4, D.5 and D.6.

The filters based on measures of the second derivatives of reconstructed phase space trajectories outperform the filters based on signal complexity, as can be seen in Table 11.3. Here the left side shows the errors for phase-space-based filters, and the right side lists the complexity-based filters. The best result is achieved using a fractal interpolated data set with nine additional data points and the los2_rand filter. The corresponding plots in Figure 11.4 depict the best results. Here the left side shows the best result for the phase-space-based filters, whereas the right side presents the best result for the complexity-based filters. The complexity-based-filtered result also depicts the increased prediction variability in terms of the yellow lines contributing to the ensemble prediction. Also, as depicted in the left side of Figure 11.4, the best result is far off right from the start but hits some right points later. Thus we conclude that this data set can not be predicted accurately using the presented approach. Still, the presented approaches can provide a rough estimate for future values of this data set.

The two best predictions, i.e., the predictions depicted in Figure 11.4, outperformed all baseline predictions from Appendix D.2. The baseline RMSEs are 0.31402 for the LSTM, 0.31784 for the GRU, and 0.28145 for the recurrent neural network.

TABLE 11.3: Best results for the annual wheat yields in Austria data set.
Left: Phase space filter predictions
Right: Signal complexity filter predictions

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**	**Filter**	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
los2_rand	Fractal Interpolated	9	0.11377	±0.02472	Lyap SVD	Fractal Interpolated	9	0.11517	±0.04367
loss_rand	Stoch. Interpolated	15	0.12246	±0.01147	Fisher SVD	Fractal Interpolated	9	0.11517	±0.04367
loss_rand	Stoch. Interpolated	13	0.12544	±0.00599	SVD	Fractal Interpolated	9	0.11517	±0.04367
loss_rand	Linear Interpolated	13	0.12657	±0.00000	SVD Shannon	Fractal Interpolated	9	0.11517	±0.04367
los2_rand	Stoch. Interpolated	13	0.12662	±0.02979	Lyap Fisher	Fractal Interpolated	9	0.11517	±0.04367

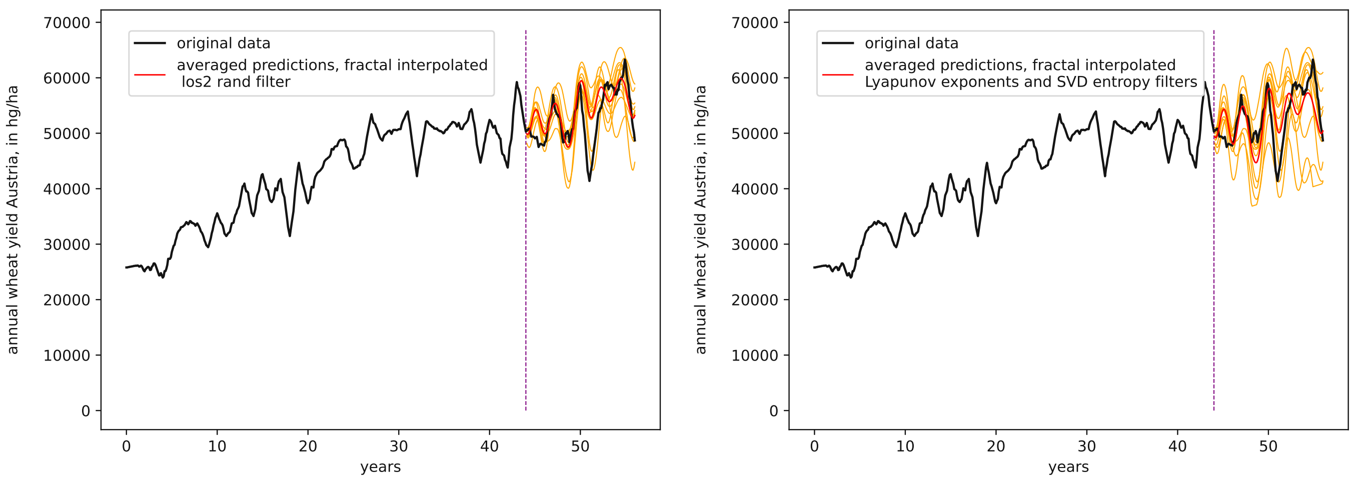


FIGURE 11.4: Best predictions for annual wheat yields in Austria data set.
Left: los2_rand-filter, fractal interpolated, 13 interpolation points, RMSE=0.11377±0.02472
Right: Hurst-filter, linear interpolated, 13 interpolation points, RMSE=0.11517±0.04367

11.4.4 Measles Cases in NYC

The results for the measles cases in NYC data set (Section 7.5), Tables 11.4 and the corresponding Figures 11.5, D.7 and D.8, show that the presented approach can predict this data set.

The filters based on measures of signal complexity outperform the filters based on the second derivatives of reconstructed phase space trajectories, as can be seen in Table 11.4. The left side shows the errors for phase-space-based filters and the right side complexity-based filters. The best result was achieved using a linear interpolated data set with 11 additional data points and filters based on SVD and Shannon's entropy. Though this was overall the best prediction for this data set, the error ΔRMSE indicates a higher variability than for the second-best results and the best result for the phase-space-based filters. The corresponding plots in Figure 11.5 depict the best results. The left side shows the best result for the phase-space-based filters, whereas the right side presents the best result for the complexity-based filters. The complexity-based-filtered result also depicts the increased prediction variability in terms of the yellow lines, which contribute to the ensemble prediction.

The two best predictions, i.e., the predictions shown in Figure 11.5, outperformed all baseline predictions from Appendix D.2. The baseline RMSEs are 0.07488 for the LSTM, 0.06591 for the GRU, and 0.07964 for the recurrent neural network.

TABLE 11.4: Best results for the measles cases in NYC data set.
Left: Phase space filter predictions
Right: Signal complexity filter predictions

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
los2_rand	Not Interpolated	-	0.04881	±0.00447
loss_rand	Linear Interpolated	11	0.05415	±0.01072
loss_rand	Linear Interpolated	15	0.06033	±0.00225
los2_rand	Fractal Interpolated	9	0.06338	±0.02327
loss_rand	Not Interpolated	-	0.06445	±0.00000

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
SVD Shannon	Linear Interpolated	11	0.02671	±0.01023
Fisher Hurst	Fractal Interpolated	9	0.03242	±0.00735
SVD Hurst	Fractal Interpolated	9	0.03242	±0.00735
Hurst SVD	Linear Interpolated	13	0.03248	±0.01705
Lyap Shannon	Linear Interpolated	9	0.03359	±0.01122

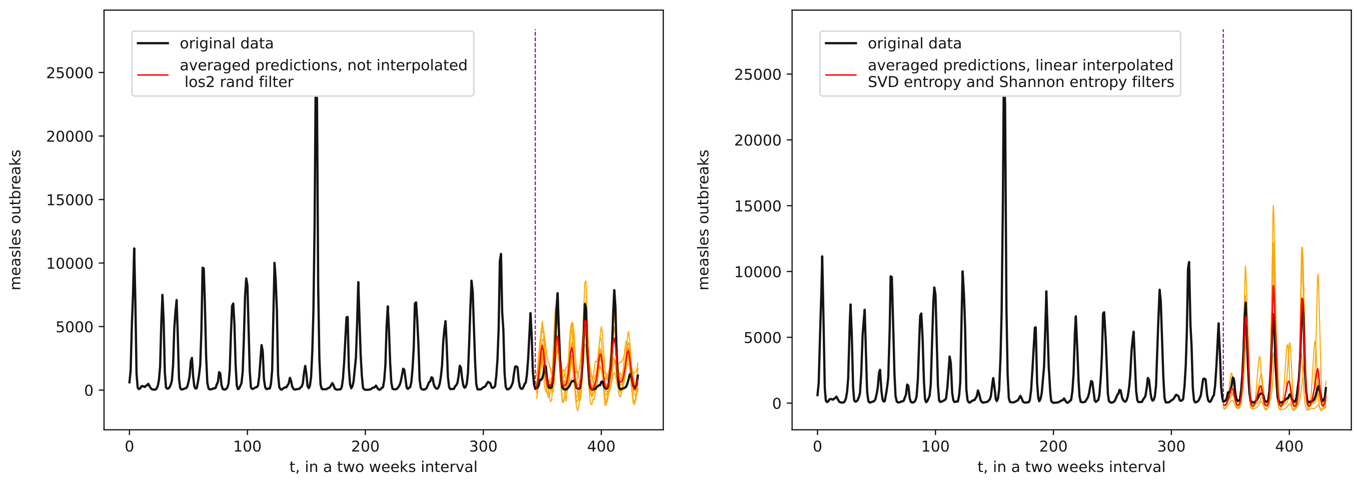


FIGURE 11.5: Best predictions for the measles cases in NYC data set.
Left: los2_rand-filter, not interpolated, RMSE=0.04881±0.00447
Right: SVD-Shannon filter, linear interpolated, 11 interpolation points, RMSE=0.02671±0.01023

11.4.5 Monthly International Airline Passengers

The results for the monthly international airline passengers data set (Section 7.1) are shown in Tables 11.5 and the corresponding Figures 11.6, D.9 and D.10.

The filters based on signal complexity outperform the filters based on measures of the second derivatives of reconstructed phase space trajectories, as can be seen in Table 11.5. Here the left side shows the errors for phase-space-based filters, and the right side lists the complexity-based filters. The best result is achieved using a linear interpolated data set with 13 additional data points and a filter based on Fisher's information. The corresponding plots in Figure 11.6 depict the best results. Here the left side shows the best result for the phase-space-based filters, whereas the right side presents the best result for the complexity-based filters. The complexity-filtered result also depicts the increased prediction variability in terms of the yellow lines, which contribute to the ensemble prediction. In contrast, the phase-space-filtered result has no variability because the filter picked only a single prediction.

The two best predictions, i.e., the predictions presented in Figure 11.6, outperformed all baseline predictions from Appendix B.1. The baseline RMSEs are 0.11902 for the LSTM, 0.10356 for the GRU, and 0.10356 for the recurrent neural network.

TABLE 11.5: Best results for the monthly international airline passengers data set.
Left: phase space filter predictions
Right: Signal complexity filter predictions

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**	**Filter**	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
loss_rand	Fractal Interpolated	9	0.04619	±0.00000	Fisher	Linear Interpolated	13	0.04450	±0.00675
los2_rand	Fractal Interpolated	11	0.04995	±0.00534	Fisher Shannon	Linear Interpolated	13	0.04450	±0.00675
los2_rand	Fractal Interpolated	9	0.05111	±0.00789	SVD Fisher	Linear Interpolated	13	0.04450	±0.00675
loss_rand	Fractal Interpolated	13	0.06016	±0.00795	Lyap Fisher	Fractal Interpolated	9	0.04465	±0.00422
loss_rand	Stoch. Interpolated	11	0.06021	±0.00627	Shannon Fisher	Fractal Interpolated	9	0.04587	±0.00476

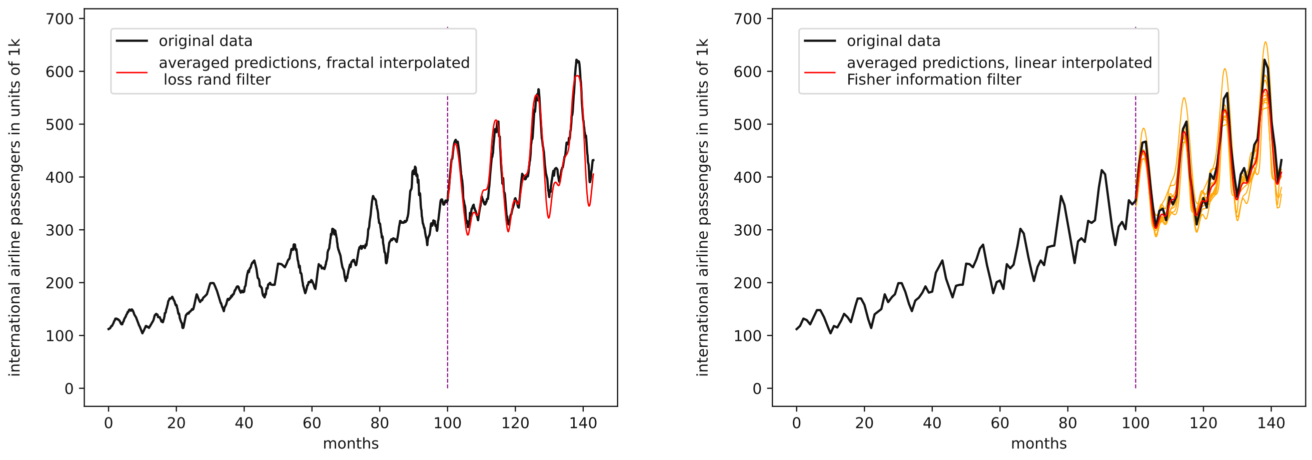


FIGURE 11.6: Best predictions for the monthly international airline passengers data set.
Left: loss_rand-filter, fractal interpolated, 9 interpolation points, RMSE=0.04619±0.00000
Right: Fisher-filter, linear interpolated, 13 interpolation points, RMSE=0.04450±0.00675

11.4.6 Canadian Lynx

The results for the Canadian Lynx data set (Section 7.10) are shown in Tables 11.6 and the corresponding Figures 11.7, D.11 and D.12.

The filters based on signal complexity outperform the filters based on the second derivatives of reconstructed phase space trajectories, as can be seen in Table 11.6. Here the left side shows the errors for phase-space-based filters, and the right side lists the complexity-based filters. The best result is achieved using a linear interpolated data set with 11 additional data points and a filter based on Lyapunov exponents. The corresponding plots in Figure 11.7 depict the best results. Here the left side shows the best result for the phase-space-based filters, whereas the right side presents the best result for the complexity-based filters. Given the presented results, we conclude that one can employ both types of filters for this data set.

The two best predictions, i.e. the predictions depicted in Figure 11.7, outperformed all baseline predictions from Appendix D.2. The baseline RMSEs are 0.15404 for the LSTM, 0.17024 for the GRU, and 0.12847 for the recurrent neural network.

TABLE 11.6: Best results for the Canadian lynx data set.
Left: phase space filter predictions
Right: Signal complexity filter predictions

Filter	Interpolation Technique	N_I	RMSE	ΔRMSE
loss_rand	Fractal Interpolated	11	0.12280	±0.10241
loss_rand	Linear Interpolated	11	0.13442	±0.00000
los2_rand	Linear Interpolated	13	0.14205	±0.04017
los2_rand	Fractal Interpolated	15	0.14345	±0.05349
loss_rand	Stoch. Interpolated	13	0.14643	±0.03443

Filter	Interpolation Technique	N_I	RMSE	ΔRMSE
Lyap	Linear Interpolated	11	0.08327	±0.04802
Shannon Hurst	Linear Interpolated	9	0.11037	±0.05037
Lyap	Stoch. Interpolated	9	0.11405	±0.03110
Lyap Shannon	Stoch Interpolated	9	0.11405	±0.03110
Fisher Lyap	Fractal Interpolated	11	0.11574	±0.05229

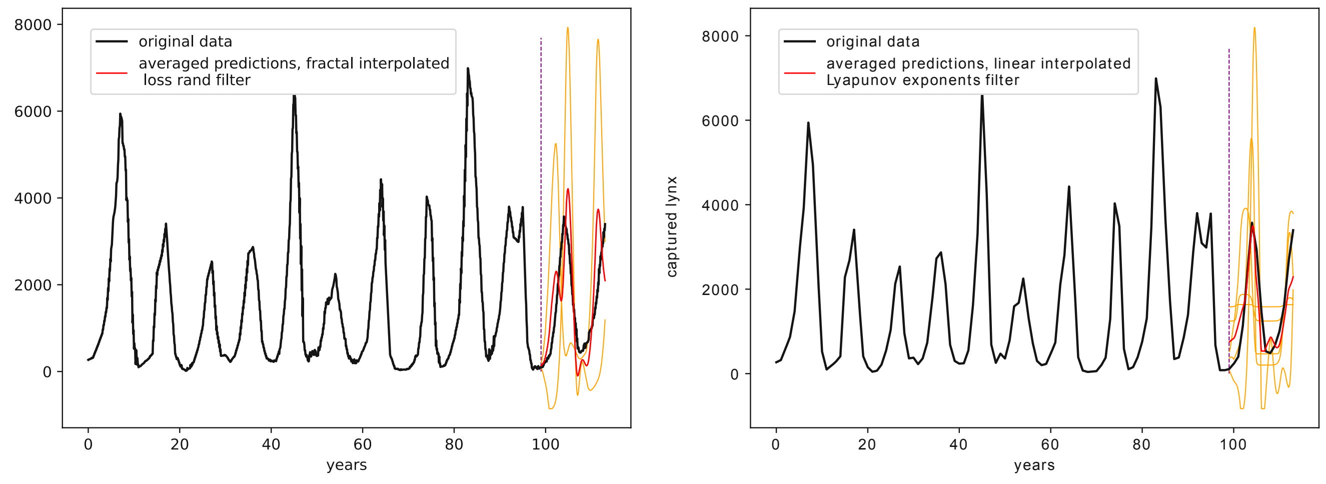


FIGURE 11.7: Best predictions for the Canadian lynx data set.
Left: loss_rand-filter, fractal interpolated, 11 interpolation points,
RMSE=0.12280±0.10241
Right: Lyap-filter, linear interpolated, 11 interpolation points,
RMSE=0.08327±0.04802

11.4.7 River Krems Discharge

The results for the River Krems discharge data set (Section 7.12) are shown in Tables 11.7 and the corresponding Figures 11.8, D.13 and D.14.

The filters based on signal complexity outperform the filters based on the variance of second derivatives along a reconstructed phase space trajectory, as presented in Tables 11.7. The left table shows the errors for phase-space-based filters, and the right table lists the complexity-based filters. We achieve the best result using a fractal interpolated data set with 13 additional data points and a filter based on Lyapunov exponents and Shannon's entropy. It's noteworthy that the combined filter based on Shannon's entropy and Fisher's information achieved the same results as the filter based on Shannon's entropy and SVD entropy. This is because Shannon's entropy and Fisher's information are based on the same single value decomposition. The corresponding plots in Figure 11.8 depict the best results. Here the left side shows the best result for the phase-space-based filters, whereas the right side presents the best result for the complexity-based filters. Given the presented results, we conclude that neither the phase-space-based nor the complexity-based filters provide convincing results. Still, the complexity-based filters outperform the phase-space-based filters in terms of RMSE. Thus, we recommend a filter based on Shannon's entropy and the spectrum of Lyapunov exponents for this data set.

For this data set, the LSTM-baseline-prediction has the lowest error overall, i.e., an RMSE of 0.13502. The RMSE for the GRU is 0.14963 and for the recurrent neural network 0.14961. All baseline predictions are collected in Appendix D.2.

TABLE 11.7: Best results for the river Krems discharge data set.
Left: phase space filter predictions
Right: Signal complexity filter predictions

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**	**Filter**	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
los2_rand	Stoch. Interpolated	11	0.15035	±0.05047	Lyap Shannon	Fractal Interpolated	13	0.13678	±0.02476
los2_rand	Linear Interpolated	9	0.15492	±0.04948	Shannon Fisher	Not Interpolated	-	0.13885	±0.01845
los2_rand	Fractal Interpolated	15	0.15943	±0.05062	Shannon SVD	Not Interpolated	-	0.13885	±0.01845
loss_rand	Linear Interpolated	9	0.16651	±0.01730	Shannon Fisher	Not Interpolated	-	0.13900	±0.01839
loss_rand	Not Interpolated	-	0.16974	±0.00000	Shannon SVD	Not Interpolated	-	0.13900	±0.01839

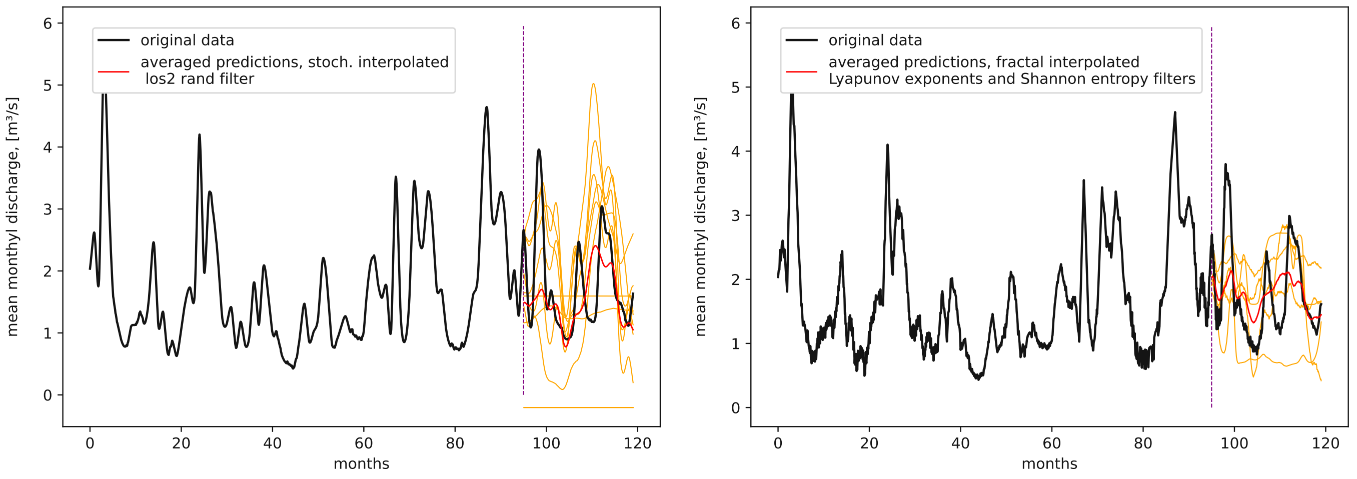


Figure 11.8: Best predictions for the river Krems discharge data set.
Left: los2_rand-filter, stoch. interpolated, 11 interpolation points,
RMSE=0.15035±0.05047
Right: Lyap-Shannon-filter, fractal interpolated, 13 interpolation points,
RMSE=0.13678±0.02476

11.4.8 Dow Jones 2018

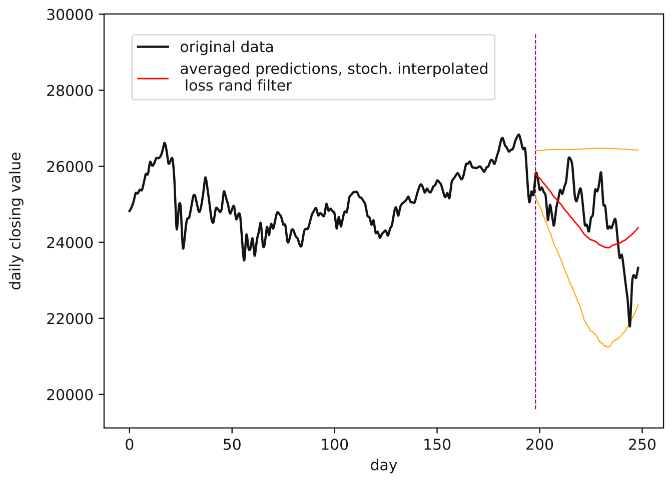
The results for the Dow Jones daily close in 2018 data set (Section 7.11) are shown in Tables 11.8 and the corresponding Figures 11.9, D.15 and D.16.

The complexity and phase-space-based filters do not seem to work on this data set as the best results merely predict the mean, depicted in Figure 11.8. As both best predictions fail, we do not conclude any insights from these results, except that the presented approaches cannot predict stock market data.

The two best predictions, i.e. the predictions depicted in Figure 11.9, outperformed all baseline predictions from Appendix D.2. The baseline RMSEs are 0.25197 for the LSTM, 0.18010 for the GRU, and 0.25112 for the recurrent neural network.

TABLE 11.8: Best results for the Dow Jones daily close in 2018 data set.
Left: phase space filter predictions
Right: Signal complexity filter predictions

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**	**Filter**	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
loss_rand	Stoch. Interpolated	11	0.17417	±0.02426	Fisher Hurst	Stoch. Interpolated	13	0.17765	±0.02802
loss_rand	Stoch. Interpolated	13	0.18660	±0.05095	SVD Hurst	Stoch. Interpolated	13	0.17765	±0.02802
los2_rand	Fractal Interpolated	13	0.21020	±0.03887	Shannon Lyap	Fractal Interpolated	15	0.18605	±0.03950
loss_rand	Fractal Interpolated	13	0.21218	±0.04764	Shannon	Stoch. Interpolated	15	0.19022	±0.01862
loss_rand	Fractal Interpolated	11	0.25052	±0.03035	Hurst Shannon	Not Interpolated	-	0.19088	±0.04289



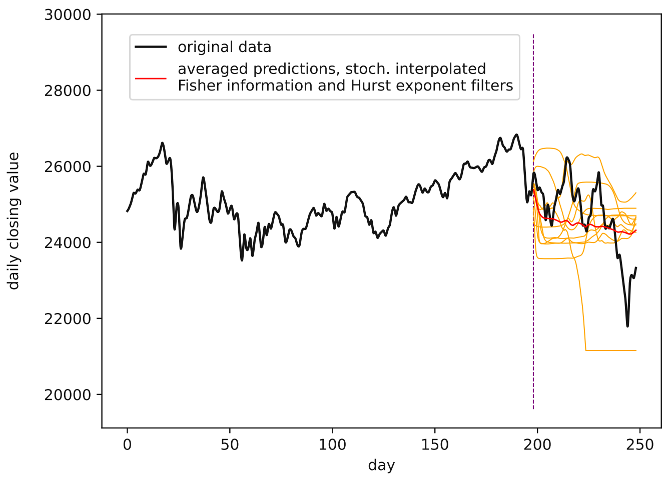


FIGURE 11.9: Best predictions for the Dow Jones daily close in 2018 data set.
Left: loss_rand-filter, stoch. interpolated, 11 interpolation points, RMSE=0.17417±0.02426
Right: Fisher-Hurst-filter, stoch. interpolated, 13 interpolation points, RMSE=0.17765±0.02802

11.4.9 USD/GBP Exchange Rate

The results for the USD/GBP exchange rate data set (Section 7.14) are shown in Tables 11.9 and the corresponding Figures 11.10, D.17 and D.18.

The results for this data set are similar to the results for the Dow Jones daily close data set (Section 11.4.8). I.e., the best results are close to a mean forecast. Thus, we

Reconstructed Phase Spaces and Neural Network Predictions

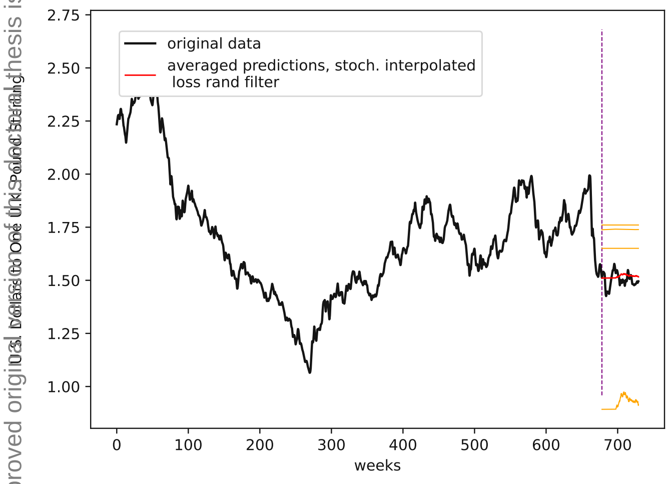
conclude that the presented approaches cannot predict this data set and subsequently are not suitable for predicting stock market and/or financial data sets.

The two best predictions, i.e. the predictions depicted in Figure 11.10, outperformed all baseline predictions from Appendix D.2. The baseline RMSEs are 0.19573 for the LSTM, 0.21395 for the GRU, and 0.25613 for the recurrent neural network.

TABLE 11.9: Best results for the USD/GBP exchange rate in 2018 data set.
Left: phase space filter predictions
Right: Signal complexity filter predictions

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
loss_rand	Stoch. Interpolated	11	0.02381	±0.03396
loss_rand	Not Interpolated	-	0.04050	±0.00000
loss_rand	Linear Interpolated	11	0.05147	±0.00333
loss_rand	Linear Interpolated	15	0.06343	±0.02411
loss_rand	Stoch. Interpolated	9	0.06592	±0.03623

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
Lyap Fisher	Fractal Interpolated	9	0.02630	±0.01617
Lyap SVD	Fractal Interpolated	9	0.02630	±0.01617
Lyap Fisher	Stoch. Interpolated	15	0.02824	±0.03077
Lyap SVD	Stoch. Interpolated	15	0.02824	±0.03077
Hurst Shannon	Linear Interpolated	11	0.03012	±0.01258



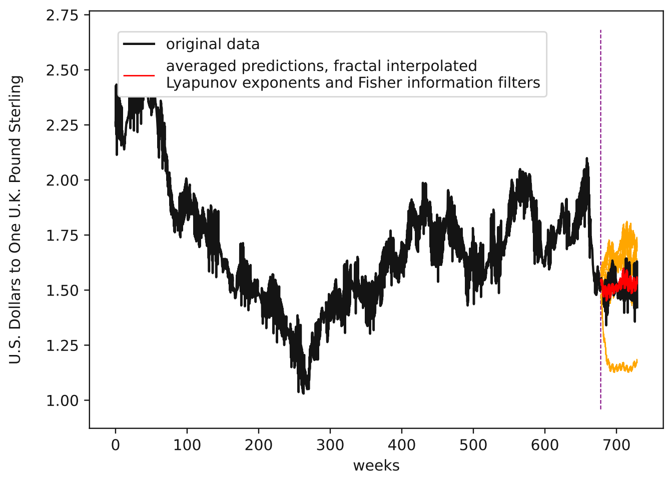


FIGURE 11.10: Best predictions for the USD/GBP exchange rate data set.
Left: loss-filter, stoch. interpolated, 11 interpolation points, RMSE=0.02381±0.03396
Right: Lyap-Fisher-filter, fractal interpolated, 9 interpolation points, RMSE=0.02630±0.01617

11.4.10 Sunspots

The results for the sunspots data set (Section 7.13) are shown in Tables 11.10 and the corresponding Figures 11.11, D.19 and D.20.

The filters based on signal complexity outperform the filters based on measures of the second derivatives of reconstructed phase space trajectories, as can be seen in Table 11.10. Here the left side shows the errors for phase-space-based filters, and the right side lists the complexity-based filters. We achieve the best result using a linear interpolated data set with 15 additional data points and a filter based on the spectrum of Lyapunov exponents. The corresponding plots in Figure 11.11 depict the best results. Here the left side shows the best result for the phase-space-based filters, whereas the right side presents the best result for the complexity-based filters. The plot for the overall best result, i.e., the complexity-filtered prediction, shows that the presented approach can somehow predict this data set as the oscillatory nature of the training data is reproduced. However, as the first three peaks are approximately right in terms of frequency, we see a slight phase shift of the predicted time series with respect to the ground truth. We expect, if specifically targeted, a short-term prediction to have increased accuracy for the major peaks, similar to the results for the monthly mean temperature in Nottingham castle data set, which can be found in Appendix C.1.6. This assumption is supported by the fact that some of the predictions, which constitute the ensemble, i.e., the yellow lines in the plot, can reproduce the major peaks quite well in terms of amplitude. We conclude that the presented approach can produce the behavior of this data set for short-term predictions.

For this data set, the GRU baseline prediction performed best with an RMSE of 0.21395, thus outperforming the two best ensemble predictions from Figure 11.11. The RMSE is 0.25354 for the LSTM baseline prediction and 0.29746 for the recurrent neural network, see Appenidx D.2.

TABLE 11.10: Best results for the sunspots data set.
Left: phase space filter predictions
Right: Signal complexity filter predictions

Filter	Interpolation Technique	N_I	RMSE	ΔRMSE
los2_rand	Linear Interpolated	11	0.29198	±0.01828
los2_rand	Linear Interpolated	15	0.29878	±0.02684
los2_rand	Fractal Interpolated	15	0.29903	±0.02257
loss_rand	Fractal Interpolated	9	0.30431	±0.02611
los2_rand	Linear Interpolated	9	0.30706	±0.01999

Filter	Interpolation Technique	N_I	RMSE	ΔRMSE
Lyap	Linear Interpolated	15	0.25339	±0.02052
Lyap Shannon	Linear Interpolated	15	0.25339	±0.02052
Shannon Hurst	Linear Interpolated	11	0.25882	±0.01969
Shannon Hurst	Not Interpolated	-	0.26277	±0.02800
Hurst	Not Interpolated	-	0.26397	±0.01979

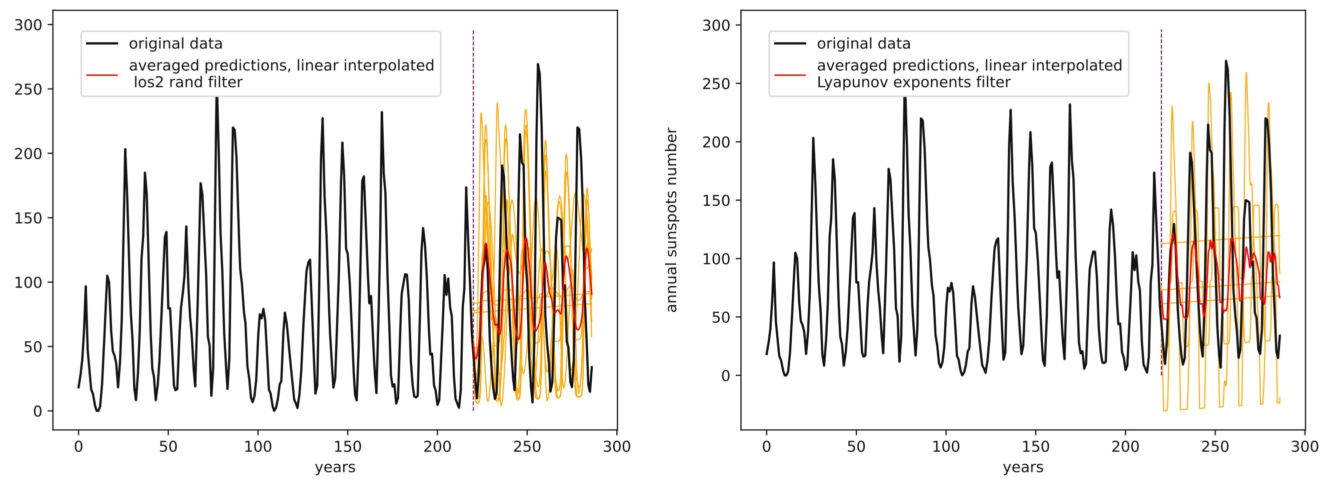


FIGURE 11.11: Best predictions for the sunspots data set.
Left: los2_rand-filter, linear interpolated, 11 interpolation points,
RMSE=0.29198±0.01828
Right: Lyap-filter, linear interpolated, 15 interpolation points,
RMSE=0.25339±0.02052

11.4.11 Shampoo Sales

The results for the shampoo sales data set (Section 7.9) are shown in Tables 11.11 and the corresponding Figures 11.12, D.21 and D.22.

Again, the presented approaches cannot reproduce the behavior of the training data for this data set. Rather, the best ensemble predictions are approximately the mean, as depicted in Figure 11.12. Here it's noteworthy that the phase space trajectory filters provide a better mean prediction than the complexity-based filters. Overall we conclude that the presented approaches cannot predict this data set.

The two best predictions, i.e. the predictions depicted in Figure 11.12, outperformed all baseline predictions from Appendix D.2. The baseline RMSEs are 0.42425 for the LSTM, 0.37911 for the GRU, and 0.38741 for the recurrent neural network.

TABLE 11.11: Best results for the shampoo sales data set.
Left: phase space filter predictions
Right: Signal complexity filter predictions

Filter	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**	**Filter**	**Interpolation Technique**	N_I	**RMSE**	**ΔRMSE**
los2_rand	Not Interpolated	-	0.17218	±0.05066	Hurst	Not Interpolated	-	0.20308	±0.05084
loss_rand	Not Interpolated	-	0.17243	±0.00000	Hurst Shannon	Not Interpolated	-	0.20308	±0.05084
loss_rand	Fractal Interpolated	11	0.18466	±0.00000	Fisher Hurst	Not Interpolated	-	0.20308	±0.05084
los2_rand	Not Interpolated	-	0.18723	±0.06246	SVD Hurst	Not Interpolated	-	0.20308	±0.05084
loss_rand	Not Interpolated	-	0.19235	±0.00000	Lyap Hurst	Linear Interpolated	11	0.20637	±0.01597

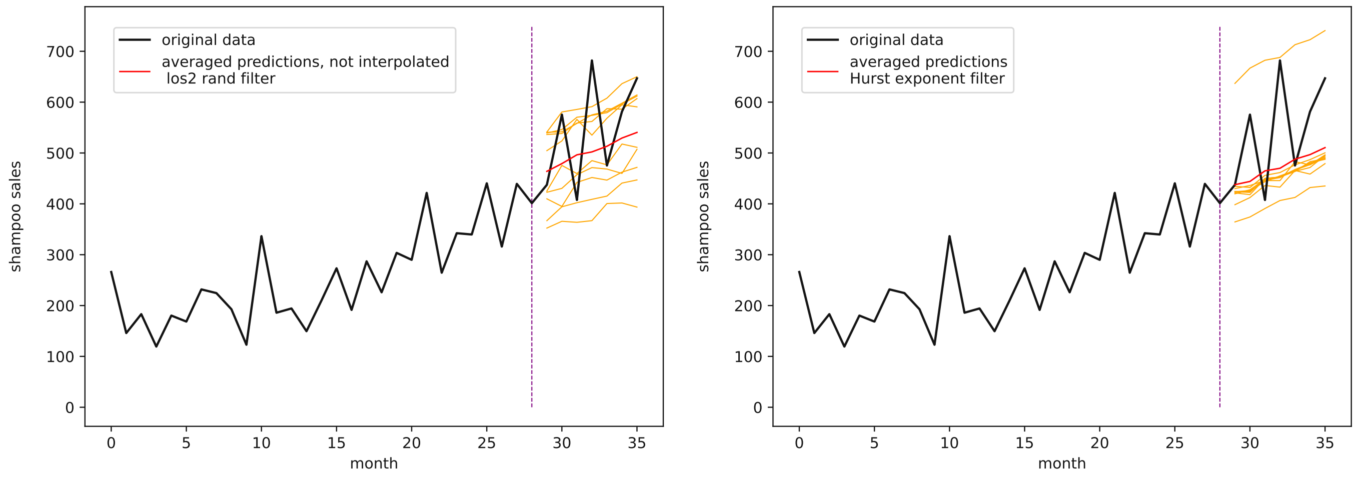


FIGURE 11.12: Best predictions for the shampoo sales data set.
Left: los2_rand-filter, not interpolated, RMSE=0.17218±0.05066
Right: Hurst-filter, not interpolated, RMSE=0.20308±0.05084

11.5 Discussion and Summary

The research presented in this chapter is a continuation of the work presented in Chapter 9. It consists of interpolating data sets, generating a multitude of predictions using randomly parameterized LSTM neural networks, and selecting a few of these predictions using filters based on the phase space properties and/or the signal complexity of the training data.

In addition to Chapter 9, this chapter tests the developed PhaSpaSto interpolation technique from Section 5.2 for its applicability to improve LSTM neural network predictions and, further, the corresponding loss-function, as a prediction filter.

11.5.1 Prediction Filters

When it comes to identifying the best ensemble prediction filters, we see that for all data sets, the signal-complexity-based filters performed best for a total of seven data sets. The filters based on the second derivates along the reconstructed phase space trajectory performed best for four data sets.

From the featured top five results for all data sets, the phase-space-based filters are distributed such that 22 of the results use the los2_rand filter and 33 the loss_rand filter. For all the top results of the signal-complexity-based filters, we find that Fisher/Hurst is featured five times and Lyap/Fisher four times. Still, given the presented results, different filters will perform best for different data sets. Further, just like in Chapter 9, the author recommends a combination of a complexity measure taking into account the phase properties of a time series like Fisher's information and/or SVD entropy and a complexity measure taking into account the stochastic fluctuating nature of the data set, e.g., the Hurst exponent. Additionally, we recommend using the loss_rand filter based on the second derivatives along a reconstructed phase space trajectory.

11.5.2 Interpolation Techniques

We cannot give a clear answer when it comes to identifying the best interpolation technique to improve LSTM neural network time-series predictions. Again, for different data sets, different interpolation techniques performed best. The best results are distributed among interpolated and not interpolated data sets such that 31 of the top results are fractal interpolated ones, 29 are linear interpolated ones, and 28 are PhaSpaSto interpolated ones; in contrast, only 21 are not interpolated results. Thus we recommend using an interpolation technique. Still, the author recommends employing a fractal or the developed stochastic interpolation. These are conceptually more consistent than the presented linear interpolation. This is because considering the phase space or stochastic properties of a time series data is more reasonable than just drawing a straight line between two points. Further, given the results featured in Section 9.4, which recommend the fractal interpolation because of their reduced error deviation, we can reiterate this for the results of this chapter.

11.5.3 Key Findings

Finally, we highlight the key findings of this Chapter:

- The presented ideas can autoregressively predict time series with visible regularities. This is shown for the measles cases in NYC, the monthly international airline passengers, the Canadian lynx, and the sunspots data set.
- Given the results of this chapter, we conclude that the presented approach cannot accurately predict complex short/sparsely sampled data sets, such as the annual maize and wheat yields in Austria. Still, the proposed ideas can provide rough estimates and outperform standard neural network approaches like the featured baseline predictions.
- The presented approach cannot predict the Lorenz system. Given that the randomly parameterized LSTM neural network approach can occasionally reproduce some of the behavior of the Lorenz system, the author concludes that more research must be done to effectively predict chaotic model data with the presented approach.
- The presented approach cannot predict the featured financial data sets effectively. Given the results in Figures 11.9 and 11.10, we conclude that our approach, at it's best, can only predict the mean for financial time series.
- The presented randomly parameterized ensemble predictions can significantly be improved by employing the presented interpolation techniques, discussed in Chapter 5. The author recommends using the developed fractal or attractor-based stochastic interpolation technique to consider the complexity and phase-space properties of the data under study, respectively.
- The prediction filters based on the signal complexity of the data under study outperformed the prediction filters based on the variance of second derivatives along the reconstructed phase space trajectory for only one case. Thus the author concludes that the prediction filters described in Section 9.3 are the best choice.

Reconstructed Phase Space and Neural Network Time Series Predictions

Further, based on the results from Chapters 9 and Section 11.4, the author recommends using a combination of SVD entropy or Fisher's information and the Hurst exponent for filtering random predictions. Still, the prediction filters based on the loss function from Sections 11.3 can effectively filter predictions and improve forecasts up to the point where they can, in some cases, outperform the presented baseline predictions.

III. Discussion and Conclusions

Chapter 12

Benchmark Comparisons

This chapter contains benchmark results from the literature to compare the presented results to state-of-the-art and related results. We present and discuss these benchmark results for eight data sets. We split the comparison between two sources. We first discuss the results for the analyzed sales data sets from the time series data library, [94]. Here we compare our best results to the best results from Ref. [110]. In this case, our results are compared to basic time series forecasts using methods such as the Holt-Winters method, SARIMA, or an LSTM neural network implementation.

Second, we compare another four data sets with the results from Ref. [93]. In this case, we compare our results to state-of-the-art hybrid models.

Our approach can outperform the forecasts for the sales data sets in three out of four cases. Regarding the state-of-the-art hybrid approaches, our best predictions are far off the best results from the literature. We thus conclude that, though the developed methods can outperform basic implementations, they are no match for more advanced state-of-the-art techniques, particularly hybrid ARIMA models.

SARIMA, ARIMA, various hybrid approaches, and everything else we compare our results to are not directly part of this research and only serve as a benchmark comparison. We will not discuss these techniques here. Instead, we refer to the results' sources for further reading and an in-depth discussion.

12.1 Sales Time Series Benchmark Results

In [110] time-series predictions are performed for many data sets from the Time Series Data Library [94]. Part of this are four data sets which are presented in Chapters 9 and 11. All of our best results, the corresponding baseline results, and the benchmark results from the literature are collected in Table 12.1. The results for the Perrin Freres champagne sales data set are from Chapter 9. The results for the shampoo sales data set are from Chapter 11. The results for the remaining two data sets were produced using the second neural network architecture discussed in Section 6.1.3 and a total of 500 different predictions, all featured interpolation techniques with the corresponding interpolation points of $N_I \in \{1, 3, 5, 7, 9, 11, 13, 15, 17\}$ and all discussed filters from Chapters 9 and 11.

The results from the literature, the LSTM, GRU and RNN baseline predictions and the best predictions using the randomly parameterized neural network architecture (Chapters 9 and 11) are shown in Table 12.1. The results are discussed using, as in the other parts of this research, the RMSE (Section 6.3) on rescaled data, i.e. the data is scaled to the interval $[0, 1]$, denoted as **RMSE** $[0, 1]$.

The results show that the here presented approach, i.e., interpolated data sets, randomly parameterized neural networks, and the corresponding prediction filters outperform the results from the literature in three out of four cases, i.e., the monthly car sales in Quebec, the CFE specialty writing paper sales and the shampoo sales data set. Still, the employed GRU baseline prediction achieved the best performance for the shampoo sales data set. In the case of the Perrin Freres champagne sales data set, the results from the literature best our results.

Our presented LSTM baseline predictions outperformed the LSTM benchmark predictions for the monthly car sales in Quebec and the CFE specialty writing paper sales data set, thus showing that the chosen architecture is reasonable for our use case. We conclude the same for the chosen GRU and RNN predictions, as these predictions are within the range of the results from the literature for the featured data sets.

Table 12.1: Benchmark results for the monthly car sales in Quebec, the Perrin Freres champagne sales, the CFE spceialty writing sales and the shampoo sales data set. The baseline predictions are taken from Appendices B.5 and D.2. The results for Holt-Winters SARIMA, LSTM and Prophet are taken from the literature. The bold marked results are the best results combining interpolation techniques randomly parameterized neural networks and prediction filters.

Data	Approach	RMSE [0,1]
Monthly car sales in Quebec	Additive Holt-Winters Method, [110]	0.08143
Monthly car sales in Quebec	Multiplicative Holt-Winters Method, [110]	0.08452
Monthly car sales in Quebec	SARIMA,[110]	0.08832
Monthly car sales in Quebec	LSTM, [110]	0.11472
Monthly car sales in Quebec	Prophet, [110]	0.09348
Monthly car sales in Quebec	Baseline LSTM	0.11269
Monthly car sales in Quebec	Baseline GRU	0.11170
Monthly car sales in Quebec	Baseline RNN	0.11827
Monthly car sales in Quebec	**Linear interpolated, 9 interpolation points, Hurst-Fisher-filter**	**0.07549**
Perrin Freres champagne sales	Additive Holt-Winters Method, [110]	0.03634
Perrin Freres champagne sales	Multiplicative Holt-Winters Method, [110]	0.04694
Perrin Freres champagne sales	SARIMA, [110]	0.02758
Perrin Freres champagne sales	LSTM, [110]	0.04894
Perrin Freres champagne sales	Prophet, [110]	0.05457
Perrin Freres champagne sales	Baseline LSTM	0.06915
Perrin Freres champagne sales	Baseline GRU	0.06506
Perrin Freres champagne sales	Baseline RNN	0.07313
Perrin Freres champagne sales	**Fractal interpolated, 13 interpolation points, Fisher-Hurst-filter**	**0.04968**
CFE specialty writing paper sales	Additive Holt-Winters Method, [110]	0.30322
CFE specialty writing paper sales	Multiplicative Holt-Winters Method, [110]	0.31035
CFE specialty writing paper sales	SARIMA, [110]	0.32729
CFE specialty writing paper sales	LSTM, [110]	0.47528
CFE specialty writing paper sales	Prophet, [110]	0.31567
CFE specialty writing paper sales	Baseline LSTM	0.21541
CFE specialty writing paper sales	Baseline GRU	0.21653
CFE specialty writing paper sales	Baseline RNN	0.21086
CFE specialty writing paper sales	**Stoch. interpolated, 13 interpolation points, Hurst-Fisher-filter**	**0.17404**
Shampoo sales	Additive Holt-Winters Method, [110]	0.22766
Shampoo sales	Multiplicative Holt-Winters Method, [110]	0.26784
Shampoo sales	SARIMA, [110]	0.33741
Shampoo sales	LSTM, [110]	0.20346
Shampoo sales	Prophet, [110]	0.38605
Shampoo sales	Baseline LSTM	0.42424
Shampoo sales	Baseline GRU	0.16190
Shampoo sales	Baseline RNN	0.38741
Shampoo sales	**Not interpolated, los2-filter**	**0.17218**

12.2 State-Of-The-Art Hybrid Model Benchmark Results

In this Section we compare our best results from Chapter 11 to state-of-the-art results from the literature. We take into account all featured results, the corresponding techniques and references from Domingos et. al. [93] and compare our results to them. We provide results and a comparison for the monthly international airline passengers data set, Section 7.1; the Canadian lynx data set, Section 7.10; the USD/GBP exchange rate data set, Section 7.14; and the sunspots data set, Section 7.13.

12.2.1 Monthly International Airline Passengers

The results from Chapter 9 are performed on a 70%/30% split. Thus, we present additional results using the first 80% as training data and the remaining 20% as a test data set to validate our results. For the baseline predictions, we employ the same architecture as in Section B.1 with 20 input nodes. Still, we use varying numbers of hidden layer neurons and epochs. We use 30 hidden layer neurons and 40 training epochs for the LSTM baseline prediction, 26 hidden layer neurons and 44 training epochs for the GRU baseline prediction, and 100 hidden layer neurons and 60 training epochs for the RNN prediction.

These results, the baseline predictions, and the best results combining interpolation techniques, randomly parameterized neural networks, and prediction filters are collected in Table 12.2. We highlight the best result from the literature and our best one with a bold font type for better visibility. The best performing approach is the nonlinear hybrid approach combining ARIMA and MLP, [93]. Our approach did not outperform the best hybrid results from the literature. Still, it did outperform older ARIMA and MLP approaches and even some of the featured hybrid approaches.

TABLE 12.2: Benchmark, baseline, and randomly parameterized neural network results for the monthly international airline passengers data set.

Approach	**RMSE**	**MSE**
ARIMA [111–113]	43.8	1918.6
MLP [111–113]	22.5	507.7
Hybrid ARIMA & ANN [111]	22.0	485.7
Hybrid ARIMA & SVR [114]	19.7	388.9
Hybrid ARIMA-ANN and a moving-average filter[115]	28.2	793.3
Hybrid ARIMA & SVR [116]	20.1	405.4
ETS-ANN [117]	20.0	400.3
ANN [112]	15.9	253.3
Hybrid ARIMA & ANN [113]	16.1	258.8
NoLiC - MLP [118]	16.1	257.9
Nonlinear Hybrid model ARIMA & MLP [93]	**13.0**	**168.5**
Nonlinear Hybrid model ARIMA & SVR [93]	14.6	211.9
Baseline LSTM	30.6	936,4
Baseline GRU	27.6	761,8
Baseline RNN	26.4	967,0
Fractal interpolated, 7 interpolation points, `loss_rand-filter`	**18.8**	**352.5**

12.2.2 Canadian Lynx

For this data set, we take into account our best results from Chapter 11, but in contrast, and in accordance with Domingos et al. [93], we need to transform our data using the logarithm with base ten to make our results comparable. We do the same for the baseline predictions and thus adapt our architecture to the transformation. We use 45 hidden layer neurons and 25 training epochs for the LSTM baseline model, two hidden layer neurons and 500 training epochs for the GRU baseline model, and 180 hidden layer neurons and five training epochs for the RNN model.

All benchmark results, the baseline predictions, and the best results for the randomly parameterized neural networks are collected in Table 12.3. We highlight the best result from the literature and our best one with a bold font type for better visibility. The best result for the Canadian lynx data set is the one from the literature, i.e., the nonlinear hybrid approach combining ARIMA and MLP [93]. Our approach did not outperform any of the listed benchmark results but still did outperform all baseline predictions.

Table 12.3: Benchmark, baseline and our best results for the Canadian lynx data set.

Approach	**RMSE**	**MSE**
ARIMA [111–113]	0.1428	0.0204
MLP [111–113]	0.1428	0.0204
Hybrid ARIMA & ANN [111]	0.1311	0.0172
Hybrid ARIMA & SVR [114]	0.1428	0.0204
Hybrid ARIMA-ANN and a moving-average filter[115]	0.1367	0.0187
Hybrid ARIMA & SVR [116]	0.1200	0.0144
ETS-ANN [117]	0.1715	0.0294
ANN [112]	0.1166	0.0136
Hybrid ARIMA & ANN [113]	0.0995	0.0099
NoLiC - MLP [118]	0.1229	0.0151
Nonlinear Hybrid model ARIMA & MLP [93]	**0.0854**	**0.0073**
Nonlinear Hybrid model ARIMA & SVR [93]	0.1005	0.0101
Baseline LSTM	0.4019	0.1615
Baseline GRU	0.3478	0.1210
Baseline RNN	0.3215	0.1034
Linear interpolated, 11 interpolation points, `lyap`-filter	**0.2746**	**0.0754**

12.2.3 USD/GBP Exchange Rate

For this data set, we take into account our best results from Chapter 11. In accordance with Domingos et al. [93], we need to transform our data using the natural logarithm to make our results comparable. We do the same for the baseline predictions and thus must adapt our models to the transformation. The new LSTM baseline model uses 35 hidden layer neurons and 60 training epochs. The GRU baseline model uses three hidden layer neurons and 500 training epochs. The RNN model uses 180 hidden layer neurons and five training epochs.

The benchmark results, the baseline predictions, and the best results for the randomly parameterized neural networks approach are collected in Table 12.4. We highlight the best result from the literature and our best one with a bold font type for better visibility. The best model for the USD/GBP exchange rate data set is the nonlinear hybrid approach combining ARIMA and MLP [93]. Our approach and the baseline predictions are far off the listed benchmark results. Still, the randomly parameterized neural networks outperform our baseline predictions.

TABLE 12.4: Benchmark, baseline and best results for the USD/GBP exchange data set.

Approach	**RMSE**	**MSE**$\cdot 10^5$
ARIMA [111–113]	0.0067	4.5297
MLP [111–113]	0.0067	4.5265
Hybrid ARIMA & ANN [111]	0.0066	4.3590
Hybrid ARIMA & SVR [114]	0.0060	3.5183
Hybrid ARIMA-ANN and a moving-average filter[115]	0.0061	3.7285
Hybrid ARIMA & SVR [116]	0.0060	3.5944
ETS-ANN [117]	0.0059	3.5313
ANN [112]	0.0061	3.7639
Hybrid ARIMA & ANN [113]	0.0060	3.6477
NoLiC - MLP [118]	0.0057	3.2641
Nonlinear Hybrid model ARIMA & MLP [93]	**0.0056**	**3.1904**
Nonlinear Hybrid model ARIMA & SVR [93]	0.0058	3.3783
Baseline LSTM	0.3048	9290
Baseline GRU	0.1741	3030
Baseline RNN	0.1291	1670
Stoch. interpolated, 11 interpolation points, `loss_rand-filter`	**0.0582**	**338.7243**

12.2.4 Sunspots

For this data set, we take into account our best results from Chapter 11.

The benchmark results, the baseline predictions, and the best results combining interpolation techniques, randomly parameterized neural networks, and prediction filters are collected in Table 12.5. We highlight the best result from the literature and our best one with a bold font type. The best result for the sunspots data set is obtained using the nonlinear hybrid approach combining ARIMA and MLP [93]. Our approach did not outperform any of the listed benchmark results.

The sunspots data set shows strong seasonality and regularities. Our approach does perform well on similar data sets. Still, our approach is far off from any of the featured benchmark results. And our best result is the GRU baseline prediction.

Table 12.5: Benchmark, baseline and best results for the sunspots data set.

Approach	**RMSE**	**MSE**
ARIMA [111–113]	17.4	306.0
MLP [111–113]	18.7	351.1
Hybrid ARIMA & ANN [111]	16.7	280.1
Hybrid ARIMA & SVR [114]	17.5	306.8
Hybrid ARIMA-ANN and a moving-average filter[115]	17.3	300.4
Hybrid ARIMA & SVR [116]	17.3	300.4
ETS-ANN [117]	17.7	312.0
ANN [112]	25.3	234.2
Hybrid ARIMA & ANN [113]	14.8	218.6
NoLiC - MLP [118]	17.6	308.8
Nonlinear Hybrid model ARIMA & MLP [93]	**14.9**	**222.4**
Nonlinear Hybrid model ARIMA & SVR [93]	14.6	213.4
Baseline LSTM	68.3	4661.9
Baseline GRU	51.5	2863.6
Baseline RNN	80.1	6416.9
Linear interpolated, 15 interpolation points, Lyap-filter	**68.2**	**6178,0**

Chapter 13

Summary and Discussion

This thesis presents concepts combining measures of signal complexity and reconstructed phase spaces with neural networks to improve autoregressive time series predictions. Though we used LSTM neural networks, the presented ideas apply to any algorithm capable of predicting time series data. One can always interpolate the data under study and filter differently parameterized model predictions.

We tested to what extent one can use interpolation techniques to improve neural networks time series predictions to deal with short and/or sparsely sampled time series data. We developed a fractal interpolation method based on the ideas from the *fractal curve fitting* technique, which was developed by Manousopoulos et al. [27]. However, we incorporated knowledge about the local scaling behavior into the fractal interpolation. Further, we developed an interpolation technique that considers the reconstructed phase space of a time series and provides an interpolation that guarantees a smooth phase space trajectory.

We further developed an autoregressive prediction approach based on randomly parameterized neural networks. I.e., we do not parameterize a neural network to achieve the optimal performance for a single data set. Instead, we produce many autoregressive predictions using randomly parameterized neural networks and filter the predictions based on their signal complexity and phase space properties.

To develop and test these ideas, we performed four experiments which are discussed in Chapters 8, 9, 10 and 11. Prediction results from these chapters are compared to state-of-the-art time series predictions and comparable results from the literature in Chapter 12.

The first experiment, described in Chapter 8, tests the applicability of the previously mentioned fractal interpolation and linear interpolation to improve LSTM neural network time series predictions. The results show that both interpolation techniques can increase the accuracy of the employed neural networks. Here, the fractal interpolation performs slightly better on unknown data, whereas the linear interpolation provides better results on the training data. This experiment is done for a single number of interpolation points. We further analyze the original and the interpolated data's signal complexities. We find that both the Hurst exponent and the fractal dimension suggest that interpolated data sets can be predicted with increased accuracy. In contrast, the spectrum of Lyapunov exponents indicates the opposite.

Chapter 9 contains the results of the second experiment. This chapter introduces randomly parameterized neural networks and filters based on measures of signal complexity. These filters are used to improve autoregressive ensemble predictions by discarding "bad" forecasts from the ensemble. I.e., we keep only predictions with a signal complexity close to the complexity of the training data. We perform the prediction experiments for non-interpolated, linear interpolated, and fractal interpolated time series data and five different data sets. The errors of final forecasts are evaluated only for the actual data points and the corresponding predicted data points. The results show that both the fractal and the linear interpolation increase the accuracy of the prediction. Further, the employed prediction filters can drastically improve these ensemble predictions. We also analyzed the data's complexity properties using five measures of signal complexity. Again, an increasing Hurst exponent and a decreasing entropy suggest increased predictability for interpolated time series data, whereas the largest Lyapunov exponent indicates the opposite.

Chapter 10 discusses the applicability of the developed PhaSpaSto interpolation. We test PhaSpaSto interpolation against linear interpolation, multi-point fractional Brownian bridges, and spline interpolation for its ability to reconstruct missing data points. The results show that cubic spline interpolation performs best for the Lorenz system, whereas PhaSpaSto interpolation performs best for real-life data sets that show oscillatory behavior. However, for more random data sets, PhaSpaSto interpolation does not perform well. We also present interpretable phase space portraits that depict the increased smoothness of the reconstructed phase space trajectory when applying PhaSpaSto interpolation.

Chapter 11 is the final experiment on the applicability of the developed methods. Here we are using fractal, linear, and PhaSpaSto interpolated data sets, prediction filters based on measures of signal complexity, and prediction filters based on the smoothness of a reconstructed phase space trajectory. The results show that all tested interpolation techniques do increase the accuracy of the neural network time series predictions. Also, depending on the time series data, different filters perform best. The results also show that the developed approaches do not work for the Lorenz system and data sets that behave randomly, e.g., financial or stock market data.

The benchmark comparisons discussed in Chapter 12 show that our best results can outperform basic forecast implementations on data sets that show regularities and seasonality. However, our results are far off for more random data sets compared to state-of-the-art hybrid models.

13.1 Interpolation Methods and Neural Networks Time Series Predictions

One aspect of this research is to improve time series prediction for sparsely sampled data sets. For this reason, we employed a total of three different interpolation techniques. We consider one of them an overall new development, i.e., the attractor-based stochastic interpolation presented in Section 5.2, which was named PhaSpaSto interpolation for simplicity. The other interpolation is a fractal interpolation, which is described in Section 5.1. We used the Hurst exponent to find the vertical scaling factors for this method. The third employed technique is a basic linear interpolation, [101].

Our experiments demonstrated that all the employed interpolation techniques enhance the performance of LSTM neural networks for time series predictions. This finding is consistent with the work of Semenoglou et al. [26]. I.e., we observe that our applied interpolation techniques can improve the predictability of a given time series. In contrast, Semenoglou et al. focused on data augmentation in general and showed that all employed methods can enhance the predictability of a dataset. The individual differences are bound to varying data sets. Recalling our best results for the benchmark comparison in the previous chapter, each of the three interpolation techniques is used once for three of the four best results for the sales data sets, Section 12.1. Chapters 9 and 11 indicate the same; the choice of the employed interpolation technique seems arbitrary, but one has to use an interpolation technique to improve the forecasts. However, we must mention that a non-interpolated data set occasionally provides the best forecast. This arbitrariness regarding the choice of interpolation techniques is partially caused by the increase in persistency when interpolating a data set, which means that with increased persistency, i.e., an increased Hurst exponent and reduced fractal dimension, two consecutive data points are closer to each other, i.e., more fine-grained. Thus a prediction is never too far off, and consequently, a robust algorithm/model is better capable of reproducing the learned behavior autoregressively. This is also discussed in Section 8.4.

We performed our experiments with varying numbers of interpolation points. We used only odd numbers of interpolation points for our prediction experiments as the obtained and then altered fractal curve fitting implementation works with odd numbers of interpolation points only[1]. Also, as the best results from Chapter 9 show, we cannot find a trend or an indicator for the number of interpolation points for different data sets. Chapter 10 indicates that PhaSpaSto interpolation requires a certain amount of interpolation points to provide a smoothed-out phase space trajectory. Thus we chose the numbers of interpolation points for the final prediction experiment in Chapter 11 to be $N_I = \{9, 11, 13, 15\}$. We had to reduce the number of interpolation points because of the increased number of different data sets and limited computational resources. Further, as Chapter 9 suggests, the number of interpolation points is rather arbitrary for the randomly parameterized neural networks, and the best results, differing in their number of interpolation points and the type of interpolation, show only minor deviations in the errors for the best settings. Further, we want to point out that, as far as the author knows, there's no exhaustive comparative analysis on neural networks and time series interpolation. As this wouldn't be within the scope of this thesis, we didn't aim to provide a complete analysis of the topic. However, we need to mention again the work by Semenoglou [26], which deals with a similar topic, i.e., how data augmentation can improve the predictability of univariate time series data using neural networks. The researchers present a variety of data sets and different methods used to increase the amount of data for univariate time series data, e.g., interpolation similar to the work presented here. One of the key findings is that the accuracy improvements provided by data augmentation techniques decrease with the amount of data in the initial data set. Roughly speaking, long data sets do not need to be augmented and/or interpolated. We see a tendency towards this in some of our results, i.e., some of the best results for the NYC measles outbreaks data set (Section 11.4.4), which is the second longest data set discussed here, are found for the non-interpolated data set. We can only guess the optimal number of interpolation points for a given data set or determine it by trial and error. Still, for PhaSpaSto interpolation, we recommend ten interpolation points for data sets with less than 100 data points to provide a smooth phase space portrait. Of course, this can vary depending on the data set under study, but it should provide a first guess on where to start.

[1]In the newest implementation of the discussed fractal interpolation, [79], we provide an updated version of this fractal interpolation where one can choose arbitrary numbers of interpolation points. This updated version was not available when the author performed all the experiments.

13.2 Measures of Signal Complexity

Throughout our experiments, we employed several measures of signal complexity as additional criteria for analysis and prediction filters.

When indicating increased predictability, the employed measures of signal complexity are contradictory to our results. Thus we discuss all employed measures of signal complexity with respect to our prediction results.

The Hurst exponent and the fractal dimension are similar concepts, such that they can be linked using $d_F - 1 = H$, [77]. Here d_F is the fractal dimension, and H is the Hurst exponent. The following discussion is on the Hurst exponent, but it approximately holds for the fractal dimension as well. This connection is also discussed in Section 8.1.

One can use both concepts to analyze the fluctuations of time series. A Hurst exponent close to one indicates a very persistent, i.e., straight behavior. In contrast, a Hurst exponent close to zero indicates anti-persistent behavior, i.e., a signal tends to change direction after each data point. A Hurst exponent close to 0.5 indicates random behavior, i.e., there is only a $50 : 50$ chance for consecutive data points to be in the same direction. When analyzing the complexity of the employed data sets in Chapters 8 and 9, the Hurst exponent indicates increased persistency and predictability for interpolated data. I.e., the persistency increases with the number of interpolation points. This is true for short-term predictions and linear interpolated time series data. A linear interpolated time series with five additional data points can be predicted five steps ahead after each original and first interpolated data point. The first two data points give the slope. Consequently, the following five data points are in a straight line to the following original data point. The same is approximately true for spline-interpolated data. We would need three points to estimate the resulting curve for a quadratic polynomial, four for a cubic polynomial, and so on. Given the results from Section 10.1.2, we conclude that the same is approximately true for PhaSpaSto interpolation. This assumption is based on the observation that PhaSpaSto interpolation performs similarly to the employed cubic spline interpolation for many data sets.

However, increased predictability is not obvious for the fractal interpolated data. Still, the employed fractal interpolation connects two consecutive points with data that, in the end, is somewhat persistent. I.e., the fluctuations of subintervals are smaller than the overall fluctuations of the data set. This is inherent in choosing the vertical scaling factors for the fractal interpolation. We want the Hurst exponents of subintervals to coincide with the Hurst exponent of the interpolated data. When we reduce the observed time interval, the fluctuations also need to become smaller, similar to zooming in on a fractal. As we observed increased predictability, i.e., lower errors, for most interpolated and filtered predictions, we conclude that the Hurst exponent does indicate increased predictability. However, we do not observe a drastic increase or approximately functional dependence on the number of interpolation as is depicted in Figures 9.2, B.1, B.3, B.5 and B.7.

Given that results that are filtered using the Hurst exponent are featured among the best results in Chapters 9 and 11, we conclude that the Hurst exponent can be used to characterize time series data such that it can be used as an ensemble filter.

We further need to mention that instead of the Hurst exponent one can choose detrended fluctuation analysis [119], or wavelet-based methods [120], to measure the long-term memory of times series data.

The complexity analysis in Chapter 9 shows that Fisher's information and SVD entropy depict the same information for time series data. We observe increasing information and decreasing entropy for growing numbers of interpolation points. This is because both measures use a single value decomposition and two parameters, i.e., d_E and τ, which we defined to be the data's phase space embedding. Thus both of these tools, to a certain extent, measure the density of the data's phase space embedding. This density increases with growing numbers of interpolated data points. Therefore, we conclude that both concepts can indicate increased predictability, similar to the Hurst exponent. Still, we cannot find this almost functional dependence on the number of additional interpolation points.

Both SVD entropy and Fisher's information are featured among the best results in Chapters 9 and 11. Thus we conclude that both SVD entropy and Fisher's information can characterize time series such that we can use them to filter ensemble predictions.

We used the spectrum of Lyapunov exponents for analysis in Chapter 8 and the largest Lyapunov exponent for analysis in Chapter 9. In both chapters, the largest Lyapunov exponent, i.e., the first of the spectrum, indicates that linear interpolated data sets behave more chaotic with increasing numbers of interpolation points. This effect is diminished for the fractal interpolated data set as it appears to be only slightly more chaotic than the non-interpolated data set. We interpret a more chaotic behavior with reduced predictability, which we didn't observe. In general, interpolation improves our forecasts. Thus, we conclude that the spectrum of Lyapunov exponents does not indicate the predictability of neural networks for non-model data sets. Still, prediction filters based on the spectrum of Lyapunov exponents are frequently found among the best results in Chapters 9 and 11. Thus we conclude that the spectrum of Lyapunov exponents is capable of characterizing time series such that we can use it as an ensemble filter.

We use Shannon's entropy for analysis in Chapter 9. Shannon's entropy shows that the fractal and linear interpolated data sets behave more unpredictably with increasing numbers of interpolation points. I.e., the entropy increases, which is not what we observe as interpolated data sets tend to be more predictable than non-interpolated ones. Thus we conclude that Shannon's entropy does not depict this behavior. This increase, though, mainly results from the increase in data points. Because we used the plain version of Shannon's entropy, i.e., no binning involved, the probabilities for each occurring event become less with increasing signal length, and thus Shannon's entropy increases. This is also shown in Appendix E.

Still, Shannon's entropy, as a filter, is featured among the best filtered forecasts in Chapters 9 and 11. The main reason Shannon's entropy in its non-continuum adapted version works well for specific data sets is that it can discard very regular signals with reoccurring values, shown in Appendix E. However, as also shown in Appendix E, it does not differentiate between fractional Brownian motions of different Hurst exponents or the Lorenz system, as these data sets feature continuous data and do, per se, not provide recurring values. This partially explains why Shannon's entropy can give good results in combination with other prediction filters. I.e., Shannon's entropy discards periodic and very constant forecasts, and another filter, which might get the periodic and too-regular behavior wrong because it looks for small lengths like, e.g., a specific Hurst exponent, will be presented only forecasts that behave in a non-recurring manner, thus closer to what one would expect from real-life or continuous data. I.e., the randomly parameterized neural networks produce very regular or even constant signals for certain data sets. These predictions, in most cases, do not depict relevant information about the original signal. These forecasts are discarded by the filter based on Shannon's entropy, and this consequently results in an improved prediction.

We employed the variance of second derivatives along a reconstructed phase space trajectory for finding smooth phase space interpolations in Chapter 10. We also used this idea to improve ensemble forecasts in Chapter 11. The ensemble results in Section 11.4 show that the variance of second derivatives can be used to filter ensemble predictions and thus drastically reduce the errors compared to non-interpolated data sets. Though this variance has not been used as a measure for predictability or complexity, we suggest further exploring this in future research. The author guesses that only very similar data sets can be analyzed using this tool, as it only yields relative values. We want to give two examples of where one might use this tool. First, one might use this tool for different excerpts from a long record of financial data and further relate the variance of second derivatives to other variables and or/predictability. Second, one might use this tool to compare different records of environmental or agricultural systems to, e.g., give estimates on resilience. Additionally, as demonstrated in Appendix E, various phase space embeddings lead to subtle changes in the capability of the variance of second derivatives to differentiate between distinct signals. Consequently, we recommend future research to select the phase space embedding for the variance of second derivatives in a way that ensures optimal separation. At present, we do not know the precise method for choosing the variance of second derivatives to achieve this optimal separation. However, considering that separability varies across different signals, we anticipate that it is possible to identify an ideal phase space embedding to accomplish this goal.

13.3 Computational Resources

The developed randomly parameterized neural networks approach is expensive. This is due to the multitude of predictions and, consequently, many generated neural networks.

We compare the yield data sets to the USD/GBP exchange time series to estimate how expensive the approach is. We calculated all neural network results on the TU Wien GPU cluster. We performed all calculations on an Nvidia Gforce RTX 2080 GPU. It takes approximately half a day to make 500 predictions for one run of the yield data sets. I.e., predicting the original, the linear, the fractal, and the PhaSpaSto interpolated time series data with a fixed number of interpolation points. In contrast, it takes roughly five days to do the same for the USD/GBP exchange time series. Both time frames depend slightly on the varying numbers of interpolation points. The yield data sets have 56 and 57 original data points, whereas the USD/GBP exchange data set has 731 data points. It took approximately 80 days to calculate all results from Chapter 11 on a single GPU.

So, in the end, one can circumvent the problem of parameterizing neural networks by using this very costly approach. Still, after generating many forecasts, one must choose the best prediction filter for each problem.

13.4 Issues

We further need to discuss the problems, flaws and issues of the presented work.

Regarding the Applicability of Data Augmentation for Time Series Data: First, interpolating annual yields and similar data sets does not provide meaningful interpolated data points because a yearly yield can only be observed once a year. Still, as discussed in [26], we often encounter univariate real-life time series data with low amounts of data. Data augmentation helps in forecasting such data sets as data augmentation techniques can increase the overall training data required to train machine and deep learning algorithms. Other augmentation techniques such as time series combinations, random noise injections, bootstrapping, and upsampling make more sense for annual yield data sets, as they do not assume in-between time steps for yearly yields. Still, to keep this thesis focused, we did not employ these techniques as we only deal with univariate time series interpolation as the employed data augmentation technique. The outcome of the experiments where we used interpolation techniques is similar to the work presented in [26], as interpolation increases our predictions' accuracy compared to the presented baseline, non-interpolated and/or unfiltered results for the two annual yield data sets. To avoid confusing these results, we need to mention that, though we trained the employed neural network ensembles on the interpolated data, we evaluated the predictions only on actual data points, e.g., the future annual yields.

In [121], the researchers argue that data augmentation, and consequently generating synthetic data, is common in computer vision but less common when dealing with time series data, e.g., time series classification. This is because time series data is particularly vulnerable to data transformations, e.g., to distort specific data points or introduce noise into a time series. Thus, the researchers are employing interpolation techniques to increase the overall amount of data and show that this is beneficial for deep learning time series classification, e.g., when dealing with simulated, ECG, and/or sensor data time series data.

Thus, though it is inherently flawed and meaningless to interpolate annual yield and similar data sets, data augmentation improves the accuracy of time series prediction in our examples. Further, it should be tested if this can be extended to time series classification for such naturally annually occurring observations, whatever the experimental design might be. We also want to mention that one can use Generative Adversarial Networks (GANs) for time series data augmentation [122]. In the author's opinion, these GAN-based techniques are promising to increase the amount of data for, e.g., annual yield data sets, such that these methods can come up with additional data based on, e.g., other annual yield series.

Regarding the Applicability of Measures of Signal Complexity for Short Time Series Data: Second, we utilized measures of signal complexity throughout this work for data sets of arbitrary length. However, this approach leads to challenges in terms of interpretability and applicability. To numerically demonstrate these issues, we included Appendix E. In this Appendix, we present the results of a range of experiments, including all employed measures of signal complexity and five different signals of varying lengths. This means that we analyze the result of the employed measures of signal complexity depending on the length of the input data. Our experiments from Appendix E indicate that all signal complexity measures suffer from significant errors and a strong bias for short signals. Consequently, the values of these measures depend on the signal's length and become unreliable for shorter signals. This issue is problematic as assigning meaning to a Hurst exponent calculated for a data set with approximately less than 1,000 data points is difficult. Note that this number is a rough estimate based on our results from Appendix E. At this stage, we cannot specify a reasonable amount of data required for estimating the Hurst exponent. Our findings also reveal that larger Hurst exponents tend to be underestimated, and lower Hurst exponents tend to be overestimated for fractional Brownian motions, resulting in a deviation from theoretical predictions. However, more data is always better for all employed signal complexity measures. Based on our results, we assume that approximately 5,000 data points will provide reliable values for all employed measures of signal complexity when comparing and interpreting different data sets. Despite comparing different complexities throughout this work, each interpretation is subject to significant bias due to small data sets. Therefore, we emphasize that, within the developed techniques, we used these measures of signal complexity in a relative manner rather than relying on their absolute interpretability.

Nevertheless, apart from the interpretability and theoretical correctness of the Hurst exponent and the employed measures of signal complexity in general, they can still enhance our ensemble forecasts as filters. One evident reason is that our randomly parameterized neural network predictions generate various forecasts that significantly deviate from the original signal, often producing a mean or very smooth periodic signals. To some extent, the employed measures of signal complexity can differentiate between a straight line, a cosine, a segment from the Lorenz system, or a fractional Brownian motion. However, distinguishing between fractional Brownian motions with different Hurst exponents of short length remains challenging for all employed measures of signal complexity, as discussed in Appendix E. Considering that measures of signal complexity perform better with increased amounts of data, it becomes clear why filtered interpolated data sets yield better predictions rather than non-interpolated ones: interpolation expands the length of the data under study, thus making the various predictions more distinguishable when employing a prediction filter.

Additionally, it is worth mentioning that further research on this topic is needed. One way to enhance the filtering process would be to reduce the inherent bias for short signals within the algorithm used to estimate a signal's complexity.

Regarding the Rather Minor Influence of Varying Phase Space Embeddings: Third, PhaSpaSto interpolation provides similar results for different phase space embeddings. To illustrate this, we present a numerical experiment in Appendix C.4, showing that PhaSpaSto interpolation provides very similar results for the Lorenz system for varying phase space embeddings. Further, the results are close to a cubic spline interpolation for the Lorenz system. Thus, roughly speaking, in this use case PhaSpaSto interpolation is just a very expensive way to do a spline interpolation, which means that the chosen phase space embedding does not matter for most problems. However, when observing the errors on known data points for PhaSpaSto interpolation, we see increasing errors for an increase in the embedding dimension. Further, we also observe a decrease in SVD entropy for an increase in the embedding dimension. Thus we cannot exclude that one might come up with a problem where the phase space embedding for PhaSpaSto interpolation makes a significant difference.

Regarding the Evaluation of the Presented Prediction Experiments: Fourth, we employed two approaches throughout this work to assess our results, specifically, to investigate how interpolation can improve neural network time series predictions on interpolated data. For the experiment from Chapter 8, we interpolated data sets, trained a neural network on them, and evaluated the predictions for both the actual and the supplementary data points generated by the interpolation. Further, in this preliminary experiment, detailed in Chapter 8, we used test fits to assess our prediction outcomes, meaning that we did not predict the full range of unknown data points but instead had the neural network predict only the next data point using the original or interpolated data sets. This experiment revealed that different interpolations contribute to varying degrees of predictability for neural networks in time series data. However, these results appear leveled, as later experiments showed that no single interpolation technique is optimal for all data sets. Instead, different data sets require distinct interpolation techniques for optimal performance and, consequently, various data augmentation techniques for ideal results. In the subsequent chapters, we evaluated all prediction experiments exclusively on actual data points without considering additional interpolated points. Moreover, we assessed the remaining experiments solely for a comprehensive autoregressive prediction of all unknown data.

13.5 Key-Findings

To further sum up this thesis we narrow down everything discussed above to the following three key findings:

1. **The variance of second derivatives along a phase space trajectory can be used to interpolate univariate time series data and produces a smoother reconstructed phase space reconstruction.**

This technique is described in Section 5.2 and it's applicability is shown in Chapters 10, 11.However it's dependence on the actual chosen phase space embedding is only minor, as discussed in Appendix B.

2. **Neural network time series predictions of short or sparsely sampled data can be improved by employing linear, fractal, and PhaSpaSto interpolation.**

   All interpolation techniques are described in Chapter 5 and their applicability to neural network time series predictions is shown in Chapters 8,9 and 11.

3. **Randomly parameterized long short term memory neural networks in combination with prediction filters, based on reconstructed phase space properties or measures of signal complexity, can be used to predict univariate time series data autoregressively.**

   The randomly parameterized LSTM neural network approach is described in Section 6.1.3 and it's applicability is shown in Chapters Chapters 9 and 11.

Chapter 14

Conclusion

This chapter brings the research findings to a close by, firstly, addressing the research question; secondly, outlining the author's contributions; thirdly, discussing the limitations and drawbacks of the developed approaches; and finally, offering suggestions for future research and related problems.

14.1 Answering the Research Question

In order to answer the research question and provide a straightforward solution to the stated problem, we first recall the research question and then dissect it into two parts:

> **To what extent can measures of signal complexity, the entropy of time series data, and the concept of reconstructed phase spaces be used to improve machine and deep learning predictions for univariate and short/sparsely sampled time series data?**

First, to what extent can the actual prediction of time series data, i.e. an autoregressive prediction, be improved by using measures of signal complexity and the concept of reconstructed phase spaces?

Here the answer is given by the developed randomly parameterized LSTM neural network approach, see Section 6.1.3. We first produce a multitude of different autoregressive predictions and then filter them based on their phase space (Section 11.3) or complexity properties (Section 9.3). These ideas outperform the presented baseline predictions and the corresponding unfiltered predictions. The results are shown and discussed in Chapters 9 and 11.

Second, how can be dealt with sparsely sampled data sets? Here the answer is given by the employed and developed interpolation techniques (Chapter 5). I.e., a linear interpolation, the developed fractal interpolation, and the developed PhaSpaSto interpolation technique. Here, interpolating the data sets drastically improved the accuracy of the predictions, Chapters 8, 9 and 11.

14.2 Author's Contributions

To fully comprehend the author's contributions and novel ideas within the presented research, we have compiled a list that outlines the key concepts and the extent to which the author has introduced new insights and approaches.

- **Fractal Interpolation to Improve Neural Network Time Series Forecasts:**

  The author used the ideas from the work of Manousopolous et al. [27], i.e., to use the discussed iterated functions system and adjust it using the vertical scaling factors to interpolate time series data. Here the new contribution is to set these vertical scaling factors such that it suffices the local scaling behavior of a time series by taking into account the local Hurst exponent of a time series data, i.e., choosing the vertical scaling factors such that the deviation in the local scaling behavior is minimal. This technique is discussed in Section 5.1.

  Further, the author provided evidence that this interpolation technique can improve a neural network time series forecast, which is discussed and shown in Chapters 8, 9 and 11.

- **PhaSpaSto Interpolation to Improve Neural Network Time Series Forecasts**

  The author developed an interpolation technique based on multi-point fractional Brownian bridges, developed by Friedrich et al. [29]. However, the author developed a way to choose the smoothest of a population of multi-point fractional Brownian bridges in reconstructed phase space by using a genetic algorithm and the variance of second derivatives for optimization. This technique is described in Section 5.2.

  Further, the author showed that this interpolation technique can improve neural network time series predictions, which is shown and discussed in Chapter 11.

- **Randomly Parameterized Autoregressive Neural Network Time Series Predictions and the Corresponding Prediction Filters:**

The author developed an idea to use LSTM neural networks as an ensemble where each neural network was randomly parameterized and afterward forced to produce an autoregressive prediction. These predictions are then filtered using a variety of complexity metrics and the variance of second derivatives to optimize each of these predictions. The corresponding prediction experiments are described in Chapters 9 and 11.

14.3 Drawbacks and Loose Ends

Concluding the presented research with all featured approaches on combining machine learning, chaos theory, and measures of signal complexity for time series analysis, we give a list of several drawbacks and loose ends that popped up during this research:

- **We actually don't know which interpolation technique is best suited to improve neural network time series predictions.**

  Like choosing the best machine learning algorithm, it mostly depends on the data set under study. Thus the author concludes that this might be related to the *No free lunch theorem*, [123]. The same is true for the number of interpolation points.

  Here the we want to mention that the fractal interpolation from Chapters 8, 9 and 11 produced slightly reduced ensemble errors. Given that, we expect that the presented fractal interpolation and the multipoint fractional Brownian bridges, [29], might be used to increase the robustness of autoregressive neural network time-series predictions. One idea to do and test this assumption would be to generate several slightly differently interpolated time series from one-time series and train a neural network with these data sets.

  Also, as discussed in [102] and researched by the author, there is no exhaustive comparative analysis for interpolation techniques to this day. Further, there is no exhaustive research on interpolation techniques and machine learning for time series data. Thus many aspects of the connections between interpolation techniques and machine learning and/or neural network time series predictions are not discussed. Also, this thesis focuses on combining ideas from chaos theory, nonlinear dynamics, and measures of signal complexity with machine learning/neural networks. We believe that it is evident from our results that one can interpolate time series data using ideas from nonlinear dynamics and complexity. And consequently, one can use the discussed interpolation techniques to enhance neural network time series predictions.

- **We cannot precisely pin down which prediction filter is the best.**

Though many prediction filters performed well, based on the presented results, the author guesses that combining two complexity measures is best. One complexity measure should consider the fluctuations and self-affinity of the data set, e.g., the Hurst exponent. The other should consider some of its phase space properties, such as, e.g., Fisher's information or the SVD entropy, by using a single value decomposition based on the reconstructed phase space of the data under study. Another option, taking into into account the phase space properties of a time series, would be to use the variance of second derivatives along a reconstructed phase space trajectory.

- **We cannot predict the Lorenz system.**

  Though the stochastic interpolation results for the Lorenz system (Section 10.1.1) suggest that the right behavior might be found using the developed prediction filters in Section 11.4, the results are still far off. Other research suggests that chaotic data sets can to some degree be predicted using neural networks [124], so the author concludes that the presented approach is best suited for small/sparsely sampled data sets. Here the author's guess is, in order to make the presented ideas applicable to chaotic data, further research on the varying architectures of randomly parameterized neural networks has to be done.

14.4 Future Research

Finally, given the presented research and the initially mentioned remarks, the author gives a non-exhaustive list of possible related future research:

- **Research on the Variety of Data a Neural Network can (Re-)Produce and the Robustness of Autoregressive Predictions**

  It would be fascinating to explore the wide variety of time series data that neural networks can generate from a trained dataset, i.e. similar to the proposed randomly parameterized neural networks from Section 6.1.3. For instance, one could deconstruct the data into different wavelets or search for time series data in the space of signal complexity that may be inaccessible to neural networks. The author believes that this might include fractional Brownian motions, which could be characterized by a combination of a scaling metric, such as the Hurst exponent, a metric based on singular value decomposition like SVD entropy and/or Fisher's information, or a smoothness criterion such as the variance of second derivatives. However, in the author's opinion, identifying such a class of time series data would require more than one complexity metric to achieve a state space diverse enough to characterize any type of time series data through clustering.

Moreover, it would be valuable to analyze the robustness of autoregressive neural network predictions to understand how small deviations can lead to entirely different behavior for an already trained neural network.

- **Application to Sparse Agricultural or Environmental Data:**

  The concepts introduced in this thesis are applicable to any sparsely sampled univariate non-model time series data. Consequently, when applied to agricultural or environmental datasets, these methods have the potential to generate meaningful predictions in cases where conventional neural network approaches may struggle to capture the data's inherent patterns. Notably, this includes annually or monthly measured observables in these domains.

- **Expanding Attractor-Based Stochastic Interpolation to Multivariate Data:**

  The attractor-preserving interpolation method described in Section 5.2, also known as PhaSpaSto interpolation, can be adapted for multivariate time series data interpolation. To achieve this, considerations must be made regarding the time delay, embedding dimension, and multivariate embedding to ensure compatibility with the proposed interpolation technique. Since the variance of second derivatives is utilized in the reconstructed phase space of the Lorenz system, it should, in theory, be applicable to the actual phase space as well. Nevertheless, further exploration and discussion on this topic are necessary in future research endeavors.

- **Applicability to Diverse Randomly Parameterized Neural Network Architectures:**

  In principle, the ideas presented in this thesis can be adapted to accommodate various neural network architectures or cell types. Examples of such architectures include gated recurrent units [89], simple recurrent neural networks, time-delayed neural networks [18], or even an ensemble consisting of multiple neural network types.

- **A Comprehensive Survey on Time Series Interpolation Techniques, Their Impact on Signal Complexity, and Their Potential to Enhance Machine Learning-Based Prediction Approaches:**

  Despite the lack of a comprehensive comparative analysis for interpolation techniques, as pointed out in [102], we suggest incorporating the complexity analysis of varying numbers of interpolation points, as demonstrated in Section 9.1, into the examination of a broad spectrum of interpolation methods. Furthermore, building on the work presented in [26], we encourage future research to investigate the effects of employing a diverse and extensive array of interpolation techniques on the performance of neural network-based time series predictions.

- **Alternative Measures of Signal Complexity:**

  The approaches presented in this thesis that utilize measures of signal complexity can be adapted to incorporate different complexity metrics by altering the filtering process or the fitness function of the two developed interpolation techniques. Some potential alternatives include:

  – Replacing the Hurst exponent with detrended fluctuation analysis (DFA [119]) or wavelet-based methods [120].

  – Using the fractal dimension calculated via Higuchi's algorithm as a potentially useful prediction filter, particularly considering its ability to distinguish between different signals, as discussed in Appendix E.

  – Exploring autocorrelation functions, three-point correlations, and fractional derivatives as additional options.

  – Substituting the employed entropy measures with alternatives such as sample, range, or approximate entropy, provided that the dataset is sufficiently long [12, 125–127].

Chapter 15

Author's Publications

During his research as a Ph.D. candidate, the author published several articles, which, to some degree, are included in and/or constitute the presented thesis. The following list gives an overview of the published articles, describes how they fit into the presented research and/or extend it, and what the author's role was in each of the publications. In the following *the author* refers to the author of the presented thesis.

- Ref. [128], Sebastian Raubitzek and Thomas Neubauer. Machine Learning and Chaos Theory in Agriculture. ERCIM News, 122, July 2020:

  This article discusses how to use ideas from nonlinear dynamics and machine learning to analyze and predict short time series data as observed in Agriculture for, e.g., annual yields or other annual or monthly sampled time series data. This article sums up the initial ideas of the work presented in this thesis. As is often the case in research, the author's work developed in a slightly different direction than initially proposed as it is difficult to obtain long-term time series data from Austrian agriculture.

  The author developed the presented concepts and wrote the first draft. The other author helped in drafting and finalizing the article.

- Ref. [28], Sebastian Raubitzek and Thomas Neubauer. A fractal interpolation approach to improve neural network predictions for difficult time series data. Expert Systems with Applications, 169:114474, 2021. ISSN 0957-4174. doi: 10.1016/j.eswa.2020.114474. URL http://www.sciencedirect.com/science/article/pii/S0957417420311234. Visited on 2023-04-20:

  A fractal interpolation method is compared to a linear one to improve neural network time series predictions. As described in Section 5.1, the fractal interpolation method uses the Hurst exponent to match the complexities of given sub-intervals of the time series data under study. Further, the neural network architecture is an LSTM neural network. Furthermore, all results and data are analyzed using the Hurst exponent, the fractal dimension, and the spectrum of Lyapunov exponents. The results show that linear and fractal interpolation techniques can significantly improve neural network predictions. This article presents the results from Chapter 8.

The author developed the initial concept, built all programs, and performed all experiments and the corresponding evaluation. Further, the author wrote the first draft of the article. The other author helped in drafting and finalizing the article.

- Ref. [32], Sebastian Raubitzek and Thomas Neubauer. Taming the Chaos in Neural Network Time Series Predictions. Entropy, 23(11), 2021. ISSN 1099-4300. doi: 10.3390/e23111424. URL https://www.mdpi.com/1099-4300/23/11/1424. Visited on 2023-04-20:

  The fractal interpolation method from [28] and Section 5.1 is used on five datasets with varying numbers of interpolation points. Further, these five data sets are forecasted using randomly parameterized ensembles of LSTM neural networks. These randomly parameterized ensemble predictions are then filtered using different complexity measures: The Hurst exponent, the spectrum of Lyapunov exponents, Fisher's information, SVD entropy, and Shannon's entropy. The predictions can be improved by filtering the ensemble predictions. Further, the predictions outperformed baseline predictions using LSTM, GRU, and RNN neural network approaches with one hidden layer. Additionally, all interpolated data sets are analyzed using the mentioned complexity measures. The results of this article are collected in Chapter 9.

  The author developed the original concept and program code and performed all experiments, the validation, and the writing of the first draft. The other author helped in drafting and finalizing the article.

- Ref. [129], Sebastian Raubitzek and Thomas Neubauer. Combining measures of signal complexity and machine learning for time series analyis: A review. Entropy, 23(12), 2021. ISSN 1099-4300. doi: 10.3390/e23121672. URL https://www.mdpi.com/1099-4300/23/12/1672. Visited on 2023-04-20:

  This review article collects many of the publications collected in Chapter 2 to outline the implications of a combined approach of machine learning and measures of signal complexity.

  The author developed the initial concept, performed the literature review, and the writing of the first draft. The other author helped in drafting and finalizing the article.

- Ref. [130], Sebastian Raubitzek and Thomas Neubauer. An exploratory study on the complexity and machine learning predictability of stock market data. Entropy, 24(3), 2022. ISSN 1099-4300. doi: 10.3390/e24030332. URL https://www.mdpi.com/1099-4300/24/3/332. Visited on 2023-04-20:

  This article is an exploratory study aiming to identify trends in signal complexity and predictability in stock market data. Further, the influence of money supply on stock market data and the corresponding change in signal complexity are discussed. This article resulted from experimenting with various complexity metrics on how they can indicate a machine learning algorithm's accuracy on stock market data and how the predictability and complexity of stock market data have changed over the years. The results of this article are not directly part of this thesis, but the performed research inherently influenced the development of the experiments in this thesis as the author's expertise on complexity metrics and the arguments on the predictability of univariate time series data was partially accumulated during the work of this article.

  The author developed all experimental concepts, did all the coding, performed all evaluations, and the writing of the first draft. The other author helped in drafting and finalizing the article.

- Ref. [30], Sebastian Raubitzek, Thomas Neubauer, Jan Friedrich, and Andreas Rauber. Interpolating strange attractors via fractional brownian bridges. Entropy, 24(5), 2022. ISSN 1099-4300. doi: 10.3390/e24050718. URL https://www.mdpi.com/1099-4300/24/5/718. Visited on 2023-04-20:

  This article introduces the developed PhaSpaSto interpolation from Section 5.2. and presents all experimental results from Chapter 10. The idea is first to generate a population of stochastically interpolated time series data using the multipoint fractional Brownian Bridges from [29] and then apply a genetic algorithm to find the pieces of the population that constitute a smooth phase space trajectory. Here a smooth phase space trajectory is found by minimizing the variance of second derivatives along a given phase space trajectory. The results are validated with the Lorenz system and a selection of non-model time series data.

  The multipoint fractional Brownian Bridges program code was developed by Friedrich et al. [29]. The author used this program code to develop PhaSpaSto interpolation. Further, the author built all program code, performed all experiments and evaluations, and the writing of the first draft of the article. The other authors helped in discussing the initial ideas, the experimental setup, the drafting, and in finalizing the article.

- Ref. [31], Sebastian Raubitzek and Thomas Neubauer. Reconstructed phase spaces and lstm neural network ensemble predictions. Engineering Proceedings, 18(1), 2022. ISSN 2673-4591. doi: 10.3390/engproc2022018040. URL https://www.mdpi.com/2673-4591/18/1/40. Visited on 2023-04-20:

  This conference paper again presents the randomly parameterized neural network ensemble predictions similar to Ref. [32]. However, we used PhaSpaSto interpolation as an additional technique and filtered the predictions based on the smoothness of their reconstructed phase space trajectories rather than their signal complexity. The results show that PhaSpaSto interpolation can be used to improve neural network time series predictions. Further, the second derivative along a reconstructed phase space trajectory can be used to filter and thus improve ensemble predictions.

  This article is an excerpt from Chapter 11 such that it features partial results and the concepts on how to interpolate and filter predictions.

  The author performed all coding, the experiments, and the writing of the first draft, where the other author contributed to drafting and finalizing the article.

- Ref. [131], Kevin Mallinger, Sebastian Raubitzek, Thomas Neubauer, and Steven Lade. Potentials and limitations of complexity metrics for the sustainable transition to Farming 4.0. Current Opinion in Environmental Sustainability, 2022. Accepted, not yet published:

  This publication reviews state-of-the-art research on complexity metrics and reconstructed phase spaces for earth sciences and related fields. These ideas are then discussed in the context of machine learning to provide a list of potential applications to improve data analysis in agriculture and related fields.

  The author's contribution to this review resulted from the author's research on the applicability of machine learning methods and complexity in agriculture, as the author was part of the DILAAG Ph.D. school, which focused on agriculture. This review collects some initial but discarded ideas for the presented thesis.

  The author contributed his expertise on the applicability of machine learning algorithms and complexity in agriculture to the literature review, just as the other authors contributed their expertise. Further, the author and the other authors wrote, drafted, and finalized the article collectively.

- Ref. [132],Sebastian Raubitzek, Kevin Mallinger, and Thomas Neubauer. Combining fractional derivatives and machine learning: A review. Entropy, 25(1), 2023. ISSN 1099-4300. doi: 10.3390/e25010035. URL https://www.mdpi.com/1099-4300/25/1/35. Visited on 2023-04-20:

This article reviews state-of-the-art combined approaches of fractional, i.e., non-integer derivatives and non-neural network machine learning. The article categorizes all relevant and referenced publications into three categories, i.e., preprocessing, machine learning, and fractional dynamics, and provides ideas on extending these ideas.

The reason for writing this article was to get an overview of the possibilities of using fractional derivatives to improve machine learning approaches. Specifically, we looked for ideas on extending the techniques presented in this thesis, e.g., to develop an interpolation technique taking into account the inherent memory of a time series by analyzing its spectrum of fractional derivatives or creating a prediction filter based on fractional derivatives.

For this article, the author of this thesis performed the literature review, the literature analysis, and the writing of the first draft. The other authors contributed to improving the discussion on the topic, providing further ideas on the applicability, and by finalizing and drafting the article.

IV. Appendix

Appendix A

Phase Space Embeddings

We discuss the phase space reconstruction for each time series in addition to the theoretical foundations given in Chapter 4. We calculate the phase space embeddings using the method of average mutual information for the time delay, calculating the time delay based on the autocorrelations of a time series and the false nearest neighbors algorithm for the embedding dimension; [64], [69]. For each time series, we first calculated the time delay and afterward, based on the time delay, calculated the embedding dimension. The results for all data sets are collected in Table A.1.

As these algorithms are considered to be estimates for real-life data sets, we also chose a third phase space embedding to be $\tau = 1$ and $d_E = 3$, for each time series. The reason for choosing the third phase space embedding is that this third phase space embedding provided us with reasonable and interpretable phase space plots for all the results presented in Chapter 10. Meaning that when we plotted the time series data in the chosen phase space, we were able to identify smoothed out phase space trajectories compared to edgy ones, as the plots, in many cases, provided us with sort of a round attractor structure. All phase space embeddings for all data sets are depicted three dimensions in the Figures A.1, A.2, A.3, A.4, A.5, A.6, A.7, A.8, A.9, A.10, A.11, A.12, A.13 and A.14. We did not depict redundant phase space portraits, i.e. if two phase space portraits were identical we only depicted it once and referenced it twice. All data sets were normalized to the unit interval for their phase space portraits. Also, data sets that have a visible increasing trend were detrended by subtracting a linear fit.

TABLE A.1: Time delay and embedding dimension calculated using the method of average mutual information (AMI), the autocorrelation function (AC) and the method of false nearest neighbors. Numbers in brackets are results where the employed algorithms didn't find a suitable time delay and we used the default, i.e. $\tau = 1$ and $d_E = 3$.

Data	τ **(AMI)**	d_E **(AMI)**	τ (AC)	d_E **(AC)**
Monthly International Airline Passengers	1	3	3	4
Monthly Mean Temperature in Nottingham Castle	2	3	3	4
Perrin Freres Champagne Sales	1	7	2	5
Car Sales in Quebec	1	6	2	4
NYC Measles Outbreaks	7	1	4	1
Annual Wheat Yields in Austria	1	1	11	1
Annual Maize Yields in Austria	3	3	5	1
CFE specialty monthly writing paper sales	2	1	2	1
Shampoo Sales	1	1	1	1
Dow Jones daily close	2	4	(1)	(3)
USD/GBP exchange	3	4	(1)	(3)
Sunspots	2	1	3	1
Canadian Lynx	5	3	1	1
River Krems discharge	1	1	4	1

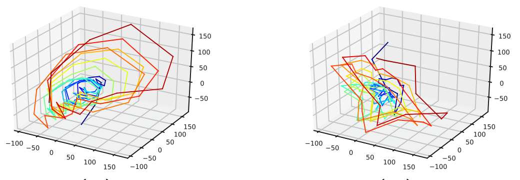


FIGURE A.1: Reconstructed phase space trajectories for different time delays for the monthly international airline passengers data set.
(a): AMI and $\tau = 1$time delay;
(b): ACF time delay;

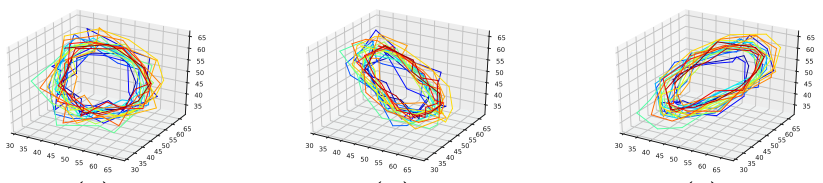


FIGURE A.2: Reconstructed phase space trajectories for different time delays for the monthly mean temperature in Nottingham castle data set.
(a): AMI time delay;
(b): ACF time delay;
(c): $\tau = 1$;

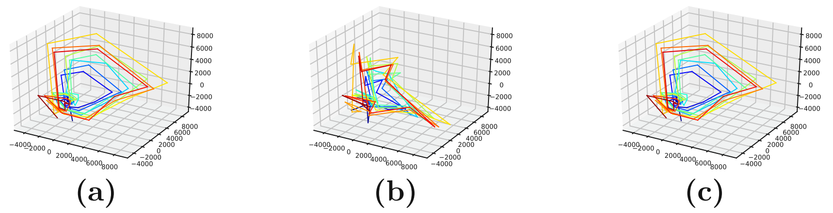


FIGURE A.3: Reconstructed phase space trajectories for different time delays for the Perrin Freres champagne sales data set.
(a): AMI time delay;
(b): ACF time delay;
(c): $\tau = 1$;

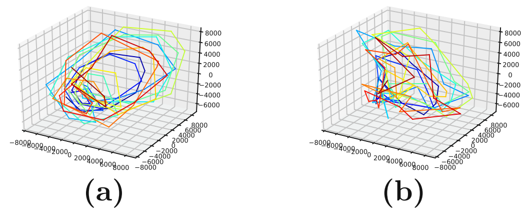


FIGURE A.4: Reconstructed phase space trajectories for different time delays for the car sales in Quebec data set.
(a): AMI and $\tau = 1$ time delay;
(b): ACF time delay;

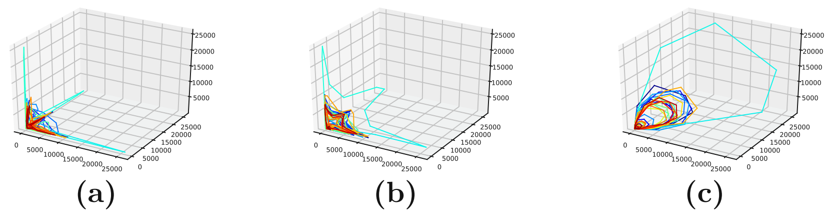


FIGURE A.5: Reconstructed phase space trajectories for different time delays for the measles cases in NYC data set.
(a): AMI time delay;
(b): ACF time delay;
(c): $\tau = 1$;

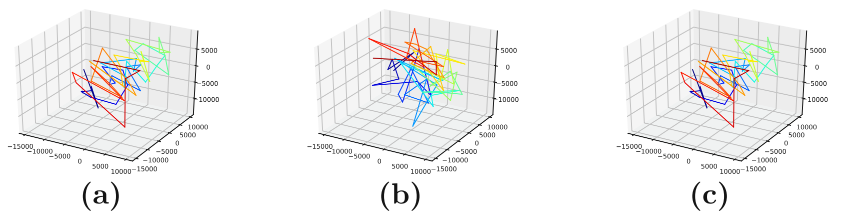


FIGURE A.6: Reconstructed phase space trajectories for different time delays for the annual wheat yields in Austria data set.
(a): AMI time delay;
(b): ACF time delay;
(c): $\tau = 1$;

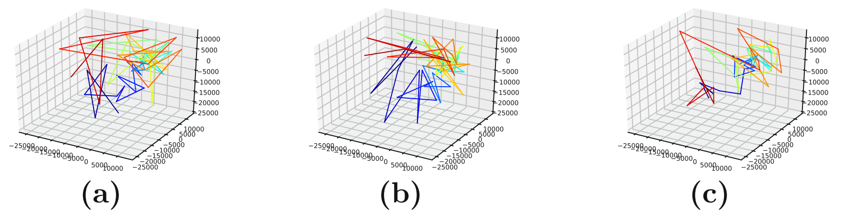


FIGURE A.7: Reconstructed phase space trajectories for different time delays for the annual maize yields in Austria data set.
(a): AMI time delay;
(b): ACF time delay;
(c): $\tau = 1$;

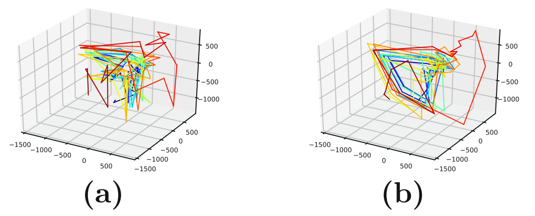


FIGURE A.8: Reconstructed phase space trajectories for different time delays for the CFE specialty writing paper sales data set.
(a): AMI and ACF time delay;
(b): $\tau = 1$;

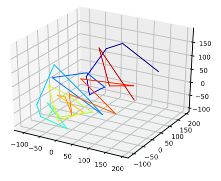
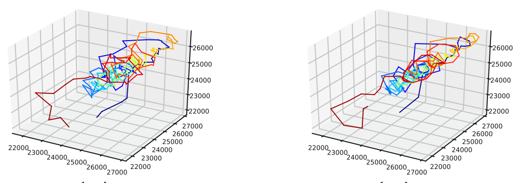
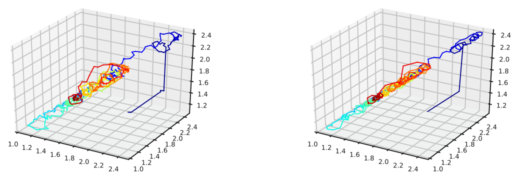


FIGURE A.9: Reconstructed phase space trajectories for the shampoo sales data set, AMI, ACF and $\tau = 1$ time delay, as all of them are the same for this data set.



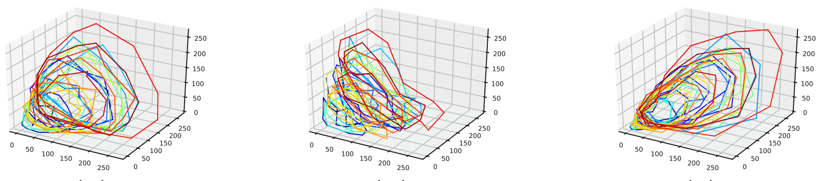
(a) **(b)**

FIGURE A.10: Reconstructed phase space trajectories for different time delays for the Dow Jones daily close in 2018 data set.
(a): AMI time delay;
(b): ACF and $\tau = 1$;



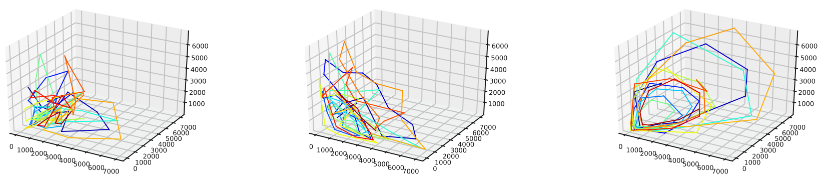
(a) **(b)**

FIGURE A.11: Reconstructed phase space trajectories for different time delays for the USD/GBP exchange rate data set.
(a): AMI time delay;
(c):ACF and $\tau = 1$;



(a) **(b)** **(c)**

FIGURE A.12: Reconstructed phase space trajectories for different time delays for the sunspots data set.
(a): AMI time delay;
(b): ACF time delay;
(c): $\tau = 1$;



(a) **(b)** **(c)**

FIGURE A.13: Reconstructed phase space trajectories for different time delays for the Canadian lynx data set.
(a): AMI time delay;
(b): ACF time delay;
(c): $\tau = 1$;

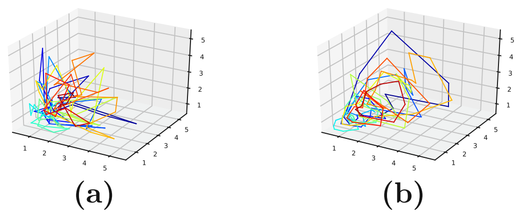


FIGURE A.14: Reconstructed phase space trajectories for different time delays for the River Krems discharge data set.
a): ACF time delay;
(b): AMI and $\tau = 1$;

Appendix B

Additional Material Chapter 9

This appendix provides additional material to Chapter 9. Further, as Chapter 9 is based on [32], all material that is part of [32] but not of Chapter 9 is collected in this Appendix.

Thus, this Appendix provides plots on the varying complexities of interpolated time series data B.2, and all prediction results that are not provided in Chapter 9.

B.1 Baseline Predictions

Baseline predictions, i.e. a simple RNN, LSTM and GRU (For a discussion of these methods see Chapter 6), are used to be compared to the obtained randomly paramterized LSTM ensemble predictions. All three types of neural network layers are reasonable tools for predicting time series data.

Each baseline prediction was done using a neural network with one hidden layer containing varying numbers of neurons, i.e. LSTM, GRU or simple RNN neurons. Further, each neural network was trained with a batch size of 2 and verbose was set to 2. For the activation of the recurrent neural network, `hard_sigmoid` was chosen. And the activation function of the output layer is `relu`. For the initialization,`glorot_uniform` was used for the LSTM layer, `orthogonal` was used as the recurrent initializer and `glorot_uniform` for the Dense layer. For the LSTM layer the bias was set to `use_bias=True`, with a corresponding `bias_initializer="zeros"`. Further, no constraints or regularizers or drop out criteria were used for the recurrent and the `Dense` layers. As optimizer `rmsprop` was used and, the loss was calculated using `mean_squared_error`. The output node returned only one result, i.e., the next time step.

TABLE B.1: [110]

Data	Architecture	Input nodes	Hidden Layer Neurons	Epochs
ailrine passengers	LSTM	20	30	50
airline passengers	GRU	20	30	50
airline passengers	RNN	20	30	50
Monthly car sales in Quebec	LSTM	20	30	45
Monthly car sales in Quebec	GRU	20	26	55
Monthly car sales in Quebec	RNN	20	30	45
Mean temperature	LSTM	20	30	50
Mean temperature	GRU	20	30	50
Mean temperature	RNN	20	30	50
Perrin Freres champagne sales	LSTM	25	33	105
Perrin Freres champagne sales	GRU	25	20	55
Perrin Freres champagne sales	RNN	20	100	60
CFE specialty writing paper sales	LSTM	25	33	104
CFE specialty writing paper sales	GRU	25	20	55
CFE specialty writing paper sales	RNN	20	100	61

TABLE B.2: Baseline RMSE for all datasets, LSTM.

Dataset	Train Error	Test Error	Step-by-Step Error
Monthly international airline passengers	0.04987	0.08960	0.11902
Monthly car sales in Quebec	0.10666	0.11423	0.11269
Monthly mean air temperature in Nottingham Castle	0.06874	0.06193	0.05931
Perrin Freres monthly champagne sales	0.05589	0.05978	0.06915
CFE specialty monthly writing paper sales	0.06282	0.20740	0.21451

TABLE B.3: Baseline RMSE for all datasets, GRU.

Dataset	Train Error	Test Error	Step-by-Step Error
Monthly international airline passengers	0.04534	0.07946	0.10356
Monthly car sales in Quebec	0.10493	0.11166	0.11170
Monthly mean air temperature in Nottingham Castle	0.07048	0.06572	0.06852
Perrin Freres monthly champagne sales	0.06276	0.05214	0.06506
CFE specialty monthly writing paper sales	0.06470	0.20859	0.21653

TABLE B.4: Baseline RMSE for all datasets, RNN.

Dataset	Train Error	Test Error	Step-by-Step Error
Monthly international airline passengers	0.05606	0.08672	0.10566
Monthly car sales in Quebec	0.08950	0.11585	0.11827
Monthly mean air temperature in Nottingham Castle	0.07467	0.07008	0.06588
Perrin Freres monthly champagne sales	0.06178	0.05685	0.07313
CFE specialty monthly writing paper sales	0.06971	0.21610	0.21086

B.2 Complexity plots for all data sets

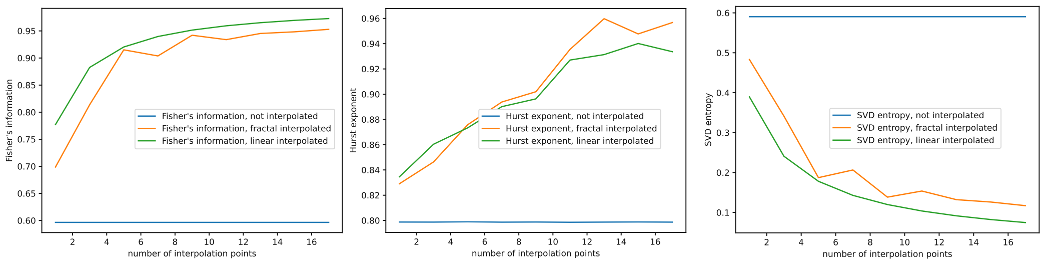


FIGURE B.1: Plots for Fisher's information, the Hurst exponent and SVD entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear-interpolated data, car sales in Quebec data set.

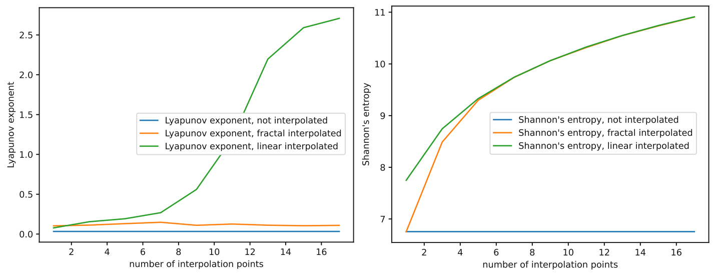


FIGURE B.2: Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear-interpolated data, monthly car sales in Quebec data set.

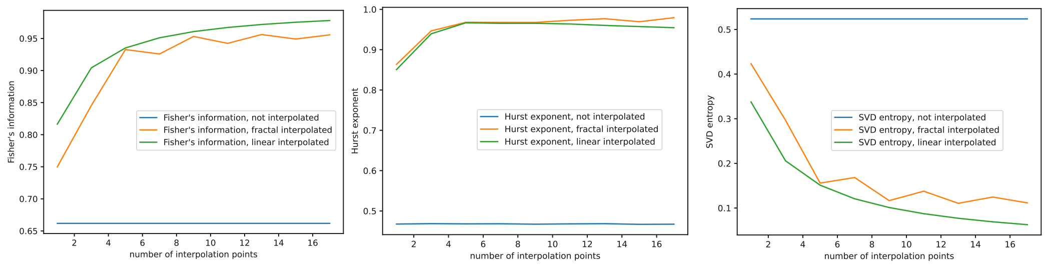


FIGURE B.3: Plots for Fisher's information, the Hurst exponent and SVD entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear-interpolated data, monthly mean temperature in Nottingham castle data set.

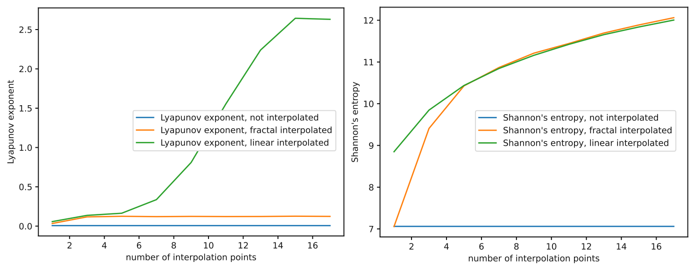


FIGURE B.4: Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear-interpolated data, monthly mean temperature in Nottingham castle data set.

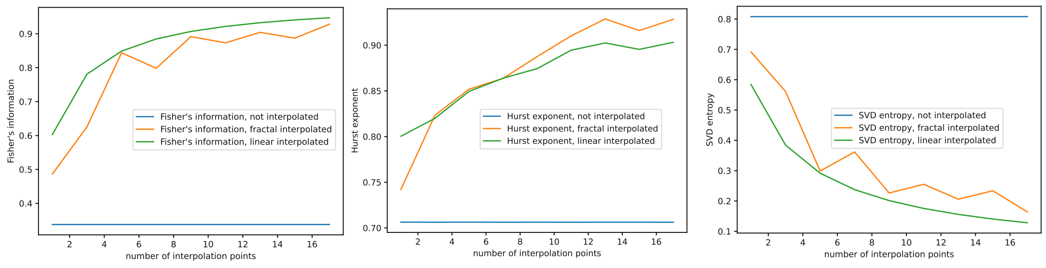


FIGURE B.5: Plots for Fisher's information, the Hurst exponent and SVD entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear-interpolated data, Perrin Freres monthly champagne sales data set.

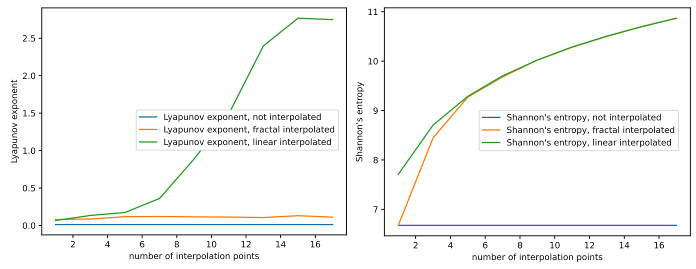


FIGURE B.6: Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear-interpolated data, Perrin Freres monthly champagne sales data set.

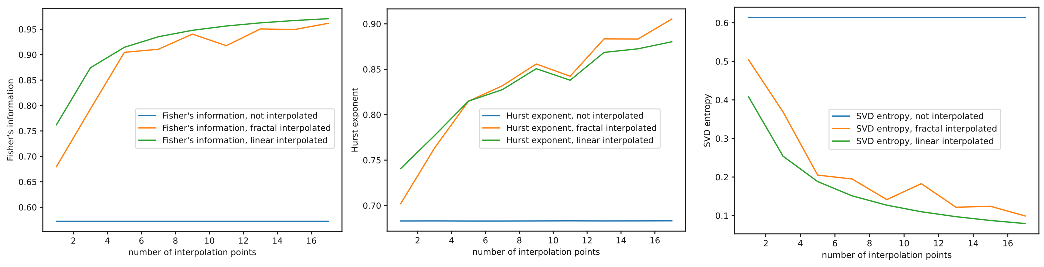


FIGURE B.7: Plots for Fisher's information, the Hurst exponent and SVD entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear-interpolated data, CFE specialty monthly writing paper sales data set.

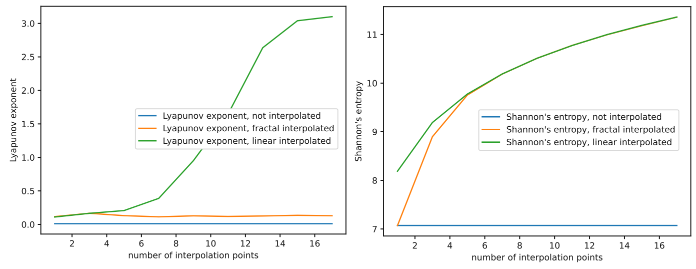


FIGURE B.8: Plots for the Largest Lyapunov exponent and Shannon's entropy depending on the number of interpolation points for the non-interpolated, the fractal interpolated and the linear-interpolated data, CFE specialty monthly writing paper sales data set.

B.3 Error Tables

Table B.5: Error table for the monthly car sales in Quebec dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	shannon fisher	0.08814 ± 0.01496
non-interpolated	-	shannon svd	0.08814 ± 0.01496
non-interpolated	-	fisher hurst	0.09098 ± 0.01511
non-interpolated	-	svd hurst	0.09098 ± 0.01511
non-interpolated	-	fisher	0.09932 ± 0.00602
fractal-interpolated	**13**	**shannon fisher**	**0.08099 ± 0.00961**
fractal-interpolated	**13**	**shannon svd**	**0.08099 ± 0.00961**
fractal-interpolated	9	lyap hurst	0.08585 ± 0.01546
fractal-interpolated	17	fisher lyap	0.08659 ± 0.01869
fractal-interpolated	17	svd lyap	0.08659 ± 0.01869
linear-interpolated	**11**	**lyap hurst**	**0.07567 ± 0.03563**
linear-interpolated	7	fisher	0.08500 ± 0.02254
linear-interpolated	7	fisher svd	0.08500 ± 0.02254
linear-interpolated	7	fisher shannon	0.08500 ± 0.02254
linear-interpolated	5	svd	0.08500 ± 0.02254

Table B.6: Error table for the monthly mean air temperature in Nottingham dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	shannon fisher	0.05728 ± 0.00418
non-interpolated	-	fisher svd	0.05877 ± 0.01496
non-interpolated	-	svd	0.05877 ± 0.01496
non-interpolated	-	svd shannon	0.05877 ± 0.01496
non-interpolated	-	shannon fisher	0.05901 ± 0.00263
fractal-interpolated	**1**	**shannon svd**	**0.05724 ± 0.00495**
fractal-interpolated	7	shannon hurst	0.05873 ± 0.00684
fractal-interpolated	5	shannon lyap	0.05943 ± 0.01648
fractal-interpolated	7	fisher hurst	0.05946 ± 0.00519
fractal-interpolated	3	hurst	0.05998 ± 0.00544
linear-interpolated	**3**	**lyap hurst**	**0.05625 ± 0.00632**
linear-interpolated	**7**	**lyap fisher**	**0.05635 ± 0.00481**
linear-interpolated	3	hurst	0.05742 ± 0.00623
linear-interpolated	7	lyap svd	0.05786 ± 0.00511
linear-interpolated	3	svd lyap	0.05862 ± 0.00416

Table B.7: Error table for the Perrin Freres monthly champagne sales dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	shannon fisher	0.06383 ± 0.02706
non-interpolated	-	shannon svd	0.06383 ± 0.02706
non-interpolated	-	hurst fisher	0.07245 ± 0.01571
non-interpolated	-	hurst svd	0.07387 ± 0.01695
non-interpolated	-	hurst lyap	0.07403 ± 0.01740
fractal-interpolated	**13**	**fisher hurst**	**0.04968 ± 0.02155**
fractal-interpolated	**13**	**svd hurst**	**0.04968 ± 0.02155**
fractal-interpolated	**11**	**shannon hurst**	**0.05001 ± 0.01416**
fractal-interpolated	17	hurst lyap	0.05166 ± 0.01066
fractal-interpolated	13	hurst	0.05386 ± 0.0154
linear-interpolated	17	hurst	0.05449 ± 0.0280
linear-interpolated	17	hurst shannon	0.05449 ± 0.0280
linear-interpolated	9	fisher	0.05730 ± 0.03250
linear-interpolated	9	fisher shannon	0.05730 ± 0.03250
linear-interpolated	9	svd fisher	0.05730 ± 0.03250

Table B.8: Error table for the CFE specialty monthly writing paper sales dataset. The bold results are the three best ones for this dataset.

Interpolation Technique	# of Interpolation Points	Filter	Error
non-interpolated	-	shannon fisher	0.18996 ± 0.00957
non-interpolated	-	shannon svd	0.19200 ± 0.01041
non-interpolated	-	hurst fisher	0.19314 ± 0.01057
non-interpolated	-	hurst svd	0.19314 ± 0.01057
non-interpolated	-	fisher hurst	0.19328 ± 0.01021
fractal-interpolated	**5**	**fisher lyap**	**0.17685 ± 0.00601**
fractal-interpolated	5	svd lyap	0.17685 ± 0.00601
fractal-interpolated	5	lyap hurst	0.18138 ± 0.00939
fractal-interpolated	1	hurst fisher	0.18332 ± 0.00751
fractal-interpolated	1	hurst svd	0.18332 ± 0.00751
linear-interpolated	**7**	**hurst fisher**	**0.17651 ± 0.01096**
linear-interpolated	**7**	**hurst svd**	**0.17651 ± 0.01096**
linear-interpolated	15	shannon lyap	0.18026 ± 0.00973
linear-interpolated	3	shannon hurst	0.18149 ± 0.01623
linear-interpolated	7	fisher lyap	0.18201 ± 0.00619

B.4 Prediction Plots

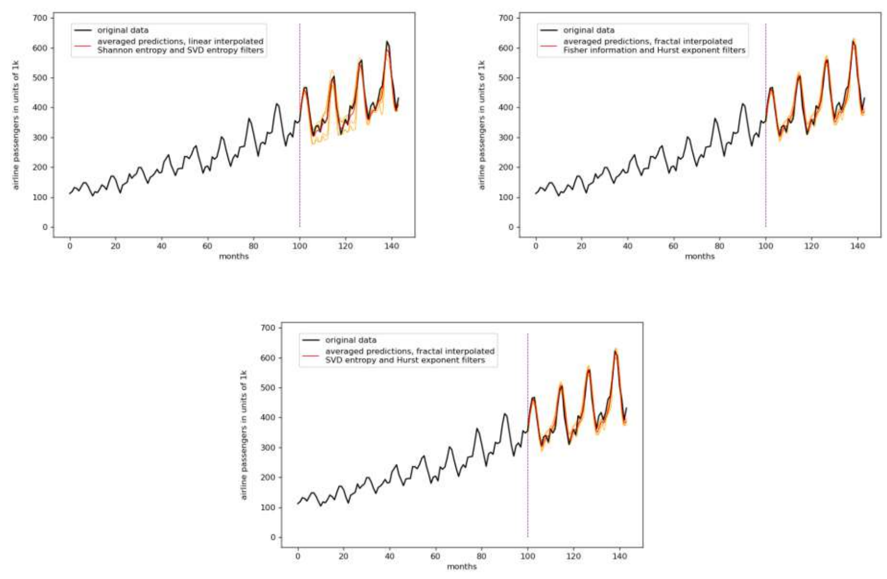


Figure B.9: Best results monthly airline passengers dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: linear-interpolated, 3 interpolation points, Shannon entropy and SVD entropy filter, error: 0.03542 ± 0.00625; fractal-interpolated, 1 interpolation point, Fisher's information and Hurst exponent filter, error: 0.03597 ± 0.00429; fractal-interpolated, 1 interpolation point, SVD entropy and Hurst exponent filter, error: 0.03597 ± 0.00429.

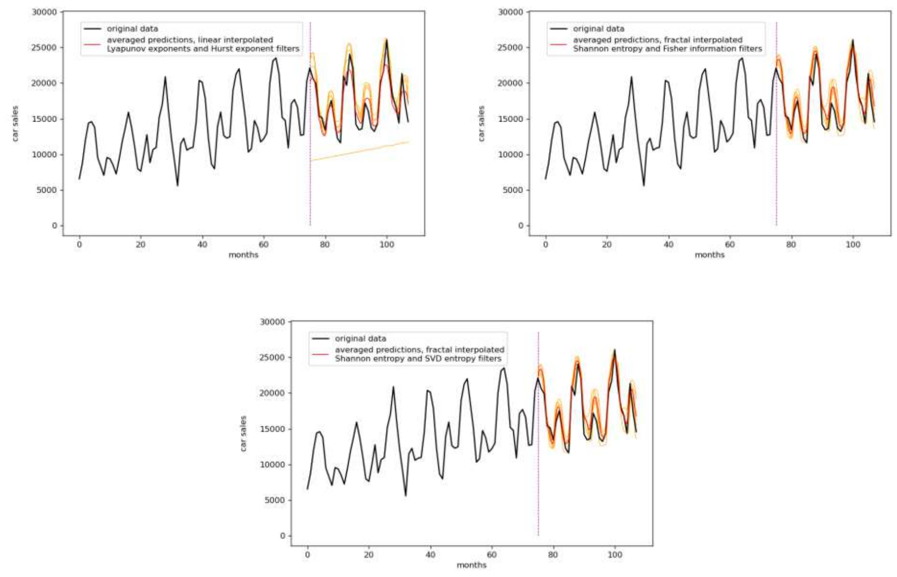


Figure B.10: Best results monthly car sales in Quebec dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: linear-interpolated, 11 interpolation points, Lyapunov exponents and Hurst exponent filter, error: 0.097567 ± 0.03563; fractal-interpolated, 13 interpolation points, Shannon's entropy and Fisher's information filter, error: 0.08099 ± 0.00961; fractal-interpolated, 13 interpolation points, Shannon's entropy and SVD entropy filter, error: 0.08099 ± 0.00961.

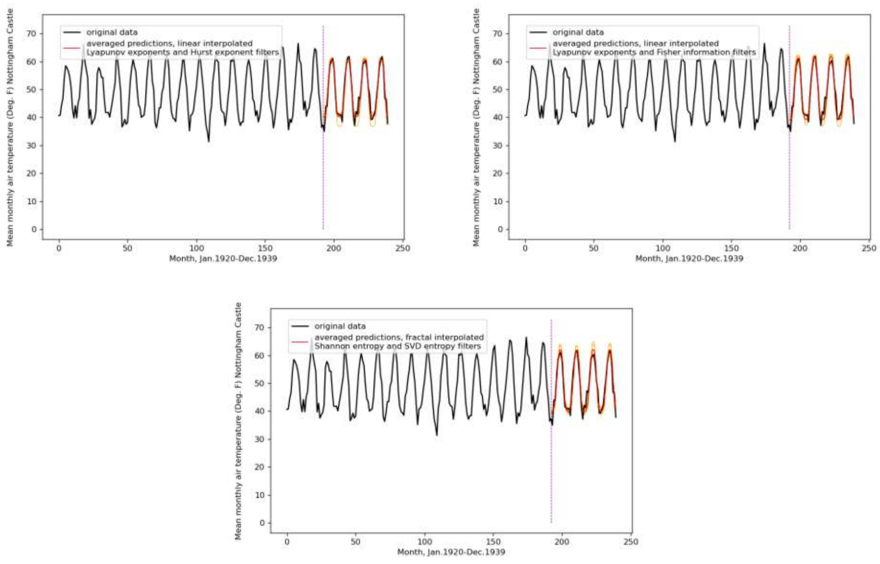


FIGURE B.11: Best results monthly mean temperatures in Nottingham castle dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: linear-interpolated, 3 interpolation points, Lyapunov exponents and Hurst exponent filter, error: 0.05625 ± 0.00632; linear-interpolated, 7 interpolation points, Lyapunov exponent and Fisher's information filter, error: 0.05635 ± 0.00481; fractal-interpolated, 1 interpolation point, Shannon's entropy and SVD entropy filter, error:0.05724 ± 0.00495.

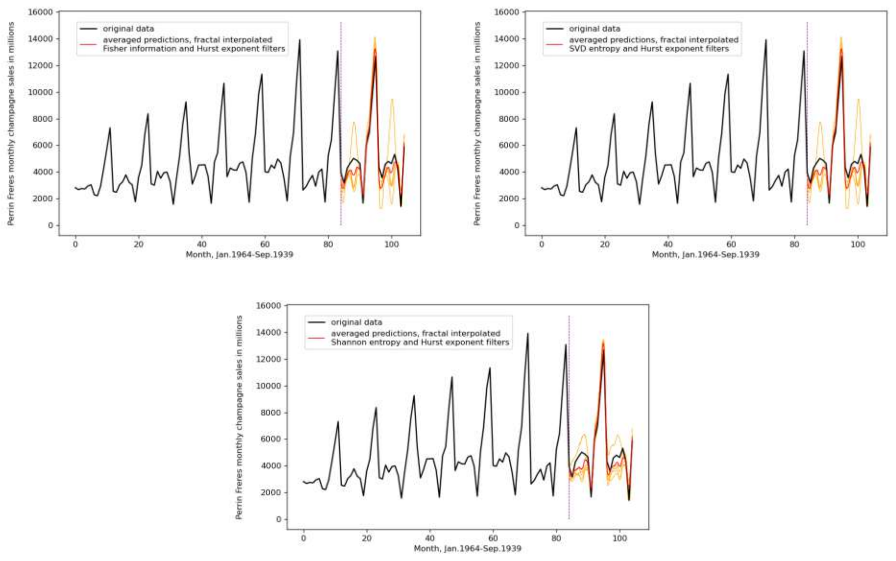


FIGURE B.12: Best results for the Perrin Freres monthly champagne sales dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: fractal-interpolated, 13 interpolation points, Fisher's information and Hurst exponent filter, error: 0.04968 ± 0.02155; fractal-interpolated, 13 interpolation points, SVD entropy and Hurst exponent filter, error: 0.04968 ± 0.02155; fractal-interpolated, 11 interpolation point, Shannon's entropy and Hurst exponent filter, error: 0.05001 ± 0.01416.

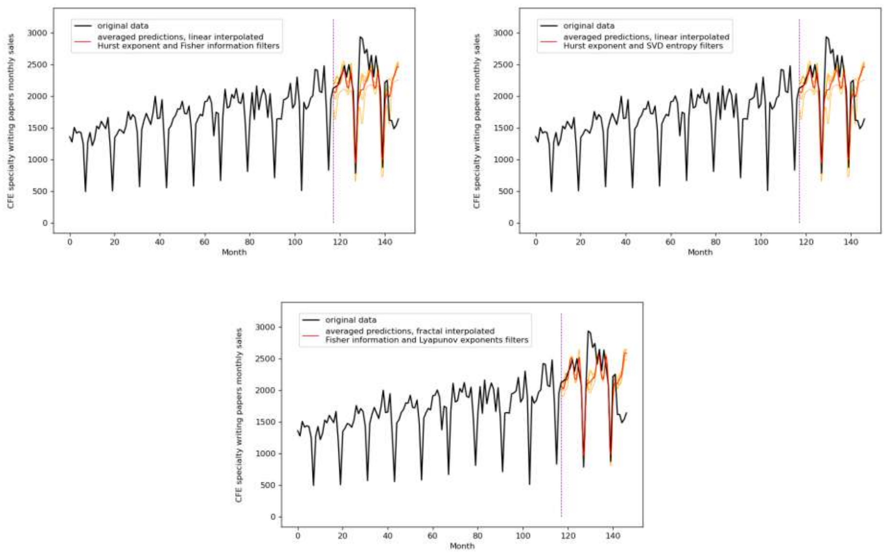


FIGURE B.13: Best results CFE specialty monthly writing paper sales dataset. The orange lines show the remaining ensemble predictions after filtering, the red line is the averaged ensemble prediction. Left to right: linear-interpolated, 7 interpolation points, Hurst exponent and Fisher's information filter, error: 0.17651 ± 0.01096; fractal-interpolated, 7 interpolation points, Hurst exponent and SVD entropy filter, error: 0.17651 ± 0.01096; fractal-interpolated, 5 interpolation points, Fisher's information and Lyapunov exponents filter, error: 0.17685 ± 0.00601.

B.5 Baseline Predictions

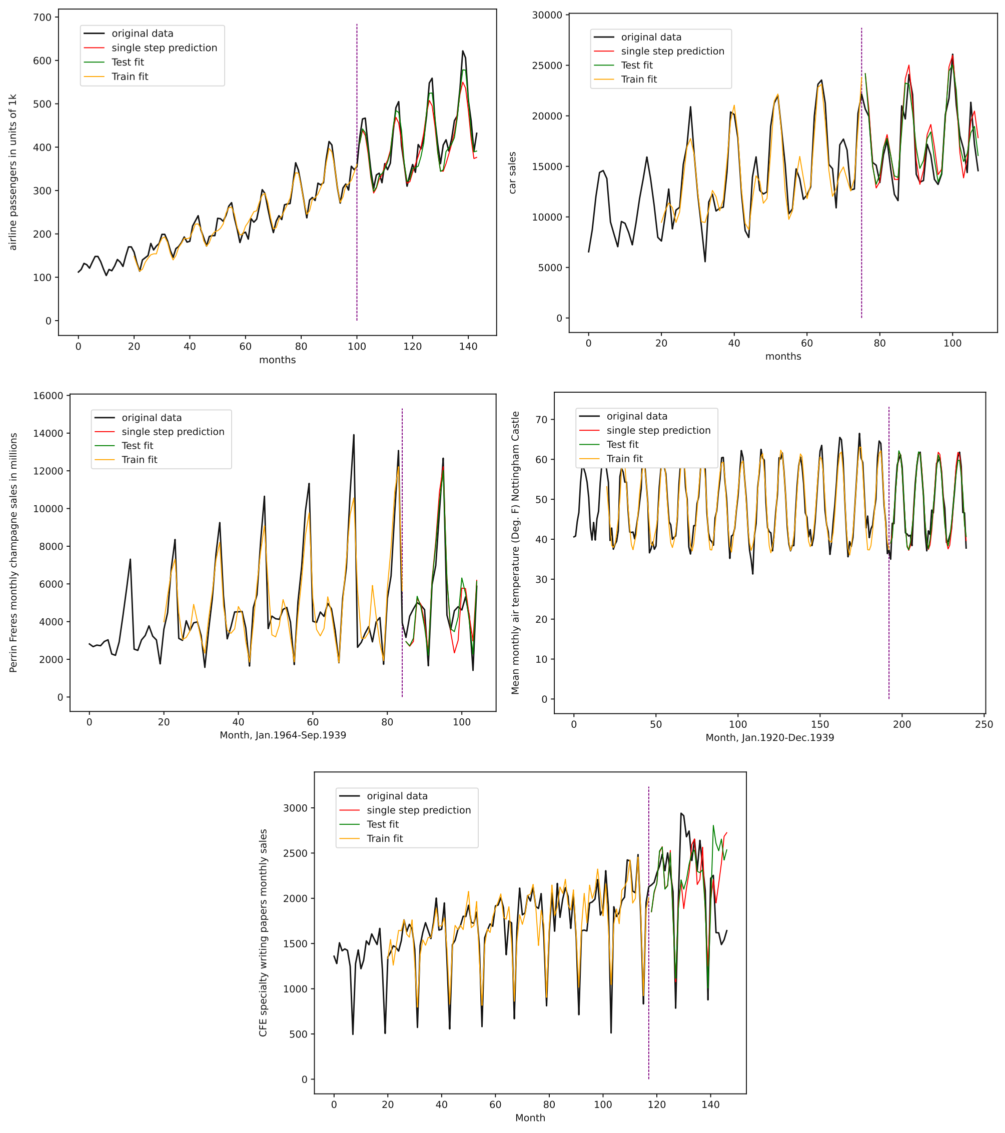


FIGURE B.14: Baseline predictions for each dataset, LSTM.

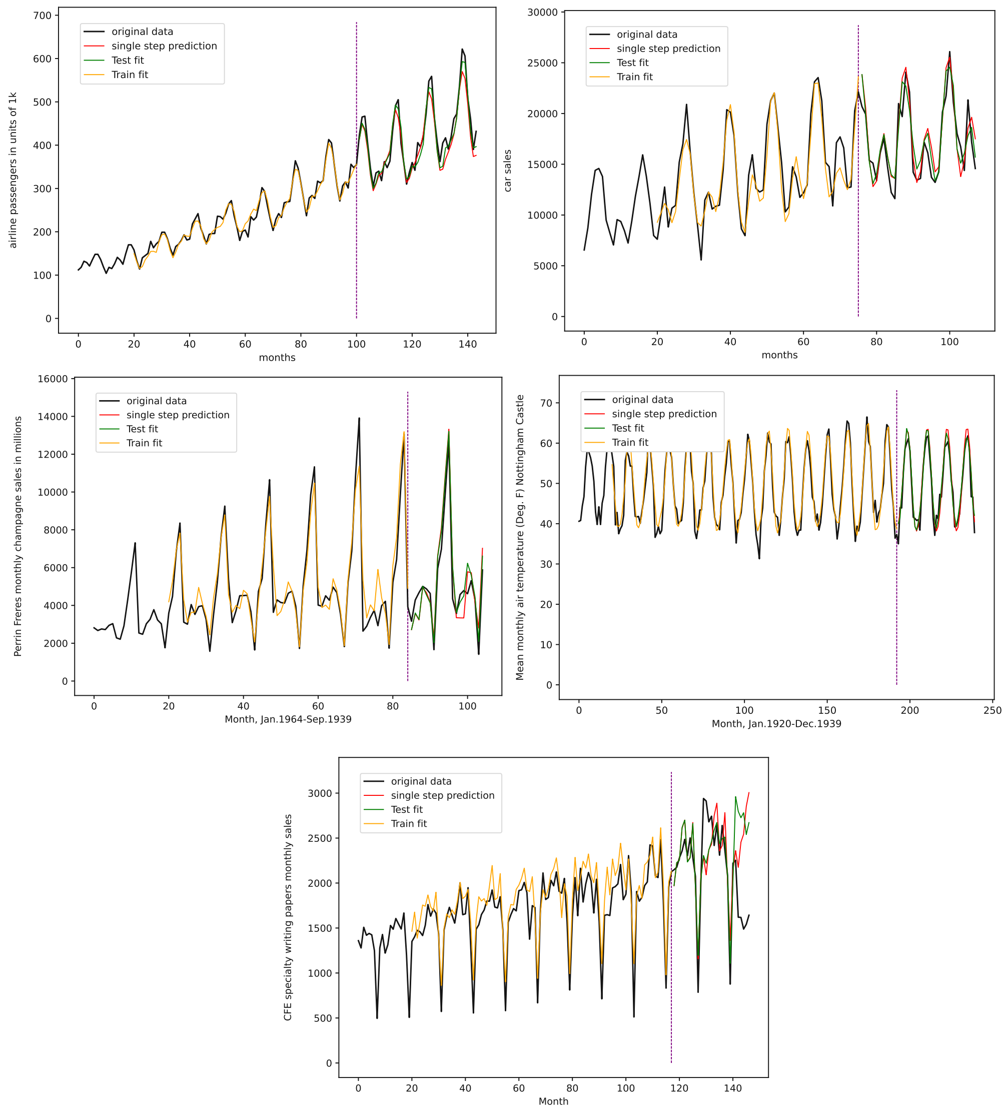


FIGURE B.15: Baseline predictions for each dataset, GRU.

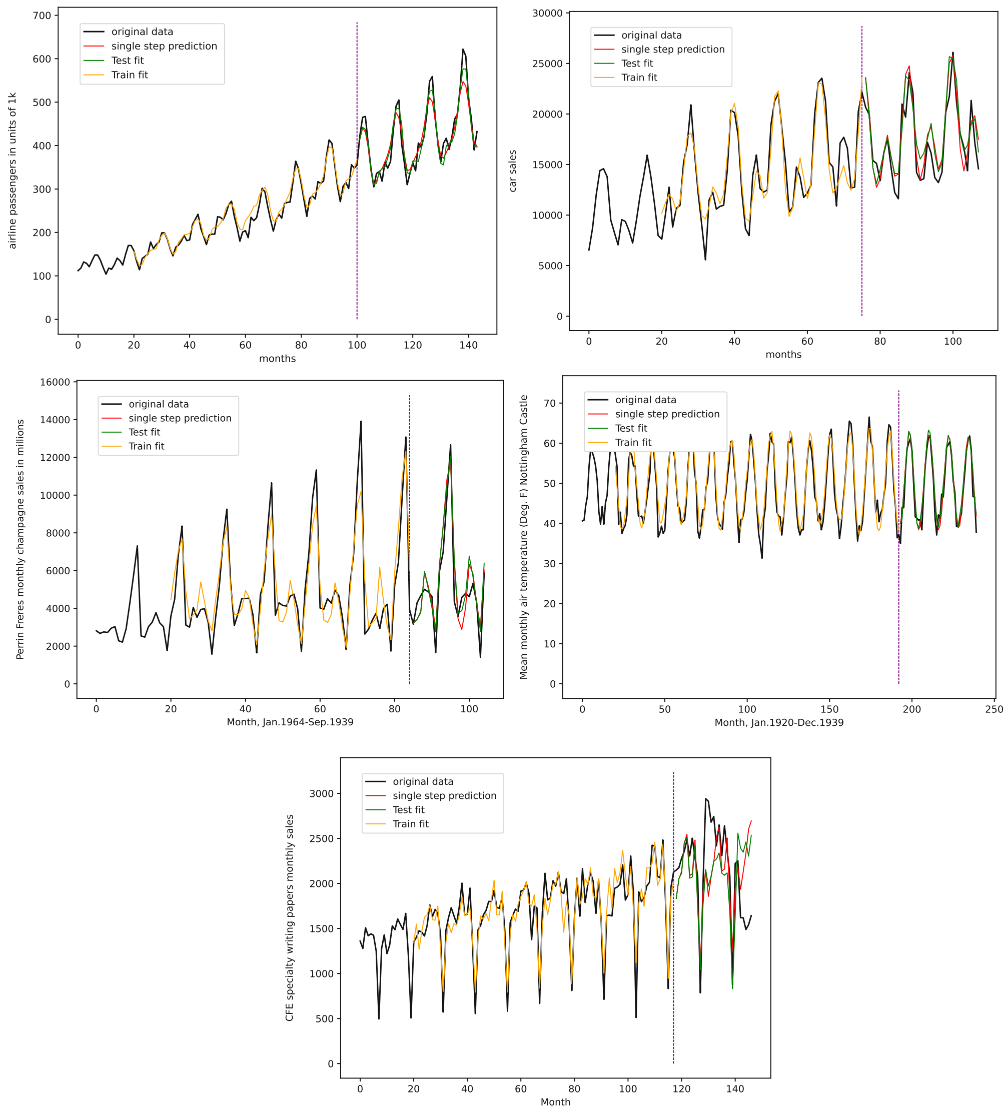


Figure B.16: Baseline predictions for each dataset, RNN.

B.6 Unfiltered Ensemble Prediction Errors

Table B.9: Unfiltered ensemble prediction errors for all data sets, fractal interpolated

# of interpolation points	**Monthly international airline passengers**	**Monthly car sales in Quebec**	**Monthly mean air temperature in Nottingham Castle**	**Perrin Freres monthly champagne sales**	**CFE specialty monthly writing paper sales**
0	0.16771±0.01537	0.20779±0.03961	0.28503±0.04816	0.20641±0.05741	0.38041±0.05203
1	0.16076±0.01917	0.20239±0.04121	0.31272±0.04877	0.19946±0.06391	0.38070±0.05338
3	0.16487±0.01758	0.18964±0.04448	0.30624±0.05029	0.18770±0.07113	0.38474±0.05309
5	0.15347±0.01988	0.18814±0.04654	0.30236±0.05090	0.18858±0.07011	0.37807±0.05443
7	0.15710±0.02002	018252±0.04657	0.30020±0.05112	0.18439±0.07311	0.37935±0.05479
9	0.14610±0.02088	0.17944±0.04784	0.29381±0.05185	0.17920±0.07327	0.39043±0.05280
11	0.15410±0.02082	0.18092±0.04787	0.30588±0.05123	0.18487±0.07284	0.36708±0.05583
13	0.15361±0.02014	0.17582±0.04781	0.30105±0.05151	0.18408±0.07380	0.39382±0.05228
15	0.15359±0.02091	0.17476±0.04773	0.31103±0.05033	0.17973±0.07522	0.39385±0.05276
17	0.16245±0.02004	0.17571±0.04754	0.30171±0.05125	0.18219±0.07404	0.37625±0.05515

Table B.10: Unfiltered ensemble prediction errors for all data sets, linear interpolated

# of interpolation points	**Monthly international airline passengers**	**Monthly car sales in Quebec**	**Monthly mean air temperature in Nottingham Castle**	**Perrin Freres monthly champagne sales**	**CFE specialty monthly writing paper sales**
0	0.16771±0.01537	0.20779±0.03961	0.28503±0.04816	0.20641±0.05741	0.38041±0.05203
1	0.16294±0.01917	0.20283±0.04332	0.26516±0.05052	0.21457±0.06303	0.36967±0.05309
3	0.15199±0.01758	0.19681±0.04584	0.29448±0.05088	0.19397±0.0691	0.37922±0.05435
5	0.15088±0.01988	0.17882±0.04761	0.27367±0.05107	0.19438±0.07132	0.35778±0.05520
7	0.14553±0.02002	0.17105±0.04771	0.28405±0.05130	0.18642±0.07327	0.37685±0.05533
9	0.15033±0.02088	0.18831±0.04813	0.28135±0.05186	0.20273±0.07183	0.35956±0.05501
11	0.15664±0.02082	0.18738±0.04832	0.28566±0.05130	0.18151±0.07370	0.38573±0.05557
13	0.15459±0.02014	0.17700±0.04855	0.30069±0.05168	0.19560±0.07281	0.38573±0.05504
15	0.15090±0.02091	0.18368±0.04825	0.30349±0.05114	0.19760±0.07235	0.38506±0.05515
17	0.15428±0.02004	0.18468±0.04903	0.28451±0.05208	0.18838±0.07347	0.36044±0.05591

Appendix C

Additional Material Chapter 10

C.1 Additional Plots

This section provides additional plots for all data sets discussed in Section 10.1.2. As such we plotted the evolution of errors for the validation depending on the varying number of interpolation points, i.e. the errors from Tables 10.2, 10.3, 10.4, 10.5 and 10.6. Further, we added each time series and the corresponding best validation interpolation, and the corresponding phase space plots.

The last part of this appendix discusses how different phase space embeddings influence PhaSpaSto interpolation.

C.1.1 Evolution of Errors

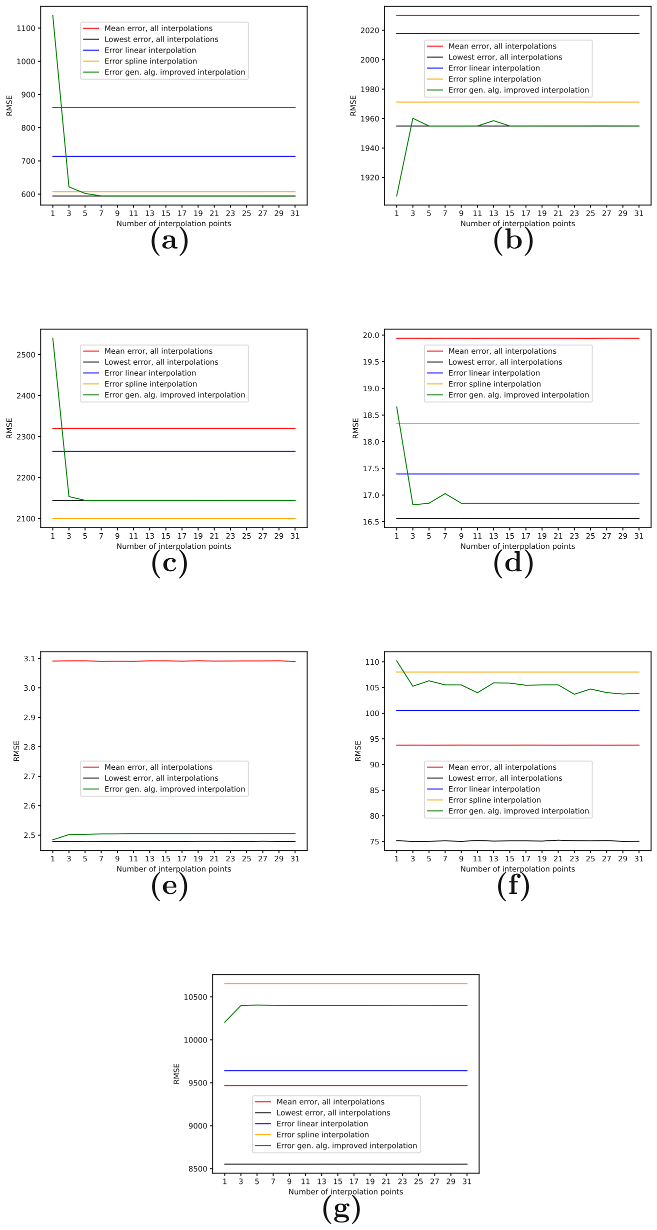
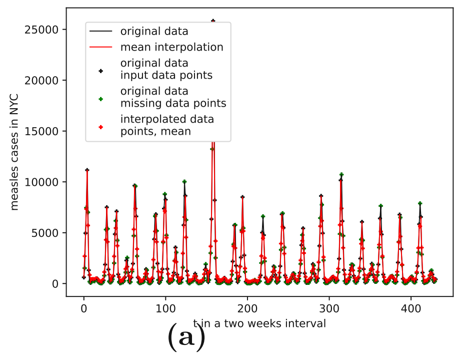
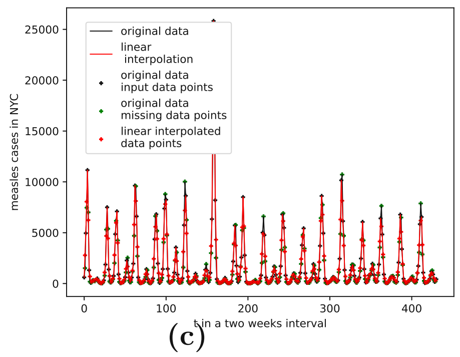


FIGURE C.1: Evolution of errors depending on the number of interpolation points for the non-model data validation.
(a): Measles cases in NYC data set, results from Table 10.2;
(b): Car Sales in Quebec data set, results from Table 10.3;
(c): Perrin Freres champagne sales data set, results from Table 10.4;
(d): Monthly international airline passengers data set, results from Table 10.5;
(e): Monthly mean temperature in Nottingham castle data set, results from Table 10.6;
(f): Shampoo sales data set, results from Table 10.7;
(e): Annual maize yields in Austria data set, results from Table 10.8;

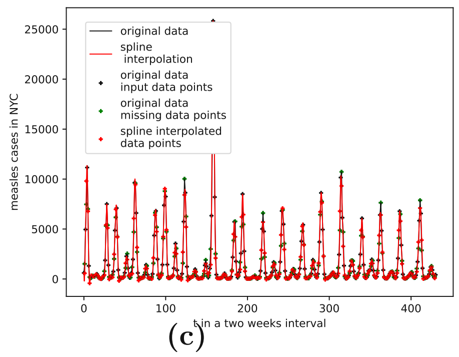
C.1.2 NYC Measles Outbreaks



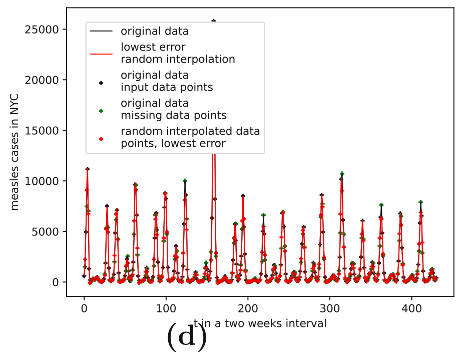
(a)



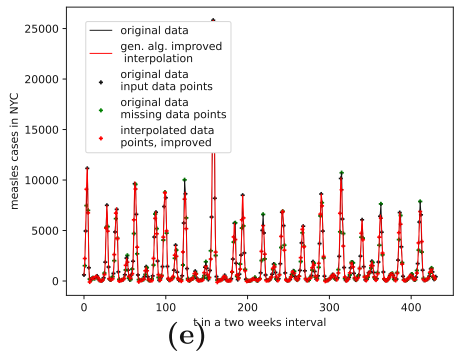
(c)



(c)

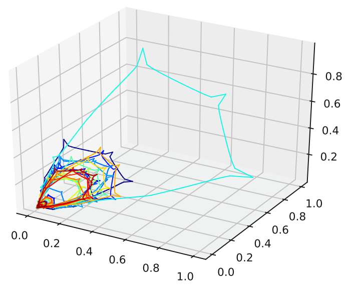


(d)

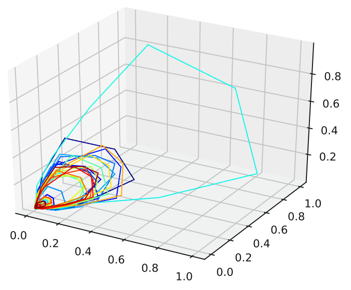


(e)

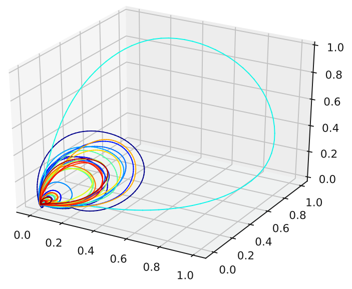
FIGURE C.2: Interpolated validation data (25 interpolation points) for the measles cases in NYC data set.
(a): Average population validation;
(b): Validation, linear interpolation;
(c): Validation, spline interpolation;
(d): Validation, best random interpolation;
(e): Validation, gen. alg. improved interpolation;



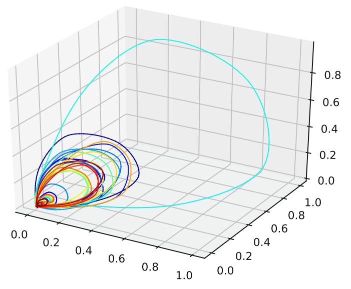
(a)



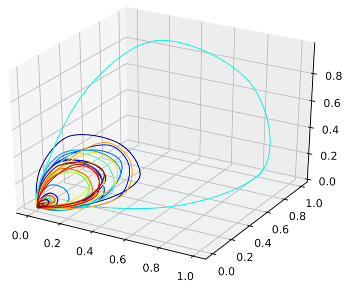
(b)



(c)



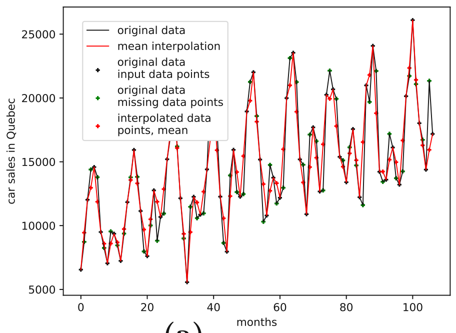
(d)



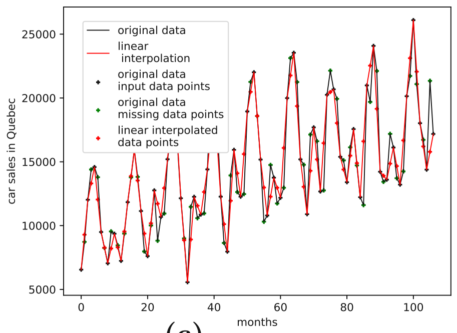
(e)

FIGURE C.3: Reconstructed validation attractors (25 interpolation points) for the measles cases in NYC data set.
(a): Reconstructed attractor, average population validation interpolation;
(b): Reconstructed attractor, linear interpolation;
(c): Reconstructed attractor, spline interpolation;
(d): Reconstructed attractor, best random validation interpolation;
(e): Reconstructed attractor, gen. alg. improved validation interpolation;

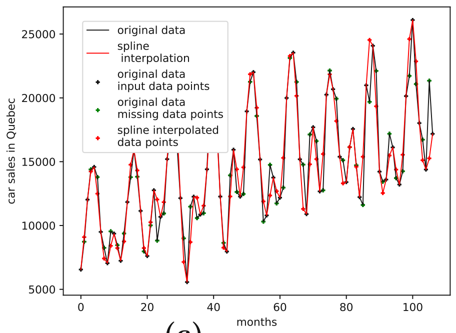
C.1.3 Car Sales in Quebec



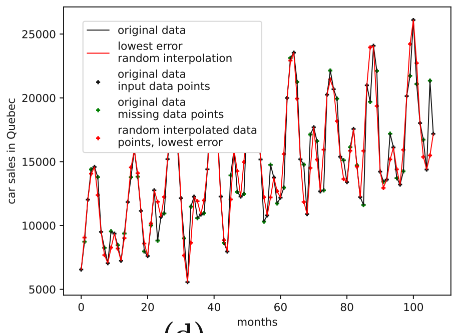
(a)



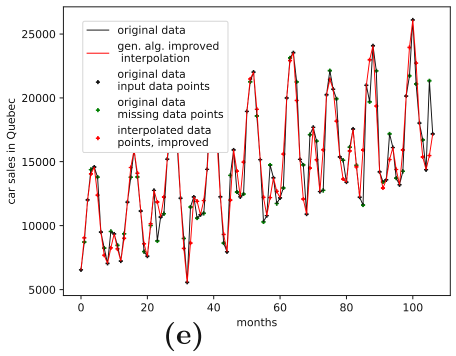
(c)



(c)



(d)



(e)

FIGURE C.4: Interpolated validation data (one interpolation point) for the car sales in Quebec data set.
(a): Average population validation;
(b): Validation, linear interpolation;
(c): Validation, spline interpolation;
(d): Validation, best random interpolation;
(e): Validation, gen. alg. improved interpolation;

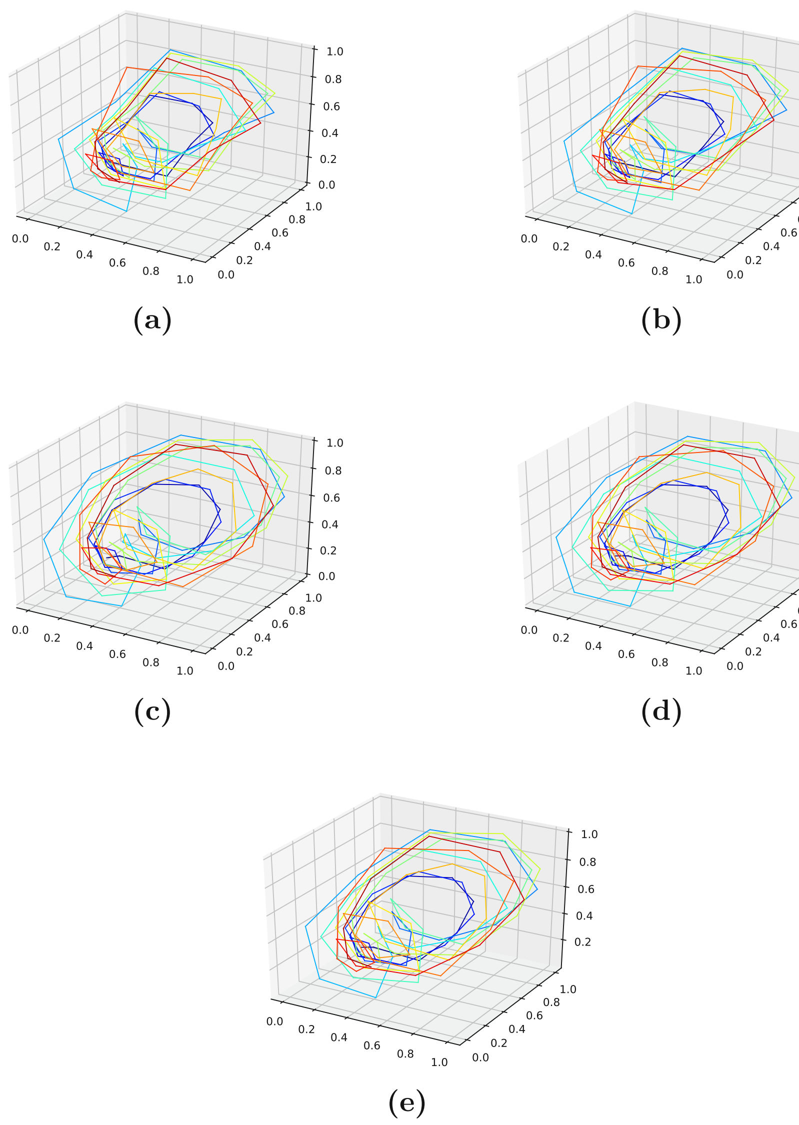
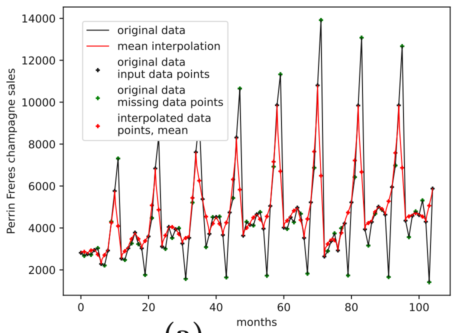
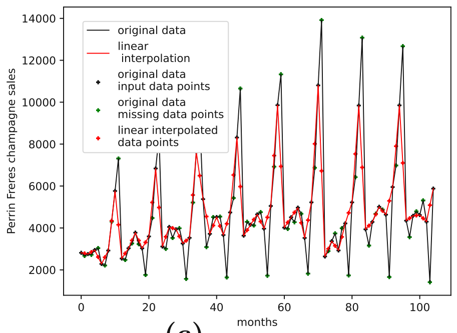


FIGURE C.5: Reconstructed validation attractors (one interpolation point) for the car sales in Quebec data set.
(a): Reconstructed attractor, average population validation interpolation;
(b): Reconstructed attractor, linear interpolation;
(c): Reconstructed attractor, spline interpolation;
(d): Reconstructed attractor, best random validation interpolation;
(e): Reconstructed attractor, gen. alg. improved validation interpolation;

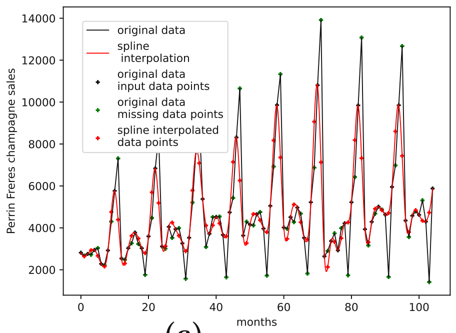
C.1.4 Perrin Freres Champagne Sales



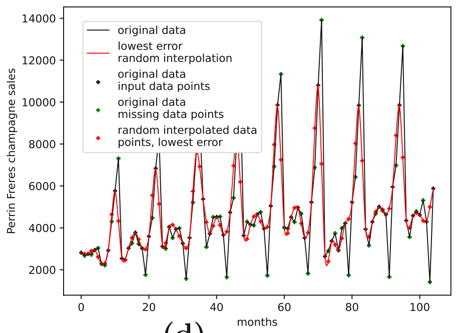
(a)



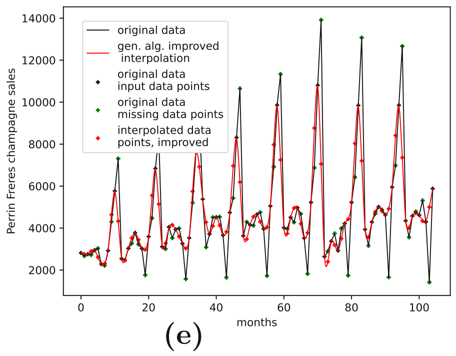
(c)



(c)

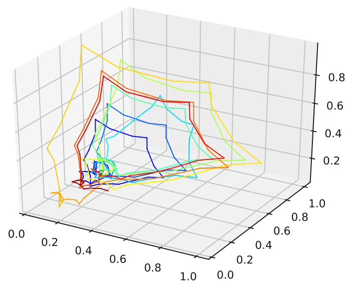


(d)

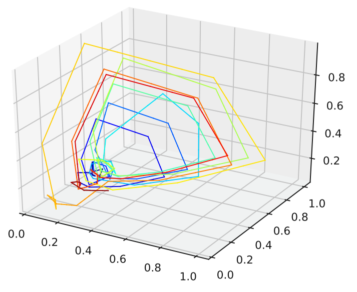


(e)

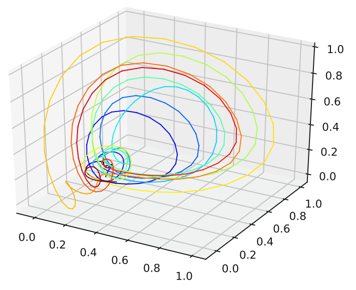
FIGURE C.6: Interpolated validation data (seven interpolation points) for the Perrin Freres champagne sales data set.
(a): Average population validation;
(b): Validation, linear interpolation;
(c): Validation, spline interpolation;
(d): Validation, best random interpolation;
(e): Validation, gen. alg. improved interpolation;



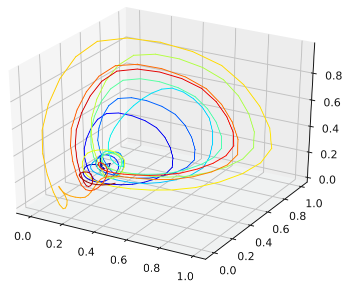
(a)



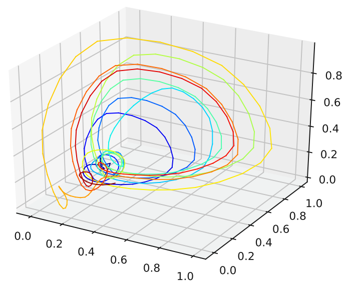
(b)



(c)



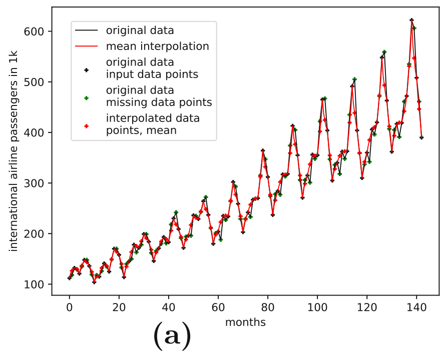
(d)



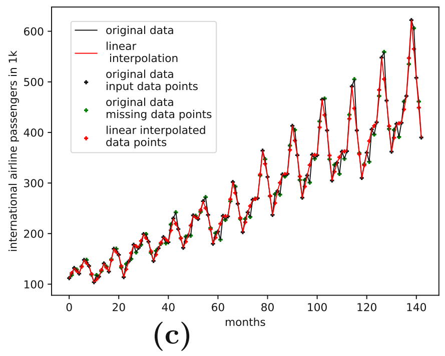
(e)

FIGURE C.7: Reconstructed validation attractors (seven interpolation points) for the Perrin Freres champagne sales data set.
(a): Reconstructed attractor, average population validation interpolation;
(b): Reconstructed attractor, linear interpolation;
(c): Reconstructed attractor, spline interpolation;
(d): Reconstructed attractor, best random validation interpolation;
(e): Reconstructed attractor, gen. alg. improved validation interpolation;

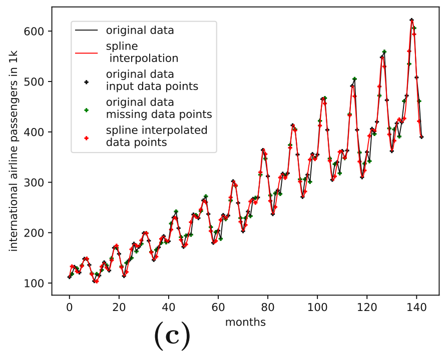
C.1.5 Monthly Airline Passengers



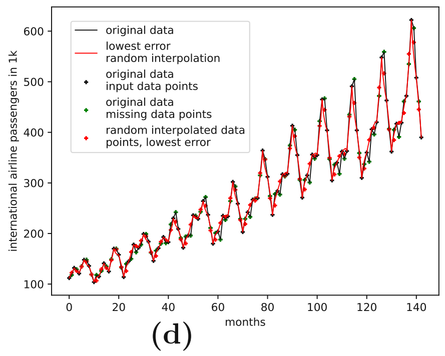
(a)



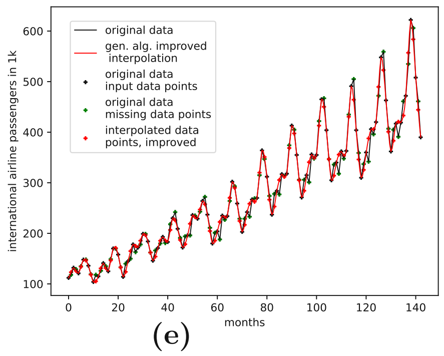
(c)



(c)

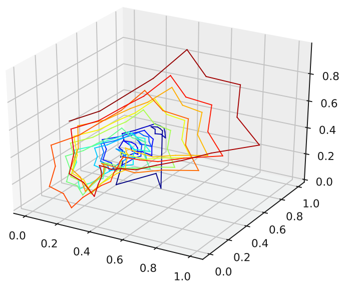


(d)

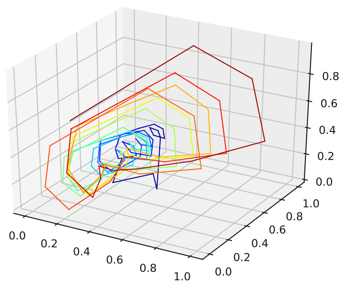


(e)

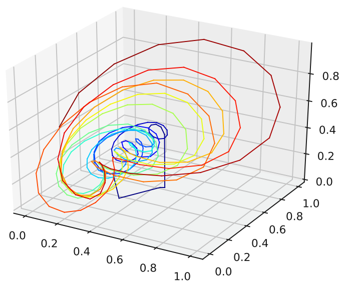
FIGURE C.8: Interpolated validation data (three interpolation points) for the monthly international airline passengers data set.
(a): Average population validation;
(b): Validation, linear interpolation;
(c): Validation, spline interpolation;
(d): Validation, best random interpolation;
(e): Validation, gen. alg. improved interpolation;



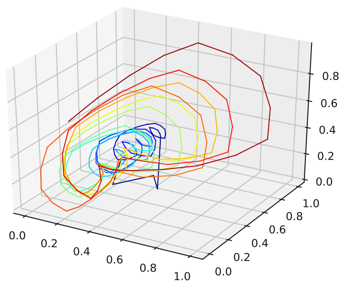
(a)



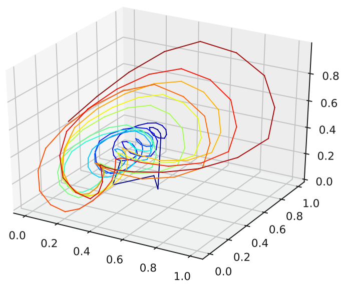
(b)



(c)



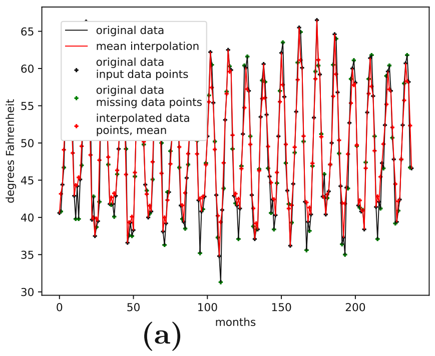
(d)



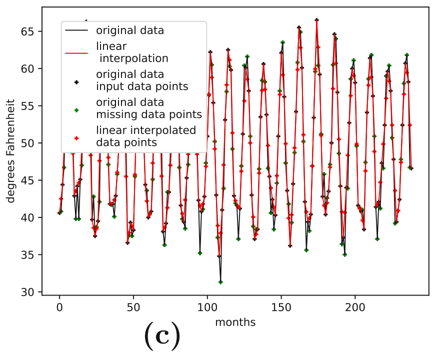
(e)

FIGURE C.9: Reconstructed validation attractors (three interpolation points) for the monthly international airline passengers data set.
(a): Reconstructed attractor, average population validation interpolation;
(b): Reconstructed attractor, linear interpolation;
(c): Reconstructed attractor, spline interpolation;
(d): Reconstructed attractor, best random validation interpolation;
(e): Reconstructed attractor, gen. alg. improved validation interpolation;

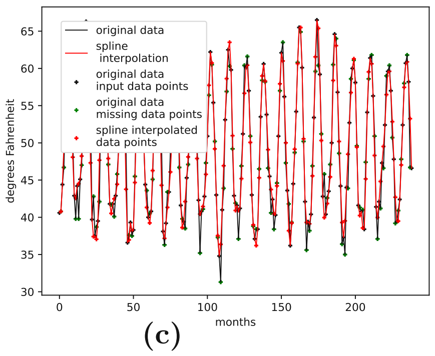
C.1.6 Monthly Mean Temperature in Nottingham Castle



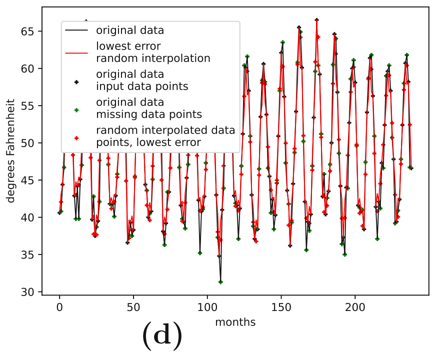
(a)



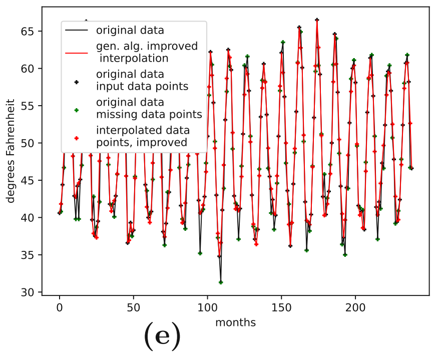
(c)



(c)

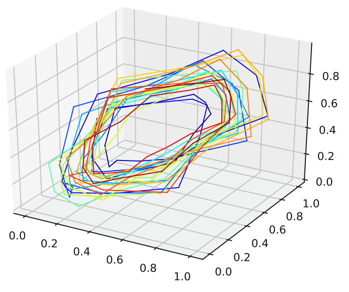


(d)

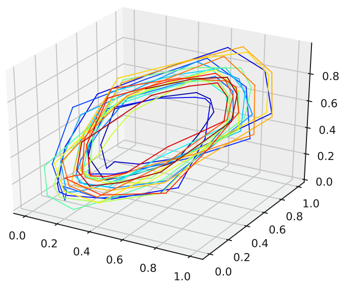


(e)

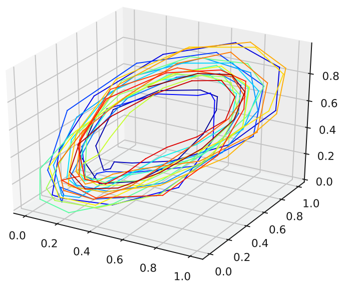
FIGURE C.10: Interpolated validation data (one interpolation point) for the monthly mean temperature in Nottingham castle data set.
(a): Average population validation;
(b): Validation, linear interpolation;
(c): Validation, spline interpolation;
(d): Validation, best random interpolation;
(e): Validation, gen. alg. improved interpolation;



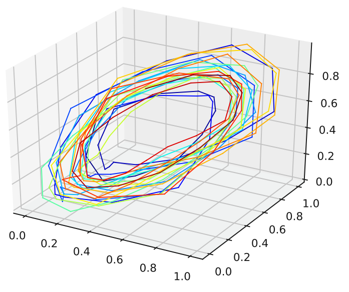
(a)



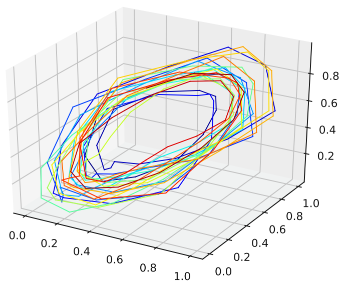
(b)



(c)



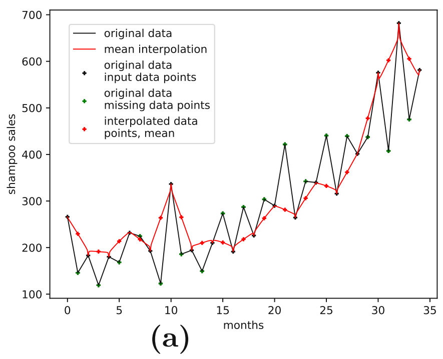
(d)



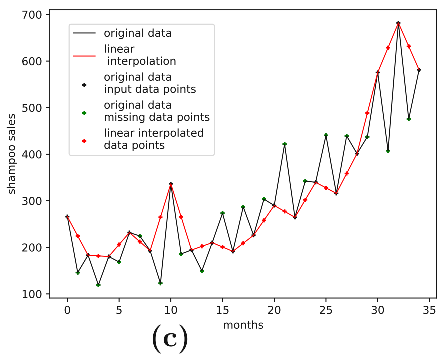
(e)

FIGURE C.11: Reconstructed validation attractors (one interpolation point)for the monthly mean temperature in Nottingham castle data set.
(a): Reconstructed attractor, average population validation interpolation;
(b): Reconstructed attractor, linear interpolation;
(c): Reconstructed attractor, spline interpolation;
(d): Reconstructed attractor, best random validation interpolation;
(e): Reconstructed attractor, gen. alg. improved validation interpolation;

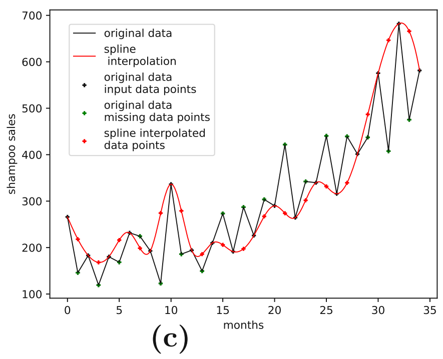
C.1.7 Shampoo Sales



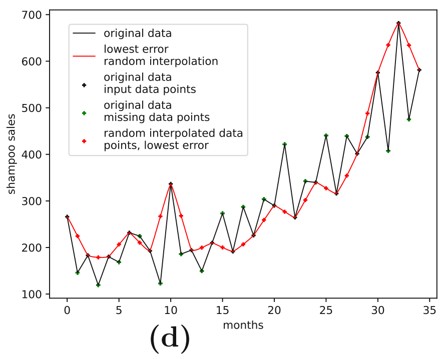
(a)



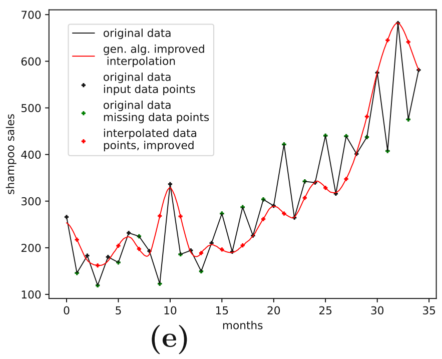
(c)



(c)

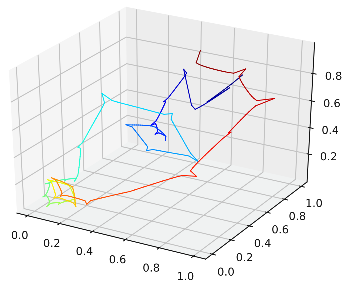


(d)

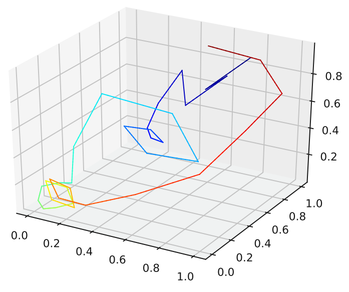


(e)

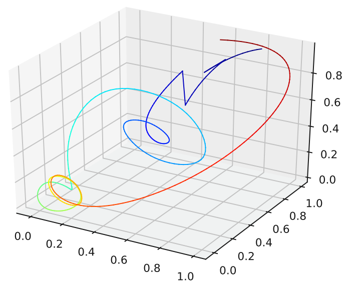
FIGURE C.12: Interpolated validation data (one interpolation point) for the monthly mean temperature in Nottingham castle data set.
(a): Average population validation;
(b): Validation, linear interpolation;
(c): Validation, spline interpolation;
(d): Validation, best random interpolation;
(e): Validation, gen. alg. improved interpolation;



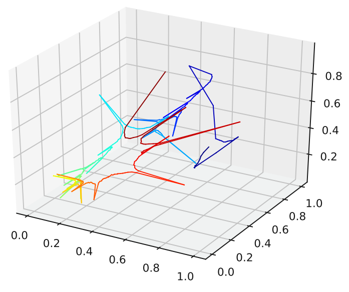
(a)



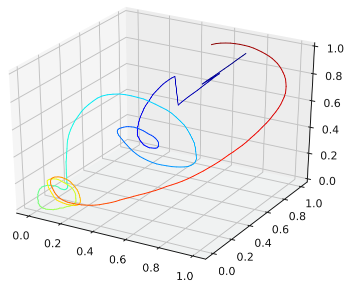
(b)



(c)



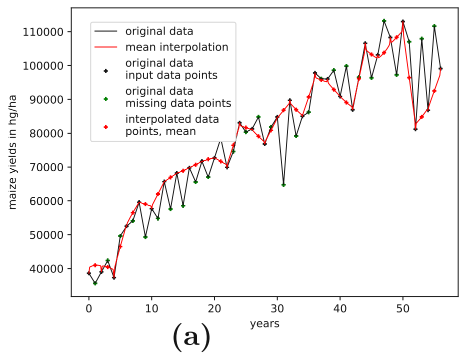
(d)



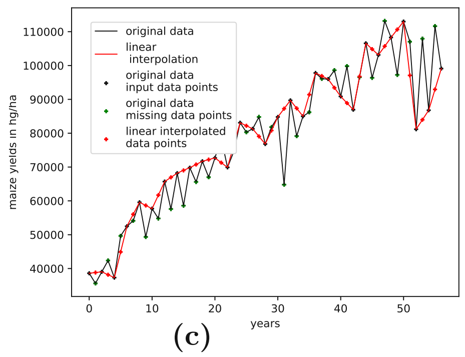
(e)

FIGURE C.13: Reconstructed validation attractors (one interpolation point)for the monthly mean temperature in Nottingham castle data set.
(a): Reconstructed attractor, average population validation interpolation;
(b): Reconstructed attractor, linear interpolation;
(c): Reconstructed attractor, spline interpolation;
(d): Reconstructed attractor, best random validation interpolation;
(e): Reconstructed attractor, gen. alg. improved validation interpolation;

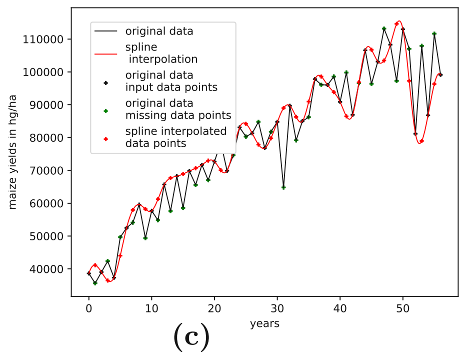
C.1.8 Annual Maize Yields



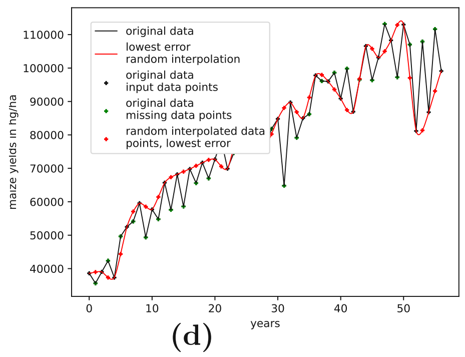
(a)



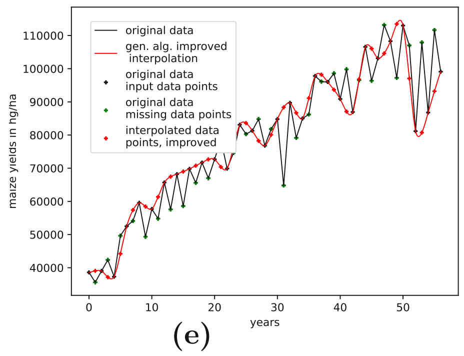
(c)



(c)

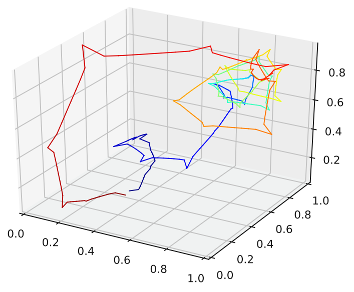


(d)

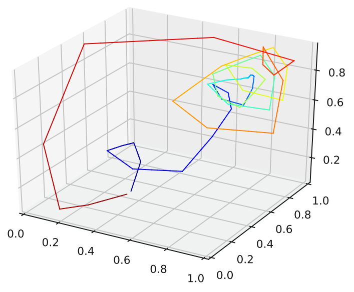


(e)

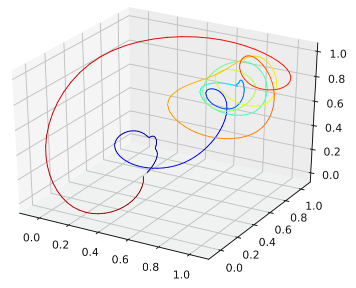
Figure C.14: Interpolated validation data (one interpolation point) for the monthly mean temperature in Nottingham castle data set.
(a): Average population validation;
(b): Validation, linear interpolation;
(c): Validation, spline interpolation;
(d): Validation, best random interpolation;
(e): Validation, gen. alg. improved interpolation;



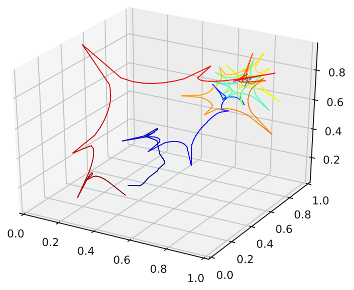
(a)



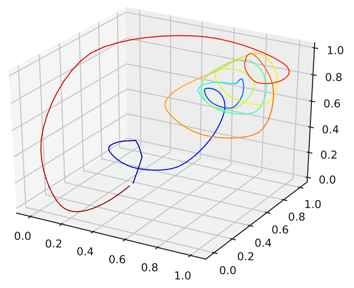
(b)



(c)



(d)



(e)

FIGURE C.15: Reconstructed validation attractors (one interpolation point)for the monthly mean temperature in Nottingham castle data set.
(a): Reconstructed attractor, average population validation interpolation;
(b): Reconstructed attractor, linear interpolation;
(c): Reconstructed attractor, spline interpolation;
(d): Reconstructed attractor, best random validation interpolation;
(e): Reconstructed attractor, gen. alg. improved validation interpolation;

C.2 Failed Attempts

This section provides additional material for failed attempts to find a smooth phase space trajectory. For this reason, we provide additional plots (Figure C.16) and the corresponding errors for the Lorenz system in Table C.1. These attempts for different loss functions include:

- Minimizing the nearest neighbour distance between phase space points.
- Minimizing the mean of first-order derivatives along the phase space trajectory.
- Minimizing the variance of first-order derivatives along the phase space trajectory.
- Minimizing the mean of second order derivatives along the phase space trajectory.

Table C.1: Errors for the interpolated data on the Lorenz system for 14 interpolation points and different loss functions. The Errors are shown for the mean interpolation of all populations, the lowest error in the population, and the interpolation that was improved using the presented genetic algorithm. Further, we give the percentage of how much of the population is outperformed by the genetic algorithm improved interpolation. Here, one can see that only methods including the second derivatives performed well. Further, the variance of second-order derivatives along the phase space trajectory performed best.

Loss Function →	**Nearest Neighbour Distance**	**First Derivative Mean**	**First Derivative Variance**	**Second Derivative Mean**	**Second Derivative Variance**
RMSE Population Mean	0.90686				
Lowest RMSE in population	0.18632				
RMSE gen. alg. improved	1.13779	0.67649	0.54291	0.19274	**0.18626**
Below Best %	73.9%	62.9%	55.5	4.2%	**0.1%**

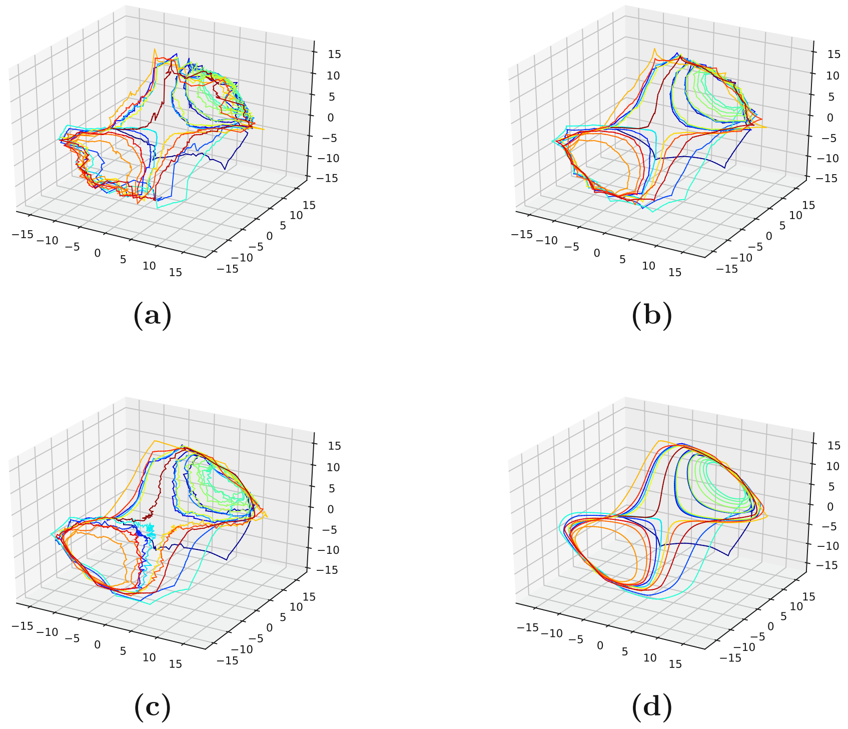
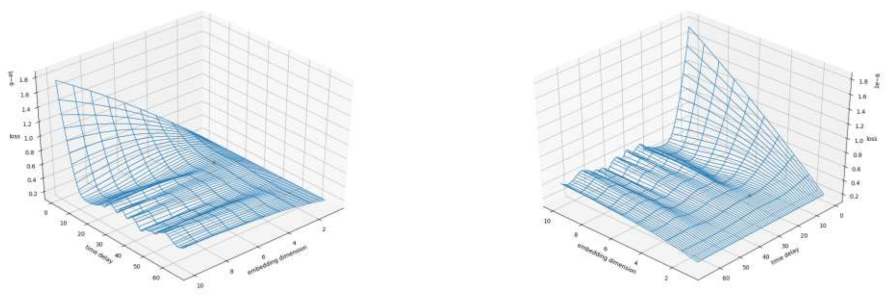


FIGURE C.16: Reconstructed attractors for the interpolated Lorenz system for different loss functions.
(a): Nearest neighbour distance loss function;
(b): First derivative mean loss function;
(c): First derivative variance loss function;
(d): second derivative mean loss function;

C.3 Loss Surface

We present the loss surface for the Lorenz attractor in Figure C.17 from two perspectives. The orange dot marks the actual embedding of the Lorenz system. The plot suggests that the correct phase space embedding is located in an area where the loss surface flattens out. At this point, we did not check for possible ways to locate the correct phase space embedding in the loss surface. Future approaches might find ways to do so.



(a) (b)

FIGURE C.17: Loss surface for the Lorenz attractor.
(a) and **(b)** both show the same surface from different angles. This is the employed loss function (Section 5.2.2) depending on a varying embedding dimension and time delay. The orange dot marks the correct embedding dimension and time delay.

C.4 Dimension and Time Delay Dependence

Our results show that PhaSpaSto interpolation provides results very close to a cubic spline interpolation. However, these results are not exactly a cubic spline interpolation. These interpolations differ slightly, not only from the cubic spline interpolation but also from the actual data points. Further, they also vary depending on the embedding dimension and the time delay. We depict this by the following experiment:

We take an excerpt from the Lorenz system (Section 7.15) with varying data points where we delete data points according to different numbers of interpolation points so that we're left with 100 data points. These "gaps" are then filled using PhaSpaSto-interpolation. We chose varying τ and d_E, where $\tau, d_E \in (1,2,3,4,5,6,7)$. Thus generated 49 different interpolations for the same data set. Additionally, we chose different numbers of interpolation points to verify that our findings do not hold for only one case, i.e., $N_I \in \{11,13,15,17,19,21\}$. Finally, PhaSpaSto-interpolation was performed such that we do not consider the artifacts at the end of each phase space embedding calculation and evaluation, i.e., where the time series is periodically wrapped around, and deactivated mutation, to see if different embeddings choose different interpolations.

First, as depicted in Figure C.18 we see that the original data (black dashed line) and the cubic spline interpolation differ both differ from all PhaSpaSto-interpolation (the remaining colorful lines). Further, we see that all the PhaSpaSto-interpolated data sets also vary slightly. However, we cannot conclude how the different interpolations differ based on this plot. To show how these interpolations differ from the original data set and the employed cubic spline interpolation, we plotted the evolution of errors depending on a varying embedding dimension, depicted in Figure C.19 for different time delays. We see that, though the cubic spline interpolation performs best, there is a dependence on the embedding dimension for different time delays. Further magnifying this makes this dependence obvious, which is on the right side of Figure C.19, i.e., the RMSE increases with increasing embedding dimension. Finally, we also calculated SVD entropy for different interpolations, showing a decreasing entropy with increasing embedding dimension for all tested time delays, which is depicted in Figure C.20.

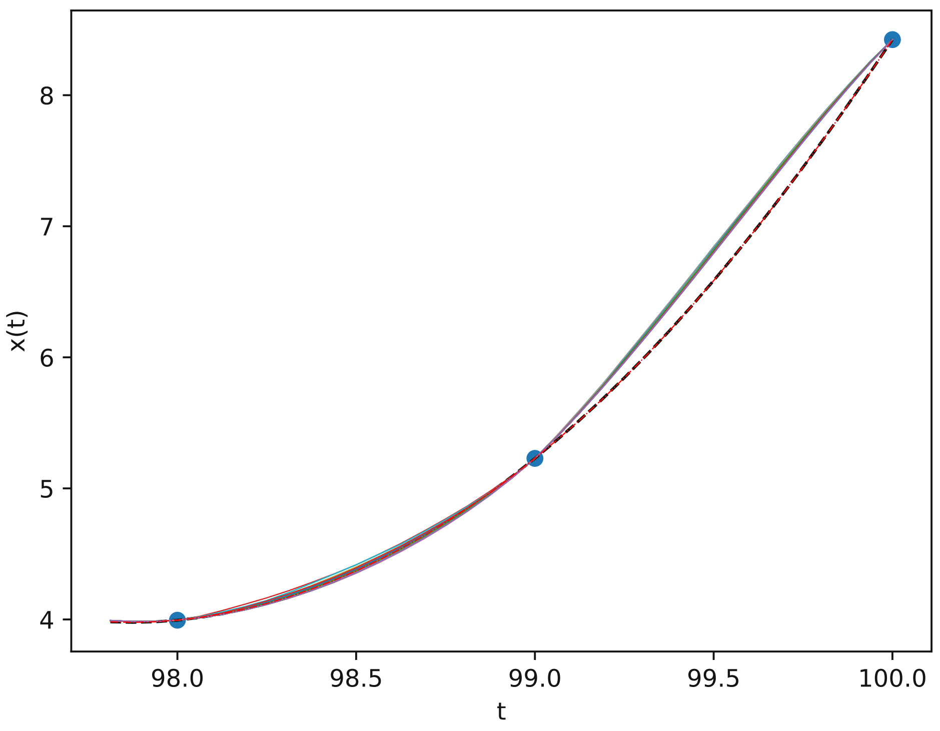


FIGURE C.18: Excerpt from the Lorenz system to show the difference between the cubic spline and PhaSpaSto-interpolation; 15 interpolation points; The black dashed line is the original data set, the red dot-dashed line is the cubic spline interpolation, and the multitude of multi-colored lines depicts the 49 different PhaSpaSto-interpolations. These lines are very close to each other, so depending on the resolution of this thesis, they might give a thick purple line.

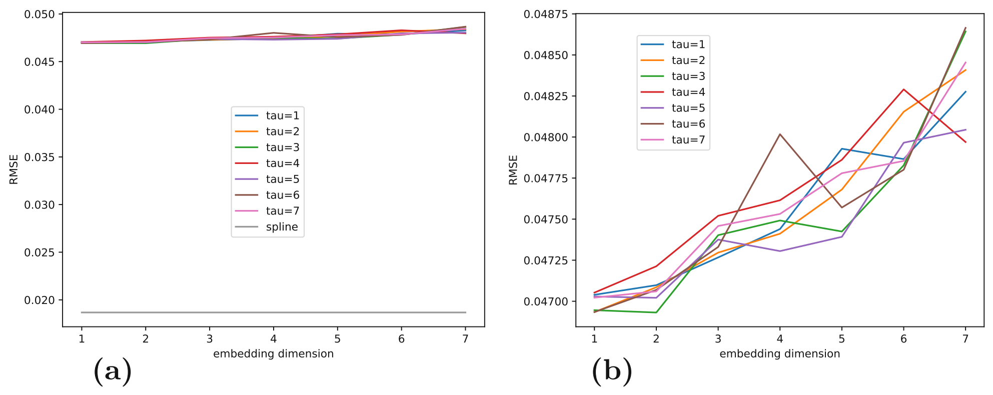


FIGURE C.19: RMSE for varying time delay and embedding dimension for 15 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation as a baseline. This baseline does not depend on whether time delay, or embedding dimension and was plotted as a constant for reference; tau denotes τ

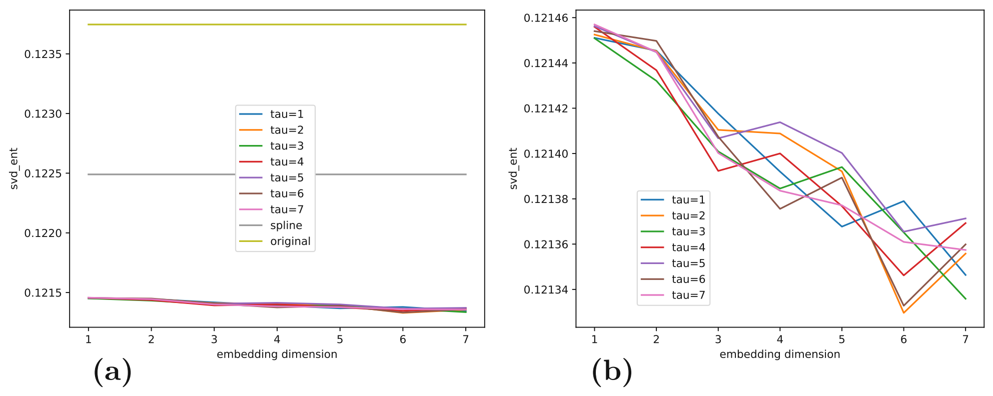


FIGURE C.20: SVD entropy for varying time delay and embedding dimension for 15 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation and the original data set as baselines. These baselines do not depend on time delay, or embedding dimension and were plotted as constants for reference; tau denotes τ

We thus conclude that, though PhaSpaSto-interpolation provides results close to a cubic spline interpolation, there is a small difference in these interpolations. Namely, the cubic spline is closer to the original time series, and PhaSpaSto provides a *sharper* interpolation as the curvature around peak points increases. Further, though different PhaSpaSto-interpolations are very close to each other, they slightly depend on time delay and embedding dimension. In the author's opinion, this won't matter for most use cases, but given the sensitivity of chaotic systems to initial conditions, one might come up with an example where these differences matter.

The plots for all other interpolation points are plotted in the following Figures C.21-C.30.

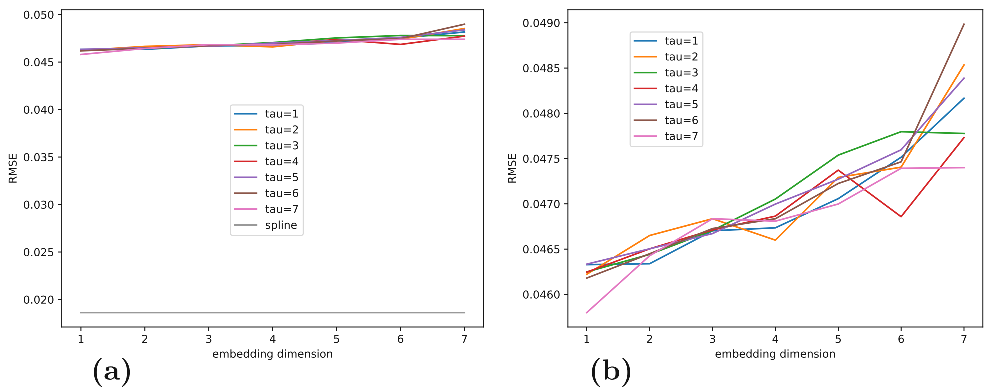


FIGURE C.21: RMSE for varying time delay and embedding dimension for 11 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation as a baseline. This baseline does not depend on time delay or embedding dimension and is plotted as a constant for reference; tau denotes τ

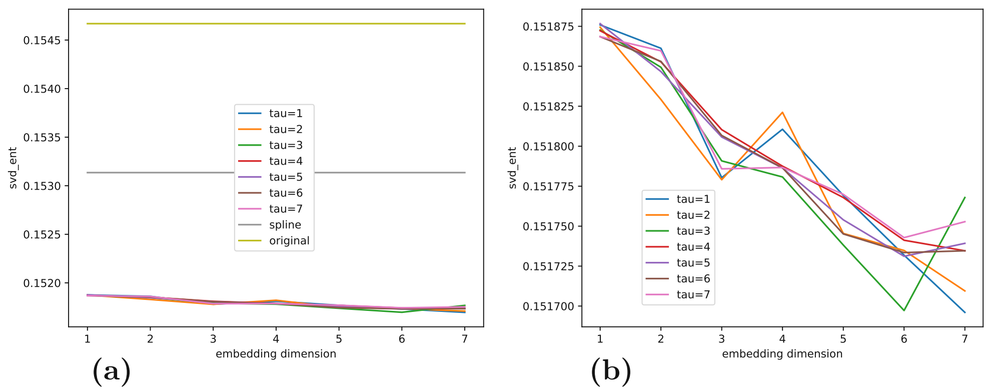


FIGURE C.22: SVD entropy for varying time delay and embedding dimension for 11 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation and the original data set as baselines. These baselines do not depend on time delay or embedding dimension and are plotted as constants for reference; tau denotes τ

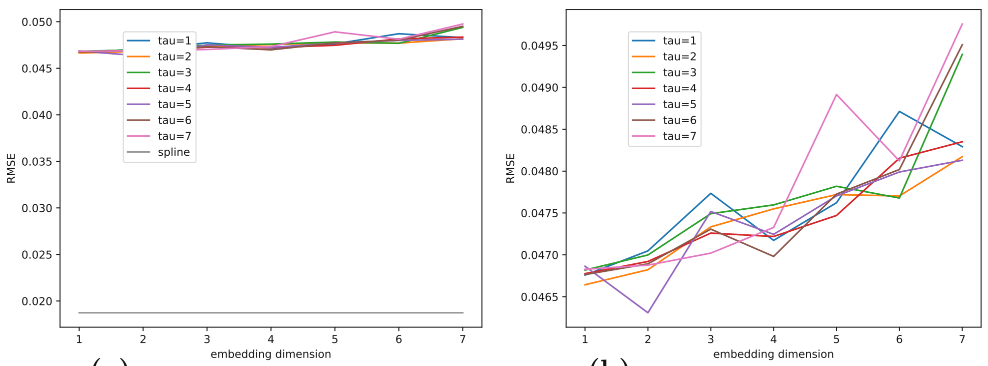

(a) (b)

FIGURE C.23: RMSE for varying time delay and embedding dimension for 13 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation as a baseline. This baseline does not depend on time delay or embedding dimension and is plotted as a constant for reference; tau denotes τ

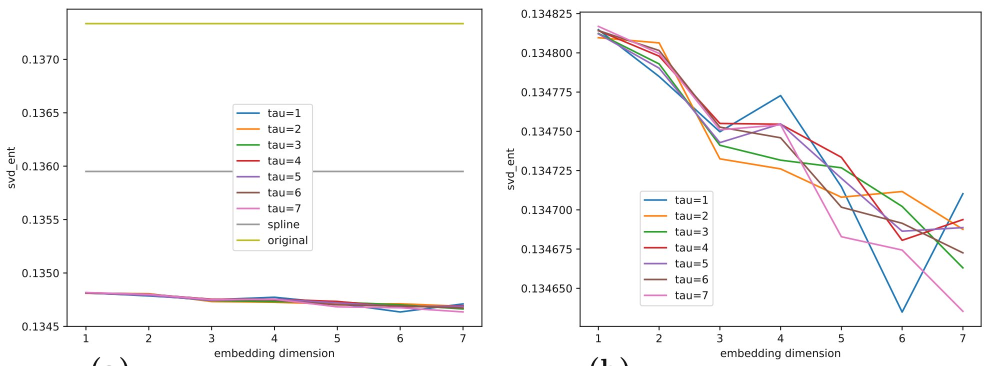

(a) (b)

FIGURE C.24: SVD entropy for varying time delay and embedding dimension for 13 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation and the original data set as baselines. These baselines do not depend on time delay or embedding dimension and are plotted as constants for reference; tau denotes τ

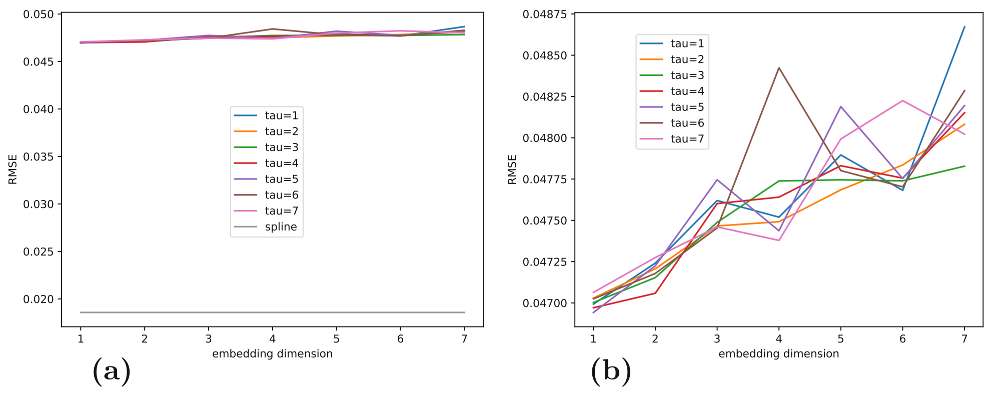


FIGURE C.25: RMSE for varying time delay and embedding dimension for 17 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolations but **(a)** also shows cubic spline interpolation as a baseline. This baseline does not depend on time delay or embedding dimension and is plotted as a constant for reference; tau denotes τ

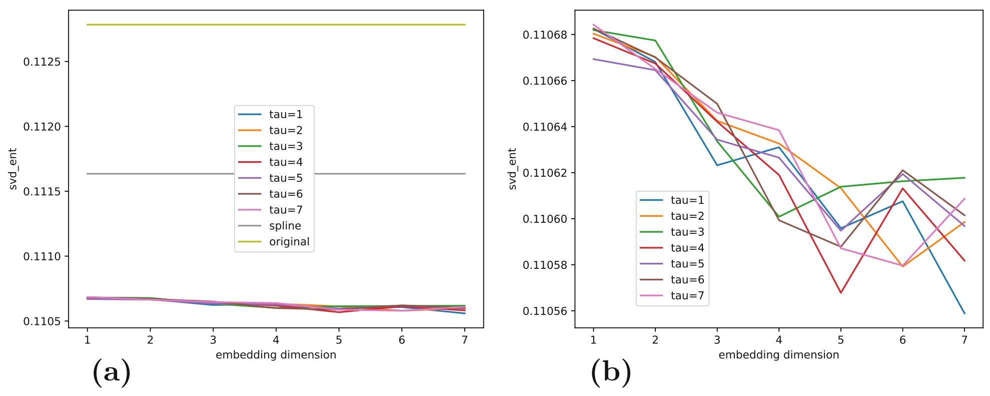
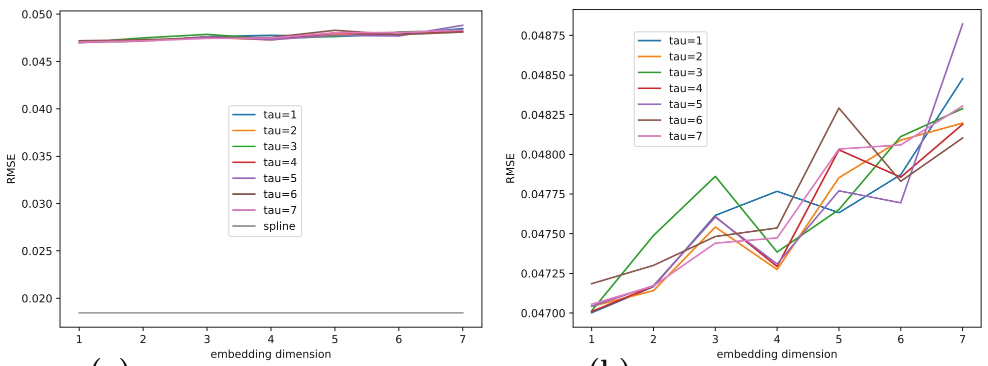
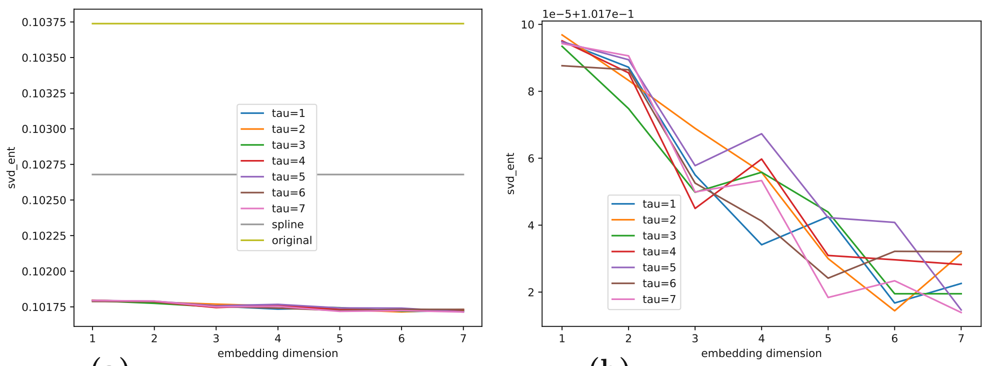


FIGURE C.26: SVD entropy for varying time delay and embedding dimension for 17 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation and the original data set as baselines. These baselines do not depend on time delay or embedding dimension and are plotted as constants for reference; tau denotes τ



(a) **(b)**

FIGURE C.27: RMSE for varying time delay and embedding dimension for 19 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation as a baseline. This baseline does not depend on time delay or embedding dimension and is plotted as a constant for reference; tau denotes τ



(a) **(b)**

FIGURE C.28: SVD entropy for varying time delay and embedding dimension for 19 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation and the original data set as baselines. These baselines do not depend on time delay or embedding dimension and are plotted as constants for reference; tau denotes τ

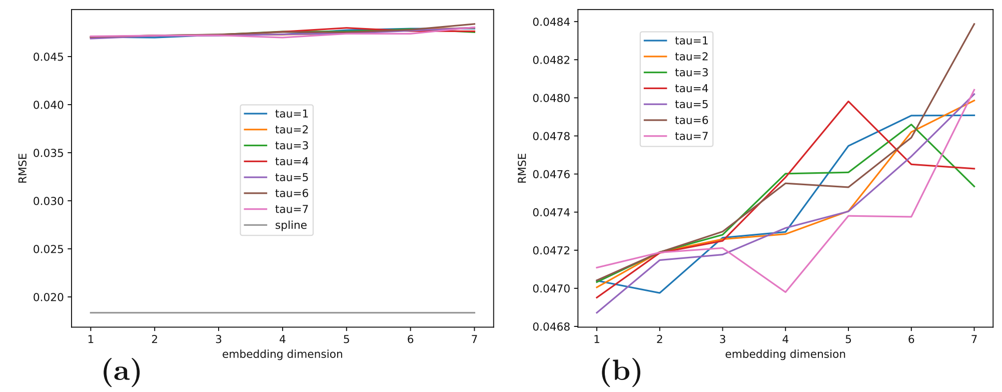


FIGURE C.29: RMSE for varying time delay and embedding dimension for 21 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation as a baseline. This baseline does not depend on time delay or embedding dimension and is plotted as a constant for reference; tau denotes τ.

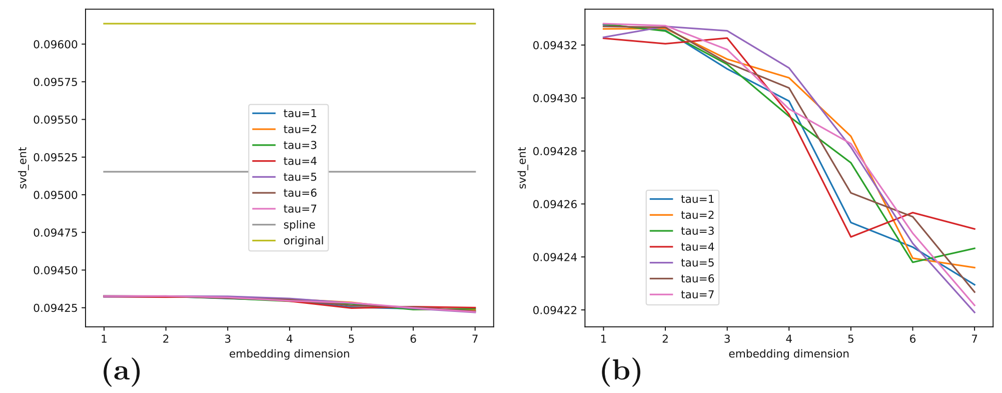


FIGURE C.30: SVD entropy for varying time delay and embedding dimension for 21 interpolation points.
(a) and **(b)** both show the same PhaSpaSto-interpolation but **(a)** also shows cubic spline interpolation and the original data set as baselines. These baselines do not depend on time delay or embedding dimension and are plotted as constants for reference; tau denotes τ

C.5 Computation Time

This appendix shows the differences in computation time between the multi-point fractional Brownian Bridges, PhaSpaSto interpolation, linear interpolation, and the employed cubic spline interpolation from Chapter 10. We experimented with an excerpt from the Lorenz system with 100 data points and had this time series interpolated with ten different numbers of interpolation points. We switched off mutation for the genetic algorithm to obtain better comparable results; however, using the genetic algorithm with mutation would, on average, slightly increase the computation time and the number of generations until the algorithm terminates. However, this won't matter much in this context, given the extremely long computation times of PhaSpaSto interpolation compared to the other methods. Apart from that, we used the same setup as in Chapter 10 for generating the population and the genetic algorithm. The results, denoted in Table C.2, show that PhaSpaSto interpolation, i.e., generating the population and running the genetic algorithm, takes on average 3.4×10^6 times longer than linear interpolation and 6.7×10^6 longer than the employed cubic spline interpolation.

TABLE C.2

N__int	**Multi-point frac. Brownian Bridge**	**Generate Population**	**Genetic Algorithm**	**#Generations**	**Linear Interpolation**	**Cubic Spline Interpolation**	**Factor PhaSpaSto Linear Int.**	**Factor PhaSpaSto Spline Int.**
1	00 : 00.242675	04 : 02.675073	09 : 48.721004	118	00 : 00.000301	00 : 00.000136	2.762×10^6	6.113×10^6
2	00 : 00.313547	05 : 13.547199	09 : 41.154012	111	00 : 00.000378	00 : 00.000176	2.367×10^6	5.084×10^6
3	00 : 00.360844	06 : 00.844346	10 : 07.320286	110	00 : 00.000340	00 : 00.000166	2.848×10^6	5.832×10^6
4	00 : 00.433789	07 : 13.789044	10 : 57.340219	111	00 : 00.000327	00 : 00.000165	3.337×10^6	6.613×10^6
5	00 : 00.528226	08 : 48.226845	12 : 46.217807	114	00 : 00.000329	00 : 00.000168	3.934×10^6	7.705×10^6
6	00 : 00.579014	09 : 39.014085	11 : 05.322099	111	00 : 00.000339	00 : 00.000172	3.671×10^6	7.235×10^6
7	00 : 00.689273	11 : 29.273492	10 : 51.783278	107	00 : 00.000447	00 : 00.000231	3×10^6	5.805×10^6
8	00 : 00.762853	12 : 42.856440	12 : 44.076810	107	00 : 00.000390	00 : 00.000218	3.915×10^6	7.004×10^6
9	00 : 00.780448	13 : 00.448667	13 : 17.423635	110	00 : 00.000351	00 : 00.000188	4.495×10^6	8.393×10^6
10	00 : 00.868501	14 : 28.501188	12 : 39.780835	105	00 : 00.000493	00 : 00.000232	3.303×10^6	7.018×10^6

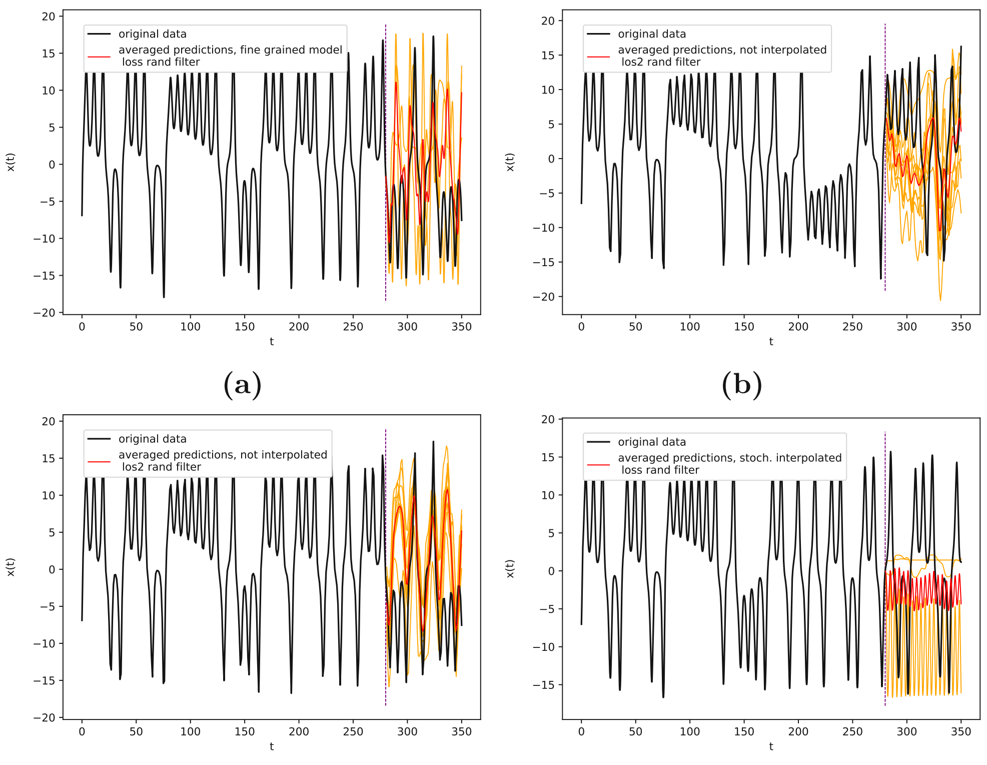
Appendix D

Additional Material Chapter 11

D.1 Additional Plots

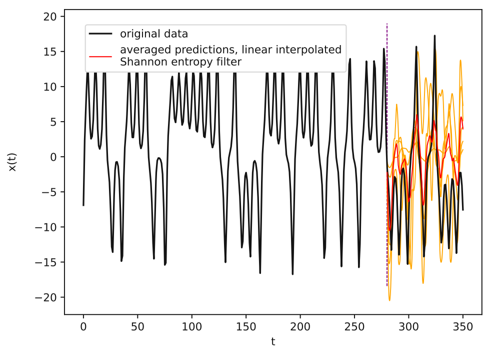
This section collects the additional plots for the experiments conducted in Chapter 11. We provide the remaining plots of the top five predictions for the complexity filters and the filters based on the second derivative along the phase space trajectory.

The Lorenz System

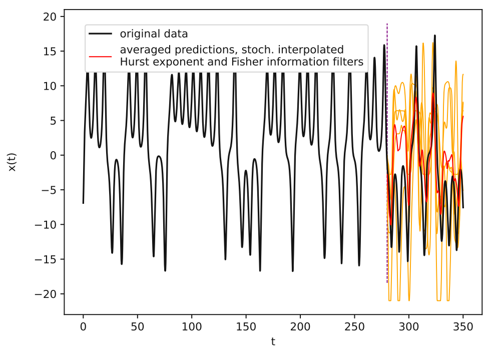


(a) **(b)** **(c)** **(d)**

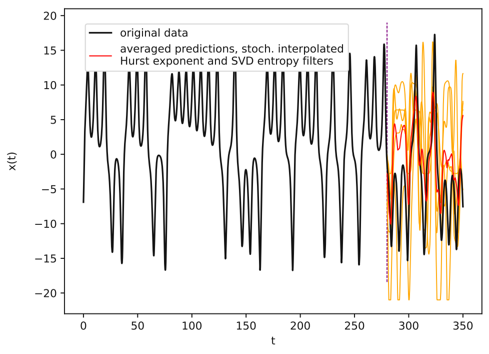
FIGURE D.1: Best predictions for the Lorenz system, phase-space filtered.
(a): loss_rand, fine-grained-model, 13 interpolation points, RMSE=0.23426±0.01975
(b):los2_rand-filter, not inteproalted, RMSE=0.24299±0.01864
(c):los2_rand-filter, not inteprolated, RMSE=0.25770±0.01242
(d): loss_rand-filter, stoch. interpolated, 15 interpolation points, RMSE=0.26656±0.02109



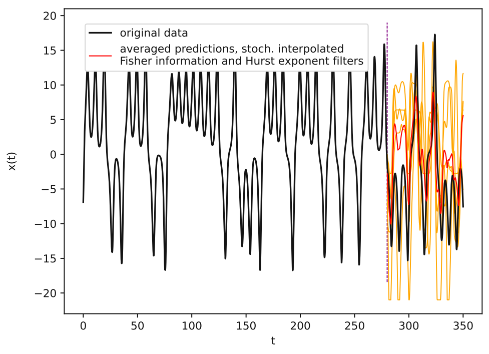
(a)



(b)



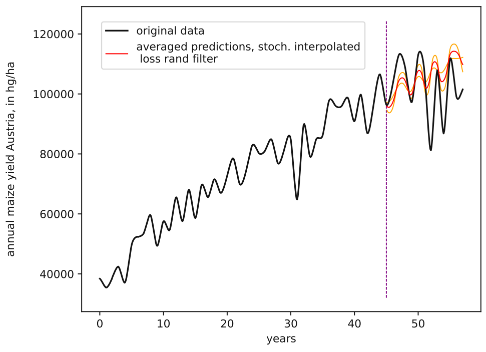
(c)



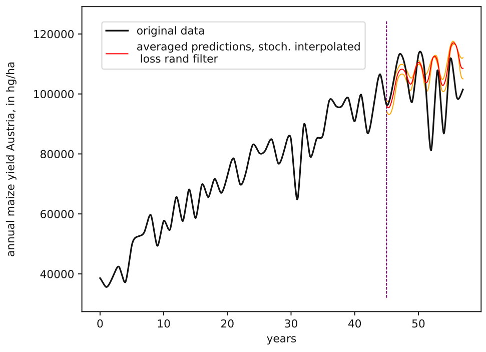
(d)

FIGURE D.2: Best predictions for the Lorenz system, signal-complexity filtered.
(a): Shannon-filter, linear interpolated, 13 interpolation points, RMSE=0.19102±0.01867
(b): Hurst-Fisher-filter, stoch. interpolated, 13 interpolation points, RMSE=0.19122±0.02710
(c): Hurst-SVD-filter, stoch. interpolated, 13 interpolation points, RMSE=0.19122±0.02710
(d): Fisher-Hurst-filter, stoch. interpolated, 13 interpolation points, RMSE=0.19122±0.02710

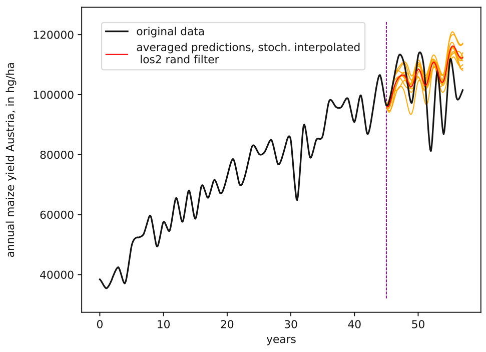
Annual Maize Yields in Austria



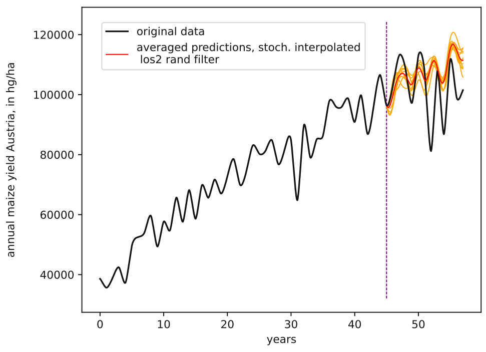
(a)



(b)

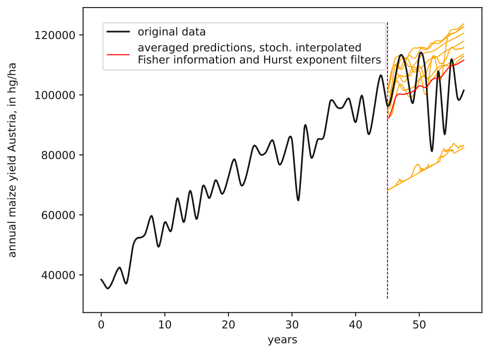


(c)

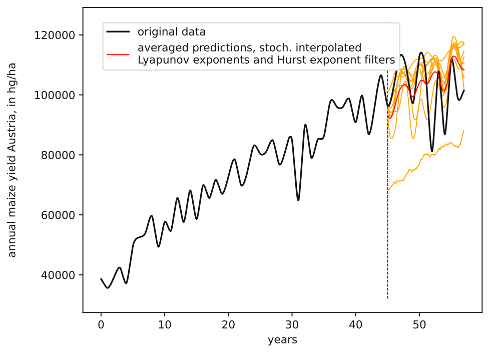


(d)

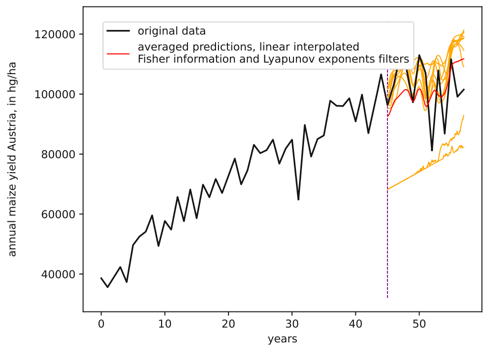
FIGURE D.3: Best predictions for the annual maize yields in Austria data set, phase-space filtered.
(a): loss_rand-filter, stoch. interpolated, 9 interpolation points, RMSE=0.13961±0.01087
(b): loss_rand-filter, stoch. interpolated, 13 interpolation points, RMSE=0.14022±0.00567
(c):los2_rand-filter, stoch. interpolated, 9 interpolation points, RMSE=0.14320±0.00801
(d):los2_rand-filter, stoch. interpolated, 15 interpolation points, RMSE=0.14414±0.00651



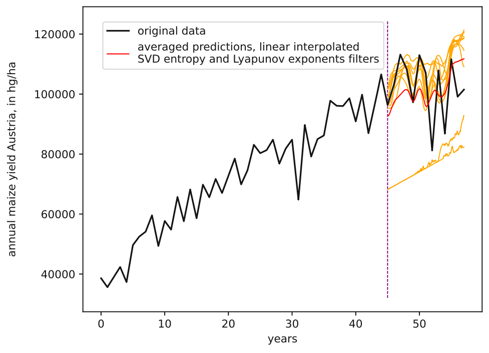
(a)



(b)



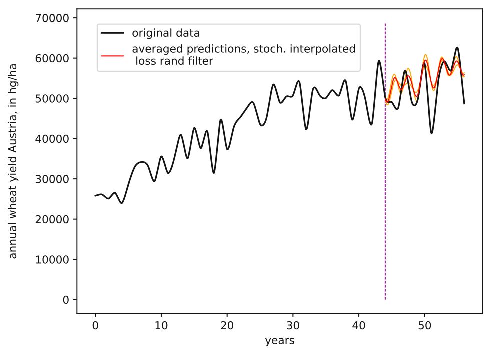
(c)



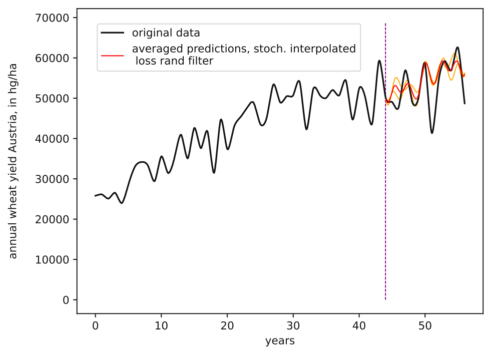
(d)

FIGURE D.4: Best predictions for the annual maize yields in Austria data set, signal-complexity filtered.
(a): Fisher-Hurst-filter, stoch. interpoalted, 11 interpolation points, RMSE=0.13511±0.03642
(b): Lyap-Hurst-filter, stoch. interpolated, 15 interpolation points, RMSE=0.13654±0.03487
(c): Fisher-Lyap-filter, linear interpolated, 15 interpolation points, RMSE=0.13759±0.04690
(d): SVD-Lyap-filter, linear interpolated, 15 interpolation points, RMSE=0.13759±0.04690

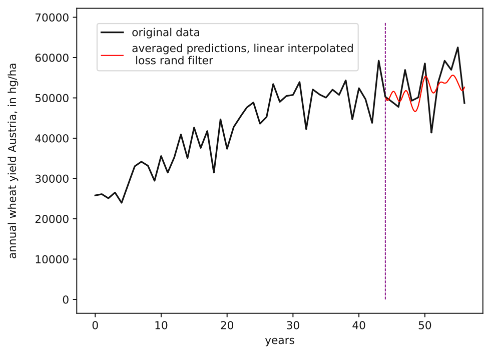
Annual Wheat Yields in Austria



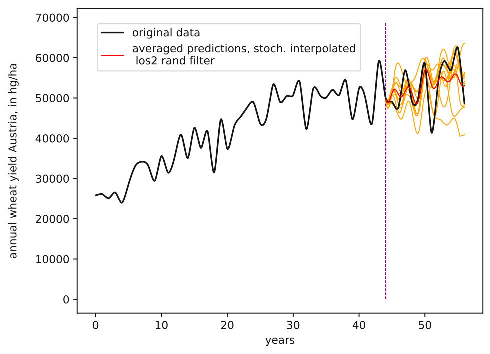
(a)



(b)

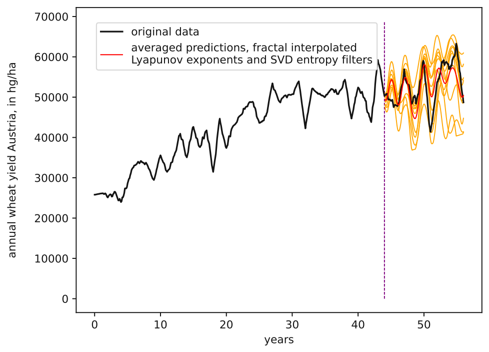


(c)

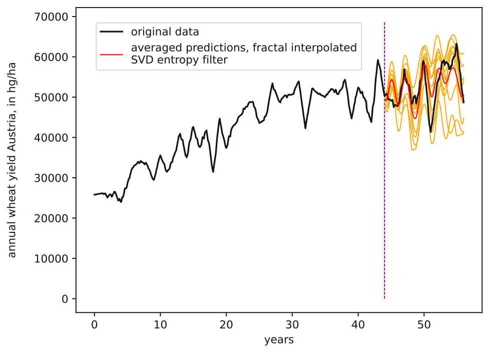


(d)

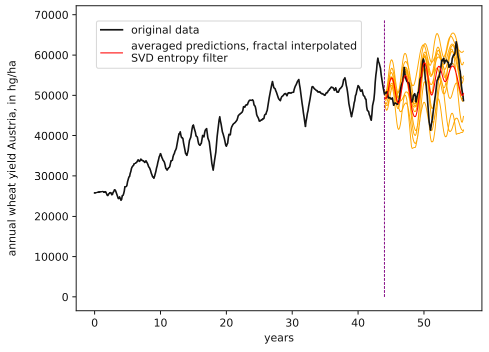
FIGURE D.5: Best predictions for the annual wheat yields in Austria data set, phase-space filtered.
(a): loss_rand-filter, stoch. interpolated, 15 interpolation points, RMSE=0.12246±0.01147
(b): loss_rand-filter, stoch. interpolated, 13 interpolation points, RMSE=0.12544±0.00599
(c): loss_rand-filter, linear interpolated, 13 interpolation points, RMSE=0.12657±0.00000
(d):los2_rand-filter, stoch. interpolated, 13 interpoaltion points, RMSE=0.12662±0.02979



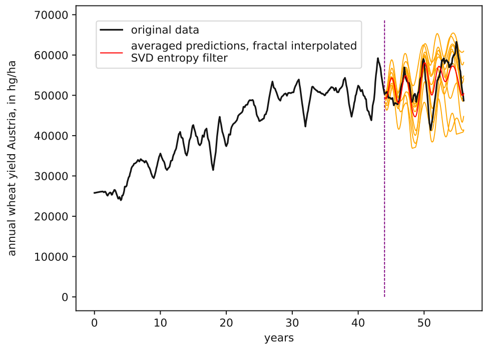
(a)



(b)



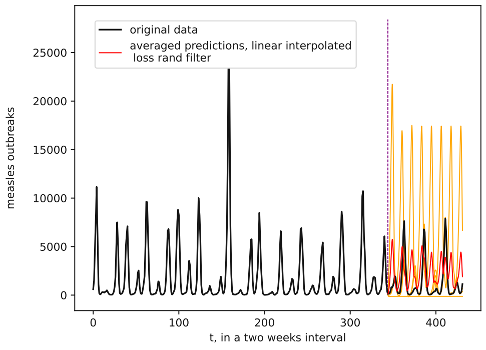
(c)



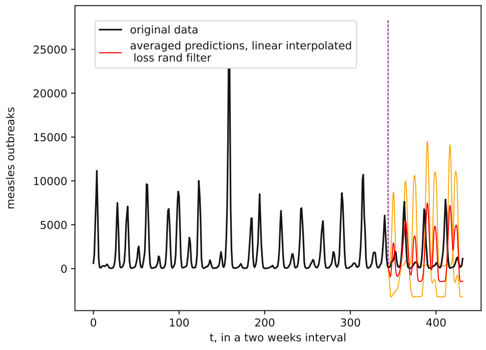
(d)

FIGURE D.6: Best predictions for the annual wheat yields in Austria data set, signal-complexity filtered.
(a): Fisher-SVD-filter, fractal interpolation, 9 interpolation points, RMSE=0.11517±0.04367
(b): SVD-filter, fractal inteprolated, 9 interpolation points, RMSE=0.11517±0.04367
(c): SVD-Shannon-filter, fractal interpoalted, 9 interpolation points, RMSE=0.11517±0.04367
(d): Lyap-Fisher-filter, fractal interpolated, 9 interpolation points, RMSE=0.11517±0.04367

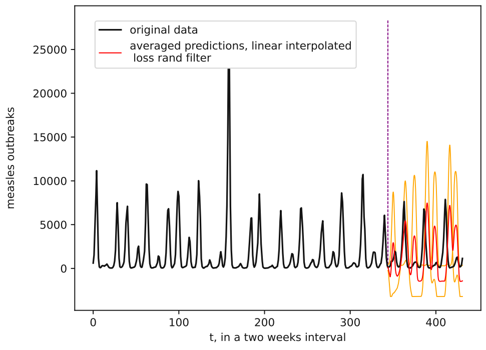
Measles cases in NYC



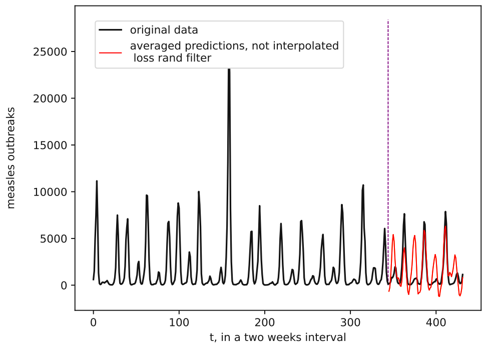
(a)



(b)

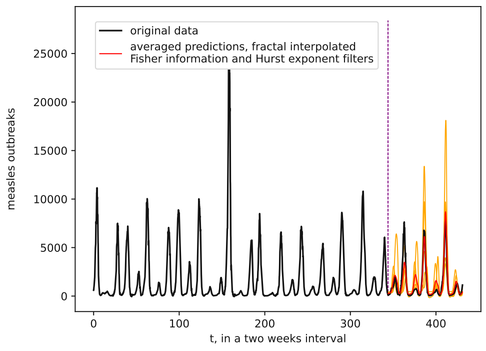


(c)

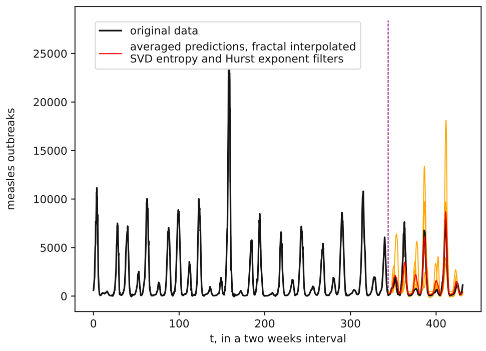


(d)

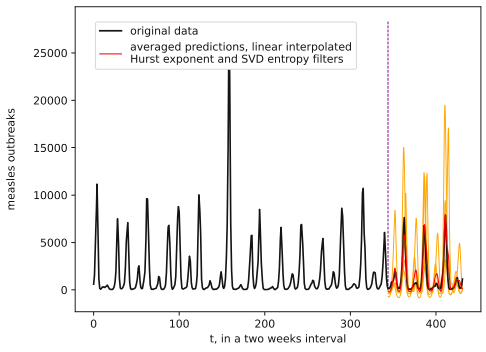
FIGURE D.7: Best predictions for the measles cases in NYC data set, phase space property filtered.
(a): loss_rand-filter, linear interpolated, 11 interpolation points, RMSE=0.05415±0.01072
(b): loss_rand-filter, linear interpolated, 15 interpolation points, RMSE=0.06033±0.00225
(c):los2_rand-filter, fractal interpolated, 9 interpolation points, RMSE=0.06338 ±0.02327
(d): loss_rand-filter, not interpolated, RMSE=0.06445±0.00000



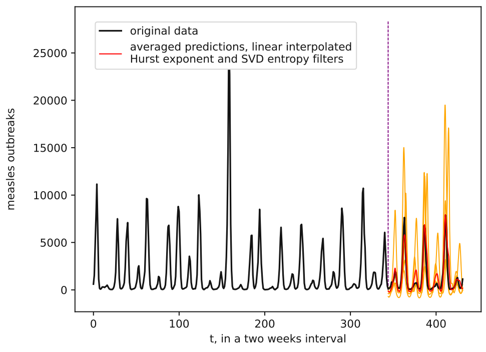
(a)



(b)



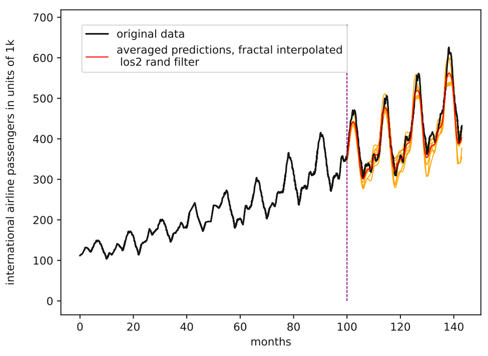
(c)



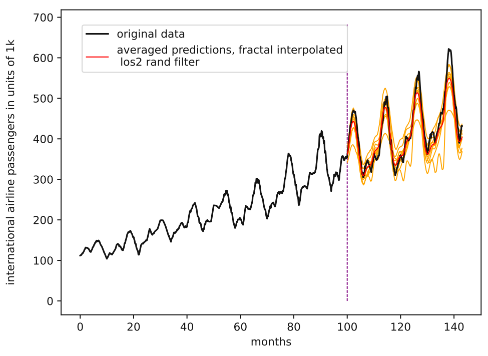
(d)

FIGURE D.8: Best predictions for the measles cases in NYC data set, signal-complexity filtered.
(a): Fisher-Hurst-filter, fractal interpolated, 9 interpolation points, RMSE=0.03242±0.00735
(b): SVD-Hurst-filter, fractal interpolated, 9 interpolation points, RMSE=0.03242±0.00735
(c): Hurst-SVD-filter, linear interpolated, 13 interpolation points, RMSE=0.03248±0.01705
(d): Lyap-Shannon-filter, linear interpolated, 9 interpolation points, RMSE=0.03359±0.01122

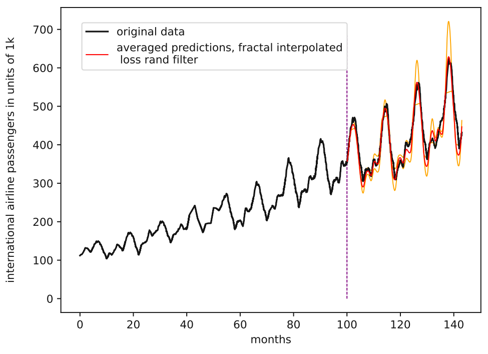
Monthly International Airline Passengers



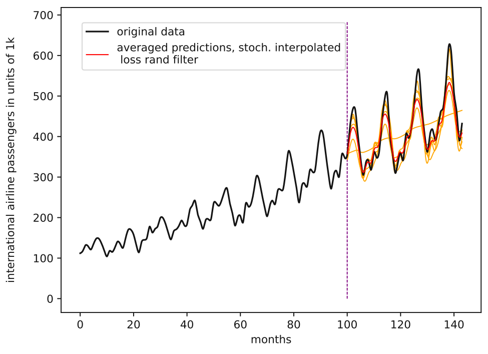
(a)



(b)

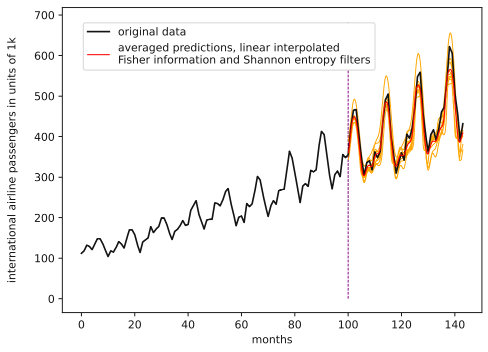


(c)

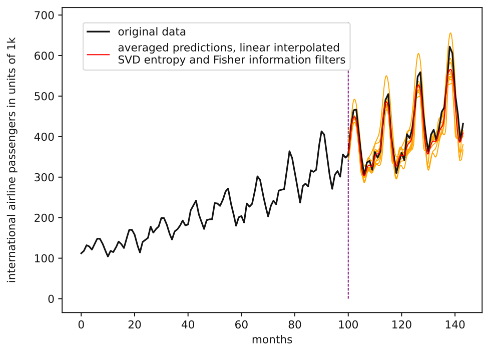


(d)

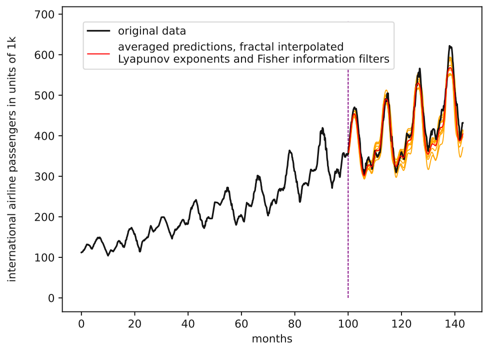
FIGURE D.9: Best predictions for the monthly international airline passengers data set, phase space property filtered.
(a): los2_rand-filter, fractal interpolated, 11 interpolation points, RMSE=0.04995±0.00534
(b): los2_rand-filter, fractal interpolated, 9 interpolation points, RMSE=0.05111±0.00789
(c):loss_rand-filter, fractal interpolated, 13 interpolation points, RMSE=0.06016 ±0.00795
(d): loss_rand-filter, stoch. interpolated, RMSE=0.06021±0.00627



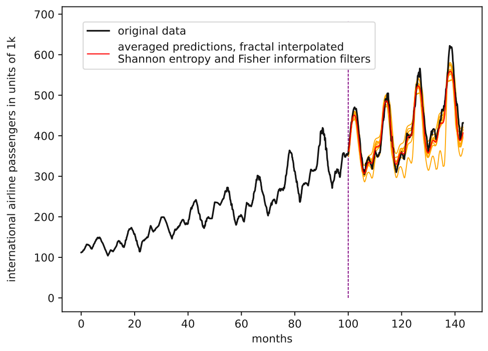
(a)



(b)



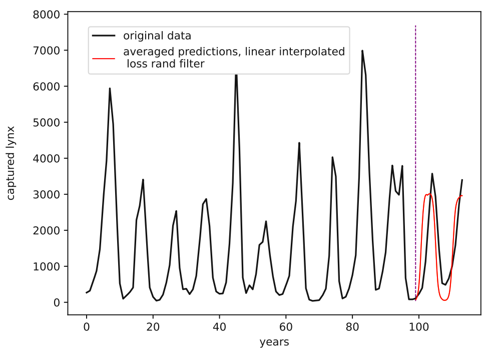
(c)



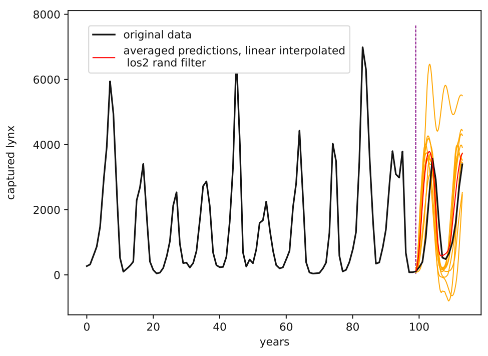
(d)

FIGURE D.10: Best predictions for the monthly international airline passengers data set, signal-complexity filtered.
(a): Fisher-Shannon-filter, linear interpolated, 13 interpolation points, RMSE=0.04450±0.00675
(b): SVD-Fisher-filter, linear interpolated, 13 interpolation points, RMSE=0.04450±0.00675
(c): Lyap-Fisher-filter, fractal interpolated, 9 interpolation points, RMSE=0.04465±0.00422
(d): Shannon-Fisher-filter, fractal interpolated, 9 interpolation points, RMSE=0.04587±0.00476

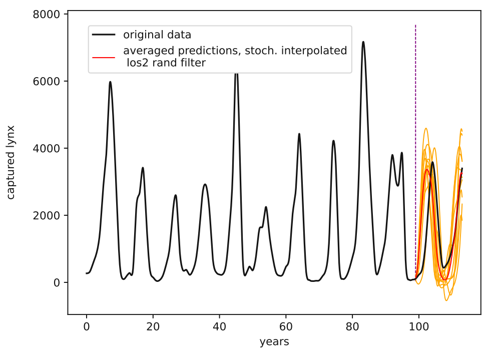
Canadian Lynx



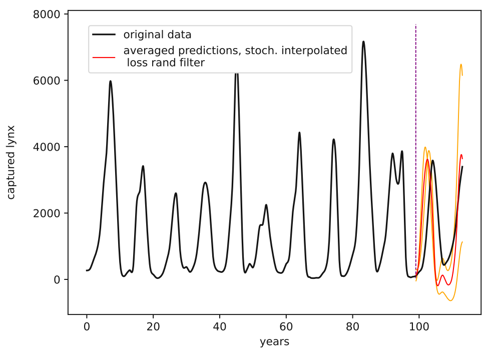
(a)



(b)

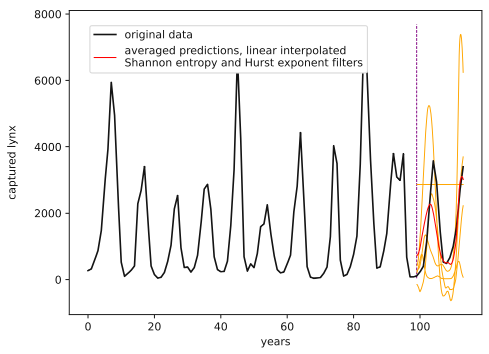


(c)

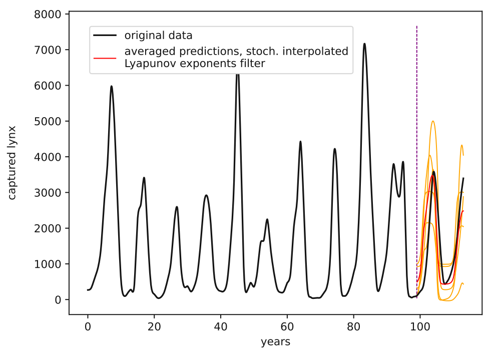


(d)

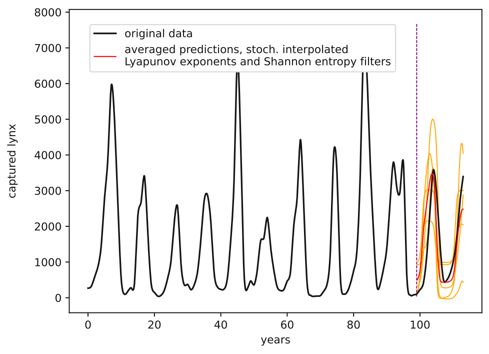
FIGURE D.11: Best predictions for the Canadian Lynx data set, phase-space filtered.
(a): loss_rand-filter, linear interpolated, 11 interpolation points,
RMSE=0.13442±0.00000
(b):los2_rand-filter, linear interpolated, 13 interpolation points,
RMSE=0.14205±0.04017
(c):los2_rand-filter, fractal interpolated, 15 interpolation points,
RMSE=0.14345 ±0.05349
(d): loss_rand-filter, stoch. interpolated, 13 interpolation points,
RMSE=0.14643±0.03443



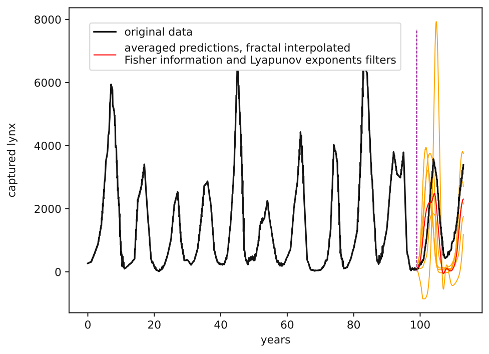
(a)



(b)



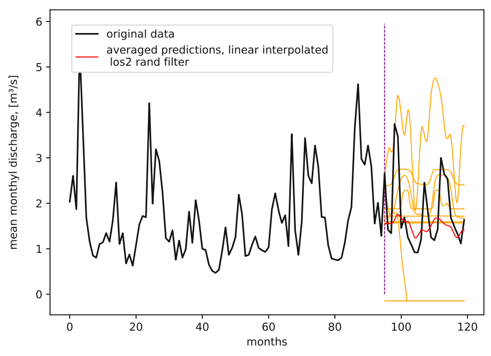
(c)



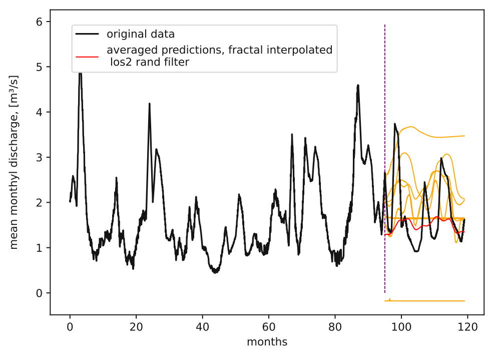
(d)

FIGURE D.12: Best predictions for the Canadian lynx data set, signal-complexity filtered.
(a): Shannon-Hurst-filter, linear interpolated, 9 interpolation points, RMSE=0.11037±0.05037
(b): Lyap-filter, stoch. interpolated, 9 interpolation points, RMSE=0.11405±0.03110
(c): Lyap-Shannon-filter, stoch. interpolated, 9 interpolation points, RMSE=0.11405 ±0.03110
(d): Fisher-Lyap-filter, fractal interpolated, 11 interpolation points, RMSE=0.11574±0.05229

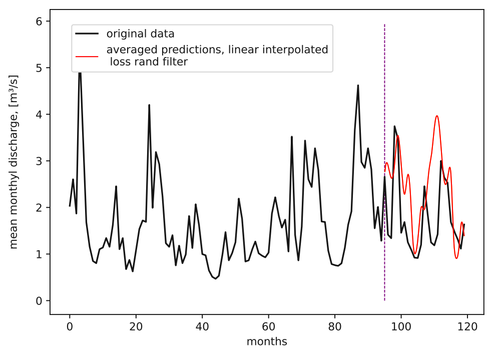
River Krems Discharge



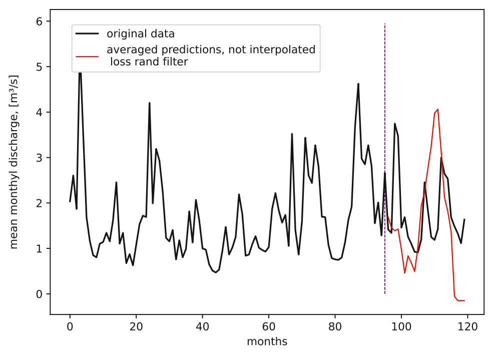
(a)



(b)

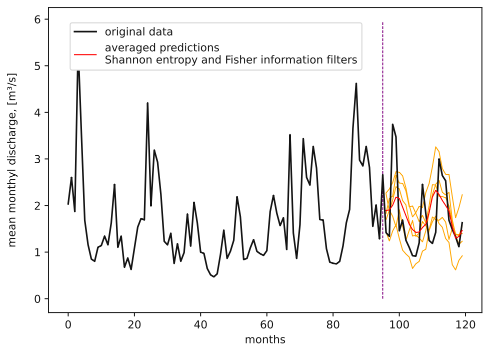


(c)

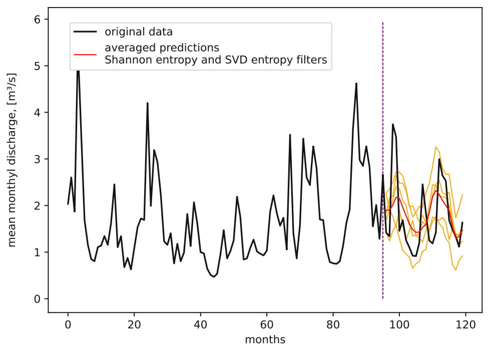


(d)

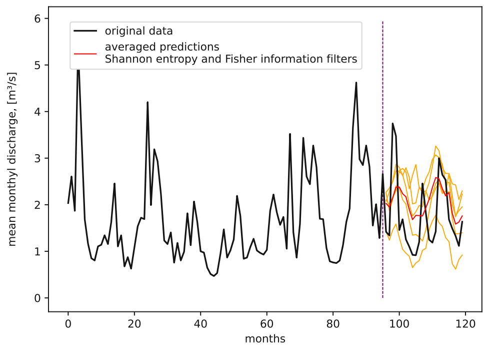
FIGURE D.13: Best predictions for the river Krems discharge data set, phase-space filtered.
(a):los2_rand-filter, linear interpolated, 9 interpolation points, RMSE=0.15492±0.04948
(b):los2_rand-filter, fractal interpolated, 15 interpolation points, RMSE=0.15943±0.05062
(c): loss_rand-filter, linear interpolated, 9 interpolation points, RMSE=0.16651 ±0.01730
(d): loss_rand-filter, not interpolated, RMSE=0.16974±0.00000



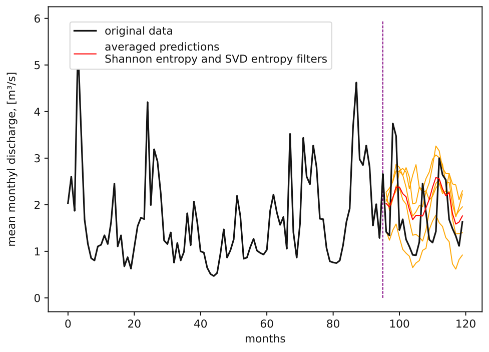
(a)



(b)



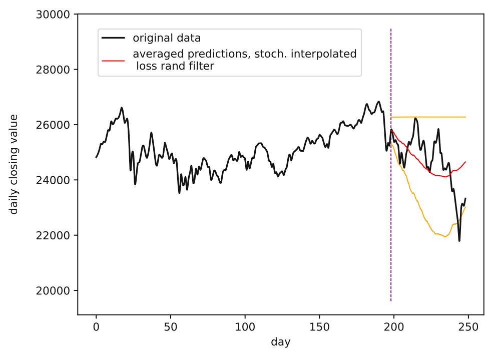
(c)



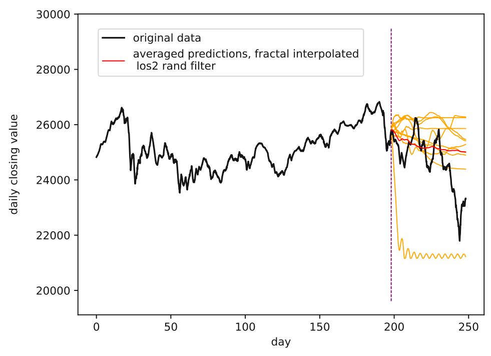
(d)

FIGURE D.14: Best predictions for the river Krems discharge data set, signal-complexity filtered.
(a): Shannon-Fisher-filter, not interpolated, RMSE=0.13885±0.01845
(b): Shannon-SVD-filter, not interpolated, RMSE=0.13885±0.01845
(c): Shannon-Fisher-filter, not interpolated, RMSE=0.13900 ±0.01839
(d): Shannon-SVD-filter, not interpolated, RMSE=0.13900±0.01839

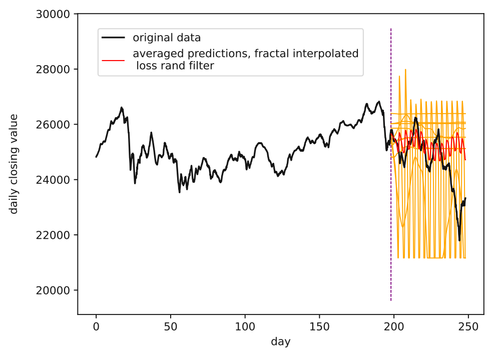
Dow Jones Daily Close



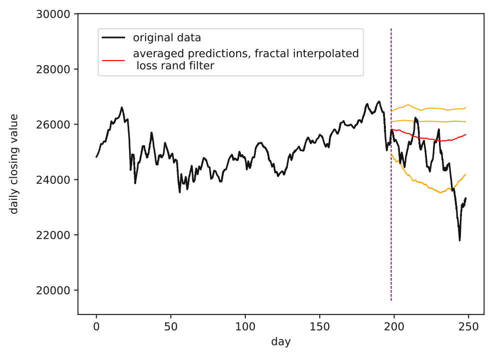
(a)



(b)

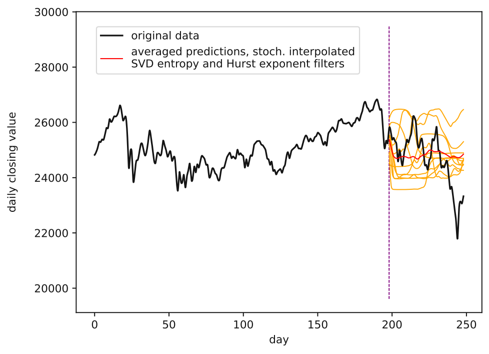


(c)

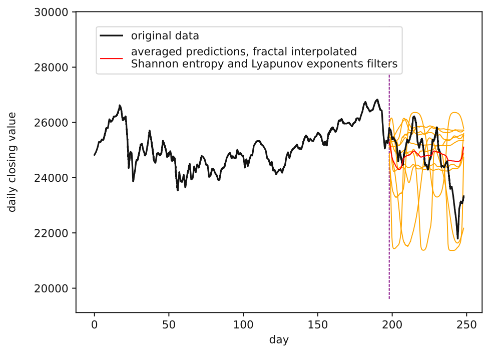


(d)

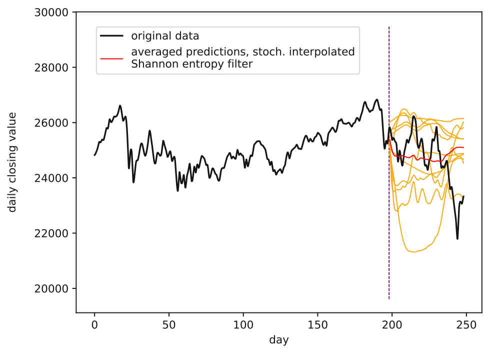
FIGURE D.15: Best predictions for the Dow Jones daily close in 2018 data set, phase-space filtered.
(a): loss_rand-filter, stoch. interpolated, 13 interpolation points, RMSE=0.18660±0.05095
(b):los2_rand-filter, fractal interpolated, 13 interpolation points, RMSE=0.21020±0.03887
(c): loss_rand-filter, fractal interpolated, 13 interpolation points, RMSE=0.21218 ±0.04764
(d): loss_rand-filter, fractal interpolated, 11 interpolation points, RMSE=0.25052±0.03035



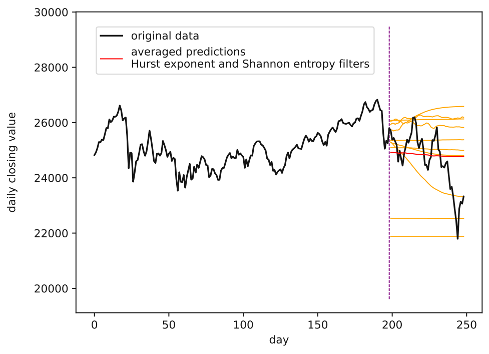
(a)



(b)



(c)



(d)

FIGURE D.16: Best predictions for the Dow Jones daily close in 2018 data set, signal-complexity filtered.
(a): SVD-Hurst-filter, stoch. interpolated, 13 interpolation points, RMSE=0.17765±0.02802
(b): Shannon-Lyap-filter, fractal interpolated, 15 interpolation points, RMSE=0.18605±0.03950
(c): Shannon-filter, stoch. interpolated, 15 interpolation points, RMSE=0.19022±0.01862
(d): Hurst-Shannon-filter, not interpolated, RMSE=0.19088±0.04289

USD/GBP Exchange Rate

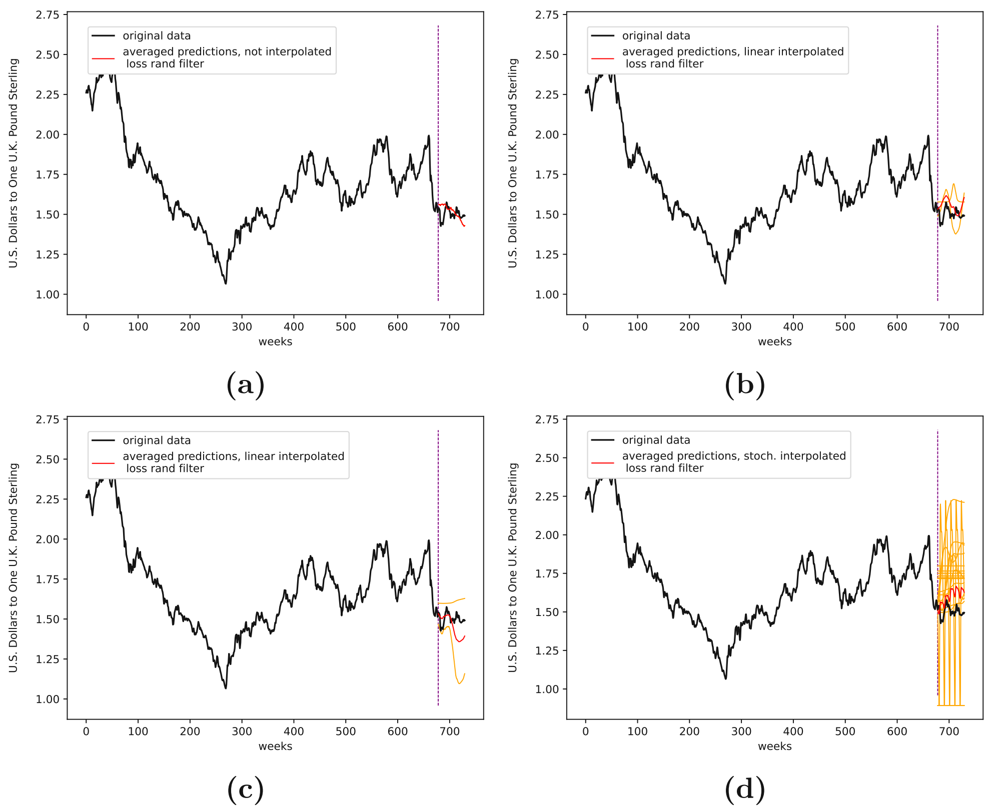


FIGURE D.17: Best predictions for the USD/GBP exchange rate data set, phase-space filtered.
(a): loss_rand-filter, not interpolated, RMSE=0.04050±0.00000
(b): loss_rand-filter, linear interpolated, 11 interpolation points, RMSE=0.05147±0.00333
(c): loss_rand-filter, linear interpolated, 15 interpolation points, RMSE=0.06343 ±0.02411
(d): loss_rand-filter, stoch. interpolated, 9 interpolation points, RMSE=0.06592±0.03623

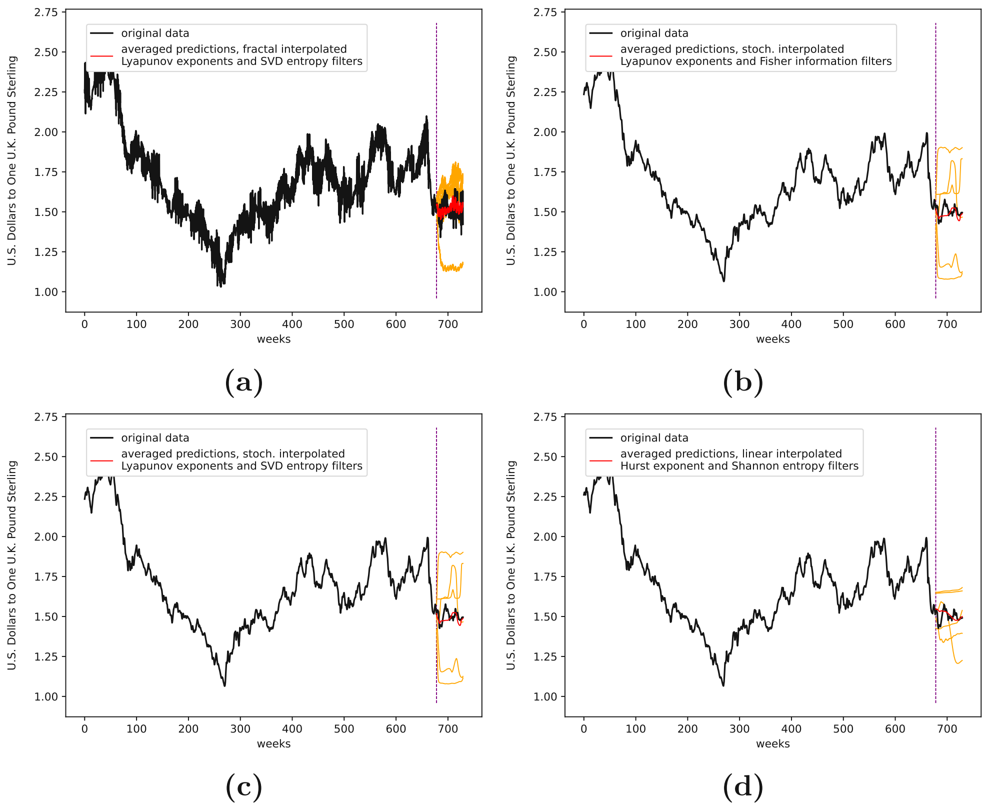
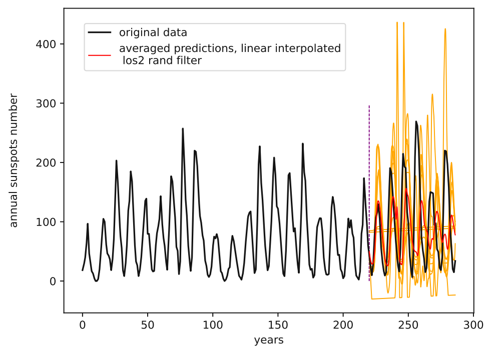
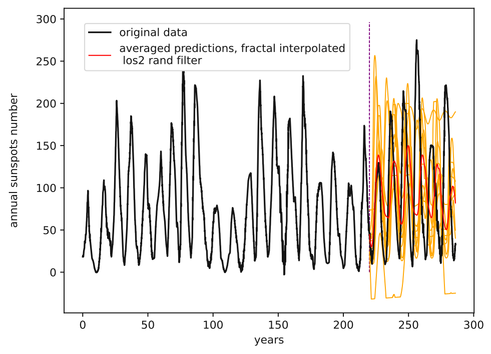


Figure D.18: Best predictions for the USD/GBP exchange data set, signal-complexity filtered.
(a): Lyap-SVD-filter, fractal interpolated, 9 interpolation points, RMSE=0.02630±0.001617
(b): Lyap-Fisher-filter, stoch. interpolated, 15 interpolation points, RMSE=0.02824±0.03077
(c): Lyap-SVD-filter, stoch. interpolated, 15 interpolation points, RMSE=0.02824±0.03077
(d): Hurst-Shannon-filter, linear interpolated, 11 interpolation points, RMSE=0.03012±0.01258

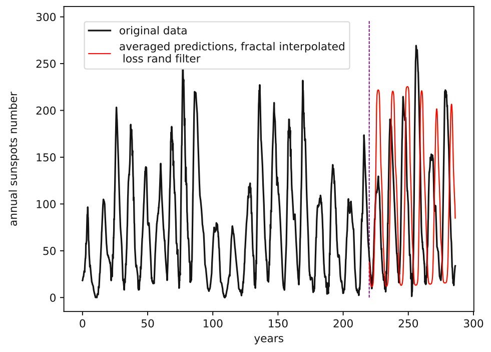
Sunspots



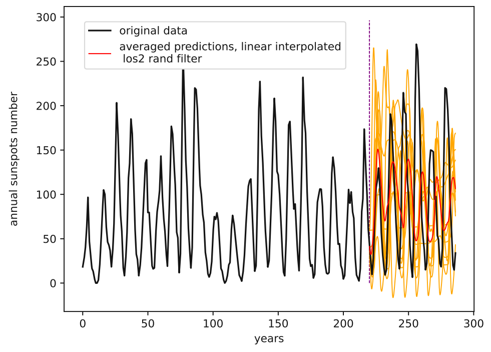
(a)



(b)



(c)



(d)

FIGURE D.19: Best predictions for the sunspots data set, phase-space filtered.
(a):los2_rand-filter, linear interpolated, 15 interpolation points, RMSE=0.29878±0.02684
(b):los2_rand-filter, fractal interpolated, 15 interpolation points, RMSE=0.29903±0.02257
(c): loss_rand-filter, fractal interpolated, 9 interpolation points, RMSE=0.30431±0.02611
(d):los2_rand-filter, linear interpolated, 9 interpolation points, RMSE=0.30706±0.01999

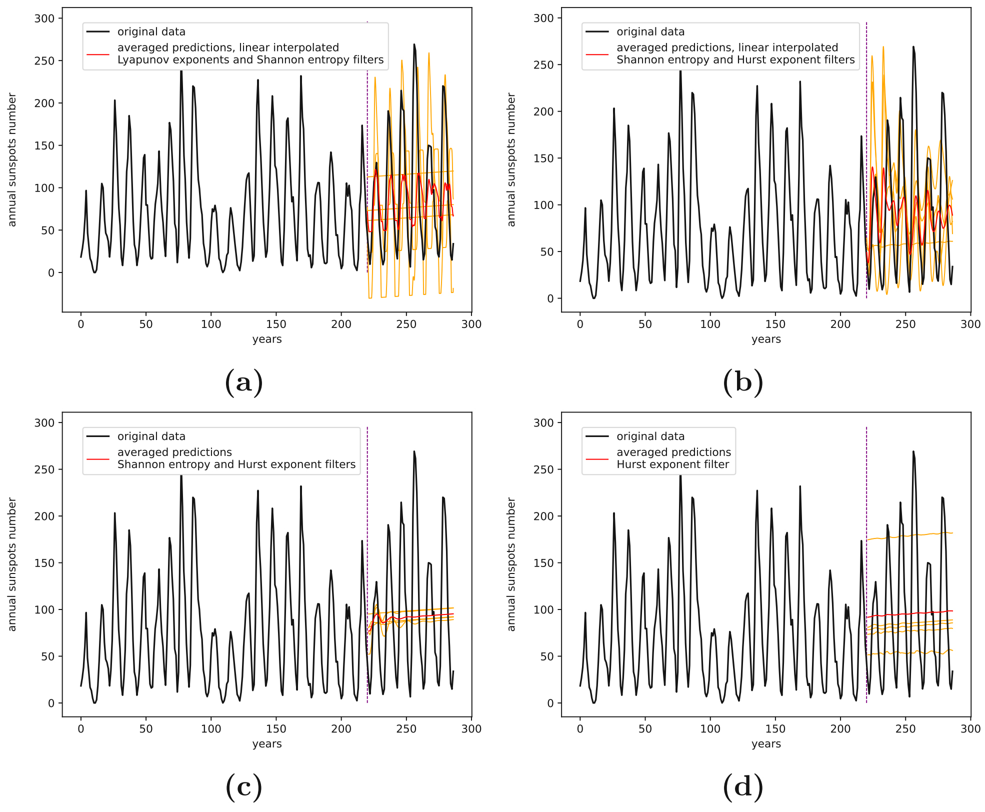
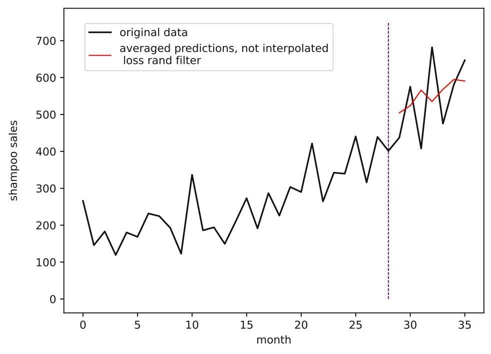
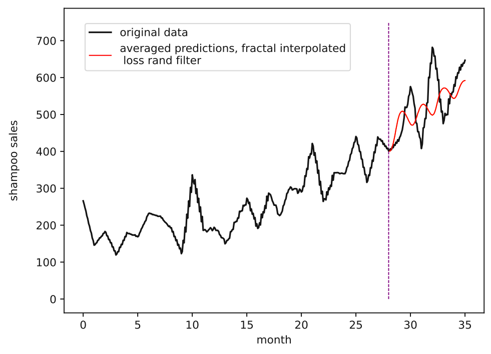


FIGURE D.20: Best predictions for the sunspots data set, signal-complexity filtered.
(a): Lyap-Shannon-filter, linear interpolated, 15 interpolation points, RMSE=0.25339±0.02052
(b): Shannon-Hurst-filter, linear interpolated, 11 interpolation points, RMSE=0.25882±0.01969
(c): Shannon-Hurst-filter, not interpolated, RMSE=0.26277±0.02800
(d): Hurst-filter, not interpolated, RMSE=0.26397±0.01979

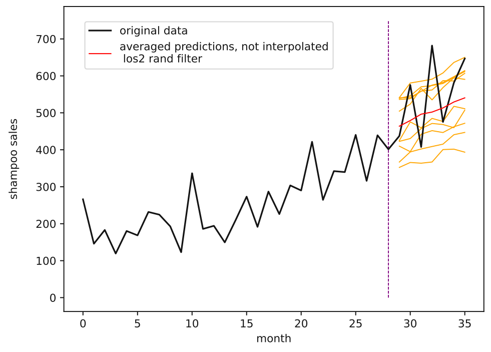
Shampoo Sales



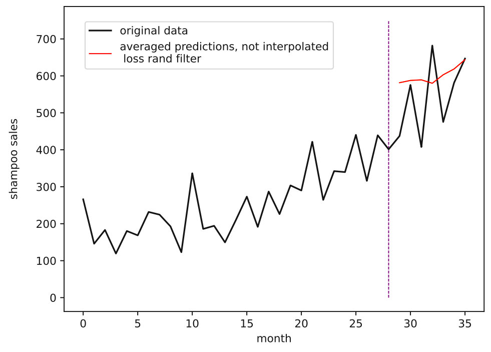
(a)



(b)



(c)



(d)

FIGURE D.21: Best predictions for the shampoo sales data set, phase-space filtered.
(a): loss_rand-filter, not interpolated,
RMSE=0.17243±0.00000
(b): loss_rand-filter, fractal interpolated, 11 interpolation points,
RMSE=0.18466±0.00000
(c):los2_rand-filter, not interpolated,
RMSE=0.18723±0.06246
(d): loss_rand-filter, not interpolated,
RMSE=0.19235±0.00000

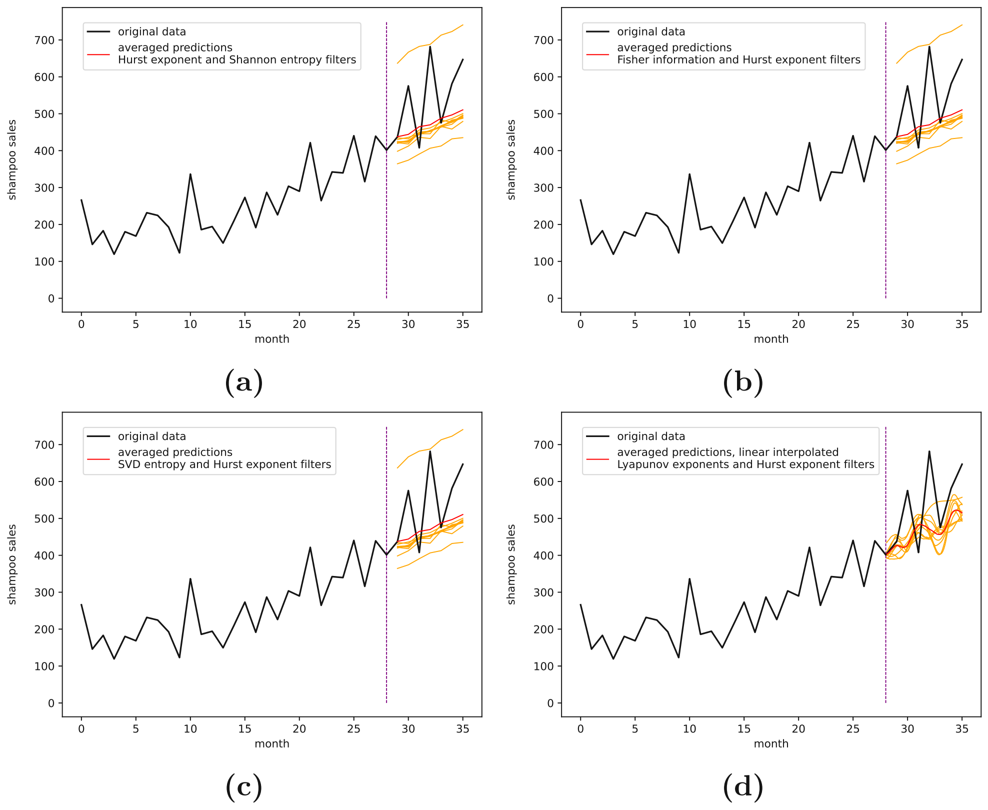


Figure D.22: Best predictions for the shampoo sales data set, signal-complexity filtered.
(a): Hurst-Shannon-filter, not interpolated,
RMSE=0.20308±0.05084
(b): Fisher-Hurst-filter, not interpolated,
RMSE=0.20308±0.05084
(c): SVD-Hurst-filter, not interpolated,
RMSE=0.20308±0.05084
(d): Lyap-Hurst-filter, linear interpolated, 11 interpolation points,
RMSE=0.20637±0.01597

D.2 Baseline Predictions

Baseline predictions, i.e. a simple RNN, LSTM and GRU (For a discussion of these methods see Chapter 6), are used to be compared to the obtained randomly paramterized LSTM ensemble predictions. All three types of neural network layers are reasonable tools for predicting time series data.

Each baseline prediciton was done using a neural network with one hidden layer containing a varying number of neurons in the hidden layer and 20 input nodes. Each neural network was trained with a batch size of 2 and varying epochs. Further `verbose` was set to 2. The corresponding varying parameters are listed in D.1. the baseline predictions for the monthly international ailrine passengers data set can be found in Appendix B.5.

For the activation of the recurrent neural network, `hard_sigmoid` was chosen. And the activation function of the output layer is `relu`. For the initialization,`glorot_uniform` was used for the LSTM layer, `orthogonal` was used as the recurrent initializer and `glorot_uniform` for the Dense layer. For the LSTM layer the bias was set to `use_bias=True`, with a corresponding `bias_initializer="zeros"`. Further, no constraints or regularizers or drop out criteria were used for the recurrent and the `Dense` layers.

As optimizer `rmsprop` was used and, the loss was calculated using `mean_squared_error`. The output node returned only one result, i.e., the next time step.

TABLE D.1: Varying parameters for the baseline predictions for Chapter 11.

Data Set	NN	Hidden Layer Neurons	Training Epochs
Measles Cases in NYC	LSTM	30	30
Measles Cases in NYC	RNN	100	23
Measles Cases in NYC	GRU	3	20
Annual Maize Yields in Austria	LSTM	30	18
Annual Maize Yields in Austria	RNN	100	23
Annual Maize Yields in Austria	GRU	3	20
Annual Wheat Yields in Austria	LSTM	30	18
Annual Wheat Yields in Austria	RNN	100	15
Annual Wheat Yields in Austria	GRU	3	20
Canadian Lynx	LSTM	39	25
Canadian Lynx	RNN	180	5
Canadian Lynx	GRU	2	1000
River Krems Discharge	LSTM	35	50
River Krems Discharge	RNN	180	300
River Krems Discharge	GRU	4	100
Dow Jones Daily Close	LSTM	39	100
Dow Jones Daily Close	RNN	180	100
Dow Jones Daily Close	GRU	4	500
USD/GBP Exchange Rate	LSTM	35	73
USD/GBP Exchange Rate	RNN	180	100
USD/GBP Exchange Rate	GRU	4	500
Sunspots	LSTM	35	100
Sunspots	RNN	180	100
Sunspots	GRU	4	350
Shampoo Sales	LSTM	30	100
Shampoo Sales	RNN	100	100
Shampoo Sales	GRU	4	400

Table D.2: Errors LSTM baseline prediction

Data Set	Train Error	Test Error	Single Step Error
Measles Cases NYC	0.05550	0.03217	0.07488
Annual Maize Yields Austria	0.16193	0.33421	0.34020
Annual Wheat Yields Austria	0.17554	0.29980	0.31402
Canadian Lynx	0.13710	0.12441	0.15404
River Krems Discharge	0.12783	0.17909	0.13502
Dow Jones Daily Close	0.05064	0.08201	0.25197
GBP/USD Exchange Rate	0.02429	0.01797	0.19573
Sunspots	0.08483	0.09009	0.25354
Shampoo Sales	0.04558	0.47413	0.42425

Table D.3: Errors RNN baseline prediction

Data Set	Train Error	Test Error	Single Step Error
Measles Cases NYC	0.05636	0.03623	0.07964
Annual Maize Yields Austria	0.17665	0.37444	0.38684
Annual Wheat Yields Austria	0.15702	0.27265	0.28145
Canadian Lynx	0.14979	0.12722	0.12847
River Krems Discharge	0.12104	0.22524	0.14961
Dow Jones Daily Close	0.06641	0.08133	0.25112
GBP/USD Exchange Rate	0.02113	0.01672	0.25613
Sunspots	0.08452	0.09540	0.29746
Shampoo Sales	0.02264	0.44773	0.38741

Table D.4: Errors GRU baseline prediction

Data Set	Train Error	Test Error	Single Step Error
Measles Cases NYC	0.06680	0.03713	0.06591
Annual Maize Yields Austria	0.17058	0.35390	0.35033
Annual Wheat Yields Austria	0.18228	0.30498	0.31784
Canadian Lynx	0.08873	0.12513	0.17024
River Krems Discharge	0.12653	0.18888	0.14963
Dow Jones Daily Close	0.04833	0.07671	0.18010
GBP/USD Exchange Rate	0.01791	0.01484	0.21395
Sunspots	0.07582	0.09127	0.21851
Shampoo Sales	0.00464	0.41179	0.37911

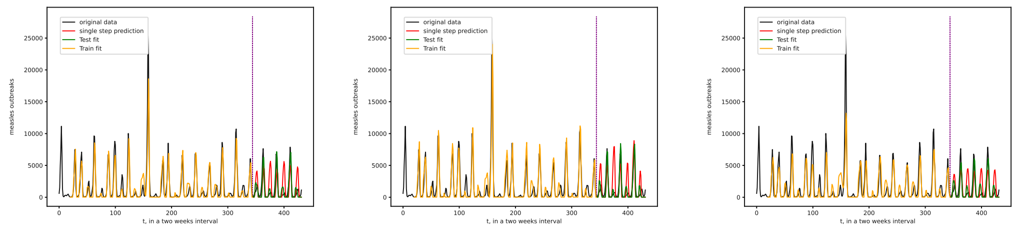


FIGURE D.23: Baseline predictions for the measles cases in NYC data set. From left to right: LSTM, RNN, GRU.

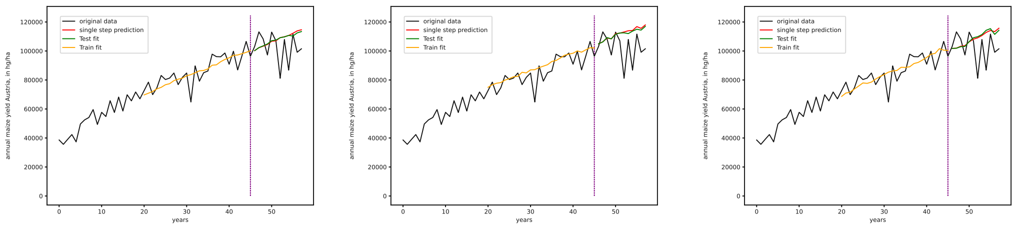


FIGURE D.24: Baseline predictions for the annual maize yields in Austria data set. From left to right: LSTM, RNN, GRU.

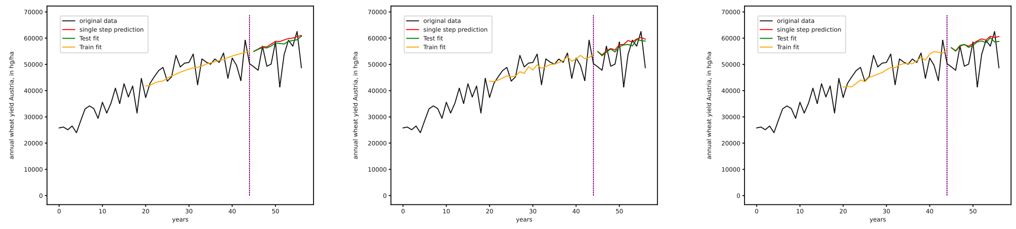


FIGURE D.25: Baseline predictions for the annual wheat yields in Austria data set. From left to right: LSTM, RNN, GRU.

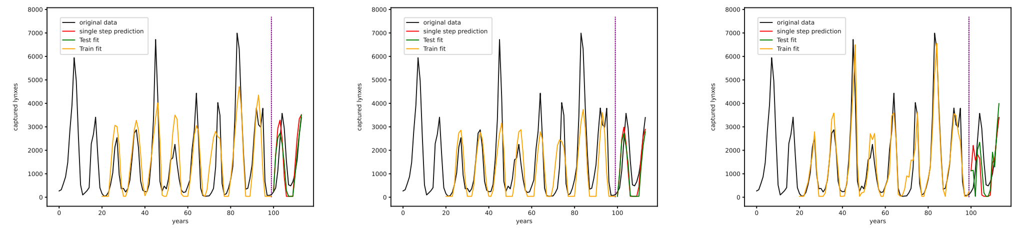


FIGURE D.26: Baseline predictions for the Canadian Lynx data set. From left to right: LSTM, RNN, GRU.

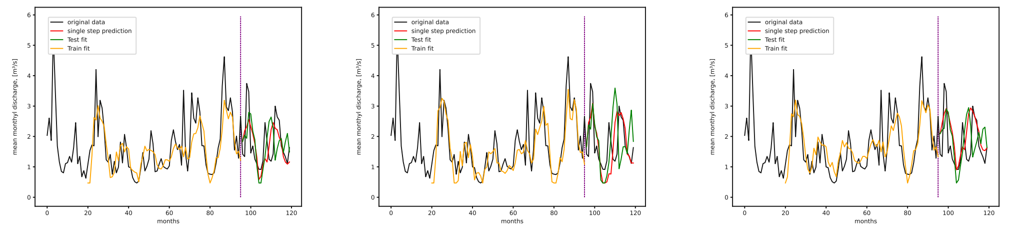


FIGURE D.27: Baseline predictions for the River Krems Discharge data set. From left to right: LSTM, RNN, GRU.

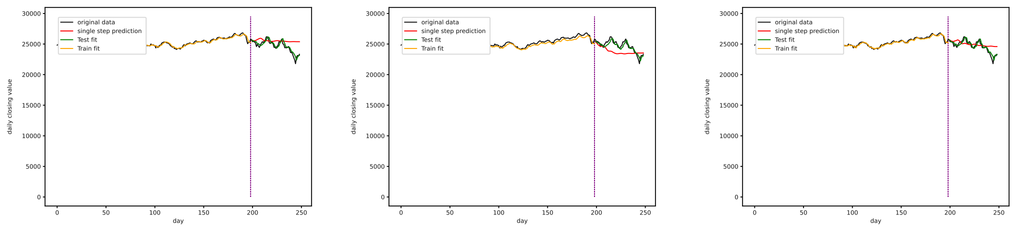


FIGURE D.28: Baseline predictions for the Dow Jones daily close in 2018 data set. From left to right: LSTM, RNN, GRU.

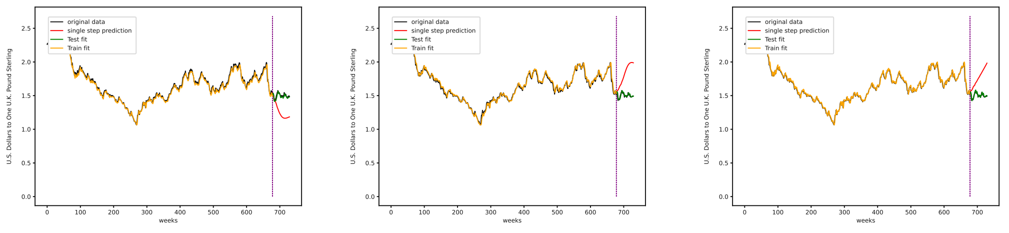


FIGURE D.29: Baseline predictions for the USD/GBP exchange rate data set. From left to right: LSTM, RNN, GRU.

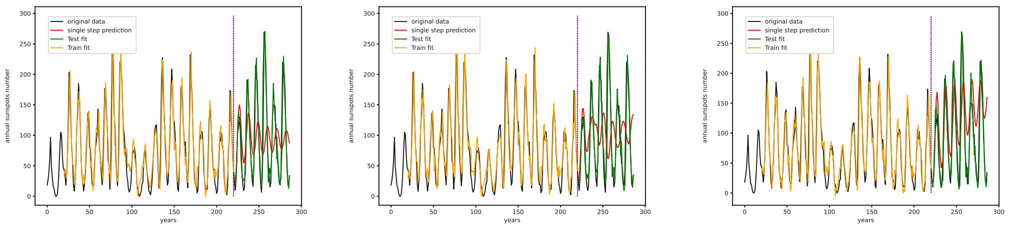


FIGURE D.30: Baseline predictions for the sunspots data set. From left to right: LSTM, RNN, GRU.

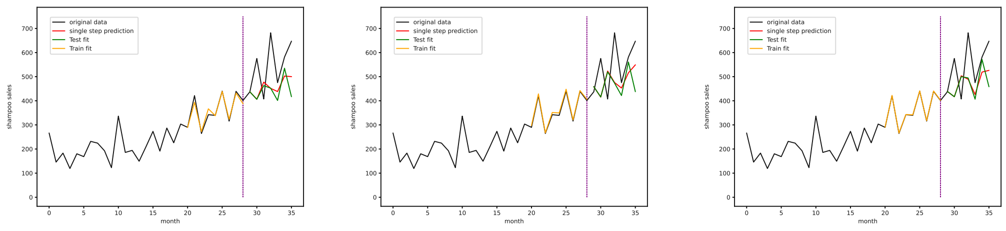


FIGURE D.31: Baseline predictions for the Shampoo Sales data set. From left to right: LSTM, RNN, GRU.

D.3 Unfiltered Results

TABLE D.5: Lowest errors unfiltered predictions measles cases in NYC data set.

Interpolation Technique	Interpolation Points	RMSE	ΔRMSE
Not Interpolated	-	0.09549	±0.01408
Fractal Interpolated	9	0.09984	±0.01414
Linear Interpolated	13	0.10962	±0.01439
Stoch. Interpolated	13	0.10020	±0.01376

TABLE D.6: Lowest errors unfiltered predictions annual maize yields in Austria data set.

Interpolation Technique	Interpolation Points	RMSE	ΔRMSE
Not Interpolated	-	0.23536	±0.05217
Fractal Interpolated	9	0.20009	±0.05822
Linear Interpolated	15	0.20328	±0.05924
Stoch. Interpolated	15	0.20499	±0.05895

Table D.7: Lowest errors unfiltered predictions annual wheat yields in Austria data set.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.23043	±0.06444
Fractal Interpolated	11	0.20101	±0.07067
Linear Interpolated	15	0.19550	±0.07183
Stoch. Interpolated	15	0.20007	±0.07381

Table D.8: Lowest errors unfiltered predictions Lorenz system.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.27121	±0.04107
Fractal Interpolated	13	0.23399	±0.04499
Linear Interpolated	13	0.25545	±0.04430
Stoch. Interpolated	13	0.26880	±0.04626
Fine-Grained Model	13	0.24652	±0.04509

TABLE D.9: Lowest errors unfiltered predictions monthly international airline passengers data set.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.19841	±0.01819
Fractal Interpolated	9	0.17977	±0.02318
Linear Interpolated	11	0.18050	±0.02323
Stoch. Interpolated	15	0.18152	±0.02410

TABLE D.10: Lowest errors unfiltered predictions Canadian lynx data set.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.23546	±0.05737
Fractal Interpolated	13	0.21280	±0.06480
Linear Interpolated	9	0.21808	±0.06587
Stoch. Interpolated	11	0.20803	±0.06419

TABLE D.11: Lowest errors unfiltered predictions river Krems discharge data set.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.24145	±0.04504
Fractal Interpolated	15	0.22149	±0.04967
Linear Interpolated	11	0.21869	±0.04915
Stoch. Interpolated	13	0.23205	±0.04894

TABLE D.12: Lowest errors unfiltered predictions Dow Jones daily close in 2018 data set.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.19841	±0.01819
Fractal Interpolated	9	0.17977	±0.02318
Linear Interpolated	11	0.18050	±0.02323
Stoch. Interpolated	15	0.18152	±0.02410

TABLE D.13: Lowest errors unfiltered predictions USD/GBP exchange rate data set.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.18917	±0.03998
Fractal Interpolated	13	0.17199	±0.04100
Linear Interpolated	11	0.18017	±0.03993
Stoch. Interpolated	15	0.14846	±0.04195

TABLE D.14: Lowest errors unfiltered predictions sunspots data set.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.37310	±0.02605
Fractal Interpolated	15	0.36459	±0.02755
Linear Interpolated	9	0.36820	±0.02727
Stoch. Interpolated	9	0.38972	±0.02599

Table D.15: Lowest errors unfiltered predictions shampoo sales data set.

Interpolation Technique	**Interpolation Points**	**RMSE**	ΔRMSE
Not Interpolated	-	0.30154	±0.04203
Fractal Interpolated	13	0.29453	±0.03971
Linear Interpolated	11	0.29291	±0.04031
Stoch. Interpolated	11	0.29473	±0.03951

Appendix E

On the Applicability of Measures of Signal Complexity

This appendix discusses the dependence of signal complexity measures on the length of the signal under study. This discussion serves to understand the results and drawbacks of Chapters 8, 9 and 11.

To better understand the dependence of the employed complexity metrics under study, we performed an experiment involving four types of time series data of variable length, first, a constant function, i.e., $f_1(t) = 1 * t^0$, second a periodic process, i.e., $f_2(t) = cos(2000 * \pi t)$, three fractional Brownian motions with three different Hurst exponents, i.e., $H \in \{0.333, 0.5, 0.666\}$ and finally an excerpt from the Lorenz system, (Section 7.15). We then calculated the discussed complexity measures employed for filtering and analysis, i.e., the Hurst exponent, Shannon's entropy, the largest Lyapunov exponent, Fisher's information, SVD entropy, and the fractal dimension for varying signal lengths, i.e., $\{50, 100, 200, 500, 1000, 1500, 2000, 5000, 10000\}$. All signals were transformed to the unit interval. Here the procedure included generating a signal with 100000 data points and then choosing 1000 times $50, 100, 200, ...$ consecutive data points to calculate the signal complexity. This resulted in an average complexity and a corresponding standard deviation shown in the following. Note that we left out $f_1(t)$ when the employed algorithm didn't give results, i.e., produced `nan`. The following plots (Figures E.1 - E.6) show the results of these experiments. We plotted the results for all the varying time steps, and a close-up of the interval $[0; 1000]$. The corresponding actual values can be found in Tables E.1 - E.6.

We observe for all employed measures of signal complexity that they converge towards a particular value with increasing signal length and that all employed complexity metrics, except for Shannon's entropy, are capable of differentiating between fractional Brownian motions of different Hurst exponents, the Lorenz system and a cosine function at a signal length of 10000. However, we cannot give a meaningful interpretation of these metrics for small signal lengths but rather explore their potential to differentiate between various signals. Also, the previously mentioned convergence towards a particular value for long signals means that these metrics give values that do not coincide with theory, e.g., the Hurst exponent is too large for small signals. Further, except for Shannon's entropy, these metrics produce large errors for small signal lengths. Also, as depicted in the following plots, all of these complexity metrics suffer from a length dependence/bias for small signal lengths.

For the Hurst exponent, we observe the value for all Brownian motions is approximately the same within errors for a signal length of 50 data points. Thus, we conclude that the Hurst exponent cannot give interpretable results for data sets of this length. Asides from interpretability, the Hurst exponent can differentiate between the cosine and the Lorenz system for all discussed signal lengths. Separating the different Brownian motions becomes possible only at ≈ 1000 data points (Figure E.1 and Table E.1). We can then justify the use of the Hurst exponent as a filter because the randomly parameterized neural networks are never producing truly random behavior but rather reproduce some aspect of the data, often resulting in sort of a periodic oscillation, a periodic dampened oscillation, or a too-smooth representation of the data, see Appendix D. This is why the Hurst exponent as a filter can give good results even for small signal lengths. It can exclude behavior that looks more like a random walk and not like regular and/or irregular/chaotic behavior.

Given the actual values of the calculated Hurst exponent, we see that the results are different from the theory, as the employed algorithm overestimates the Hurst exponent of the simulated random walks, except for the fractional Brownian motion with $H = 0.666$, where the algorithm produced a very good and even lower estimate on average at around 100 and 200 data points. Still, the corresponding errors for the average are comparatively large at these signal lengths.

Similar to the Hurst exponent, Fisher's information can also differentiate between the cosine function and the Lorenz system, Figure E.2 and Table E.2, except for a signal length of 50, where we observe comparatively large errors for the Lorenz system. Also, starting at around 1500 data points, Fisher's information can distinguish between the random walks of different Hurst exponents. Our conclusion is similar to that of the Hurst exponent because of the behavior of the predictions of the randomly parameterized neural networks. Fisher's information can differentiate between regular oscillations and more irregular behavior. Thus it can be employed as a filter for the multitude of predictions.

As SVD entropy and Fisher's information depict very similar behavior because both are based on singular value decomposition (Chapters 9 and 11), our conclusion about why SVD entropy works for filtering predictions holds just the same, Figure E.3 and Table E.3. However, SVD entropy can better differentiate between the Lorenz system and the discussed cosine function, even for the smallest discussed signal lengths. We further expected the number of dynamic components to be constant for the Lorenz system and the cosine function for increasing signal lengths, which is just what we observed, as SVD entropy provides constant values except for minor deviations. Also, the expected hierarchy, i.e., the Lorenz system has increased values compared to the cosine function, is visible in our results. However, this does not hold for fractional Brownian motion, such that the value of SVD entropy depends on the length of the signal. However, the more fluctuating fractional Brownian motion with a Hurst exponent of 0.333 has an increased SVD entropy compared to the smoother fractional Brownian motion with a Hurst exponent of 0.666, meaning that it requires more singular values above the noise floor to adequately describe the states of a fractional Brownian motion with increased Hurst exponent. On the other hand, SVD entropy provides similar results for both the Lorenz system and the cosine function at varying signal lengths, which is evident from Figure E.3 as the lines for the Lorenz system and the cosine function intersect the curves of the fractional Brownian motions. We chose the embedding for Fisher's information and SVD entropy in accordance with [60] as $d_e = 20$ and $\tau = 1$ to have a range of singular values to depict the orthogonal vectors required to adequately explain the space-state. Also, intuitively, one would expect an increasing SVD entropy for increasing signal lengths, requiring more and more singular values to describe a certain state adequately. However, we observe the SVD entropy decreases with increasing signal length for the fractional Brownian motion, which results from the averaging present in the SVD from choosing a finite embedding dimension of $d_E = 20$.

For the largest Lyapunov exponent, we see similar behavior to the previously discussed complexity metrics, i.e., it can differentiate between cosine and Lorenz system for small signal lengths, Figure E.4 and Table E.4. Also, it can differ between the Lorenz system and fractional Brownian motions for all signal lengths. What is strange is that, with increasing signal length, the largest Lyapunov exponent of the cosine becomes larger and, at one point, turns from negative to positive. Still, we conclude that the largest Lyapunov exponent can differentiate between periodic functions and irregular signals. Still, this increase for the cosine lacks a meaningful interpretation but points toward a flaw in the employed algorithm.

The behavior of Shannon's entropy (Figure E.5 and Table E.5) shows that it can differentiate between a constant function, the cosine, and irregular behavior and gives only negligible errors. However, it cannot distinguish between non-repeating signals, such as the Lorenz system, and the fractional Brownian motions. Again, being able to differentiate between regular functional and irregular behavior is the reason why Shannon's entropy can, as a filter, provide good results for specific data sets. However, Shannon's entropy suffers from a strong bias with respect to the length of the signal, i.e., short signals have a smaller entropy.

The last complexity metric discussed here is the Fractal dimension, calculated using Higuchi's algorithm, Figure E.6 and Table E.6. We did not use this metric as a filter but only for discussion in Chapter 8. The fractal dimension again shows that it can differentiate between the Lorenz system and the cosine function. However, in the author's opinion, one might be able to choose the cosine function such that the fractal dimension for these two time series is the same, or very similar. The fractal dimension can differentiate between the discussed random walks of different Hurst exponents at approximately 1500 data points for the fractional Brownian motion. If we take into account the relation from Section 8.1, i.e., $H \approx 2 - d_F$, where d_F is the fractal dimension, then we see that Higuchi's algorithm can give a better estimate for the Hurst exponent for the fractional Brownian motion with Hurst exponent $H = 0.5$. However, this approach overestimates the Hurst exponent for $H = 0.333$ and underestimates the Hurst exponent for $H = 0.666$. Though we did not use the fractal dimension for filtering, one might test this in future research.

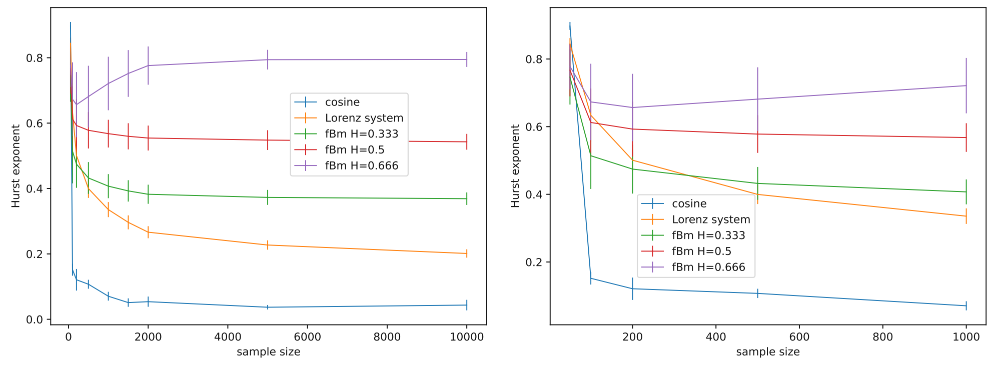


FIGURE E.1: Hurst exponent for different time series data and varying input windows. Left: full range of the experiment; Right: Close up on smaller input window sizes.

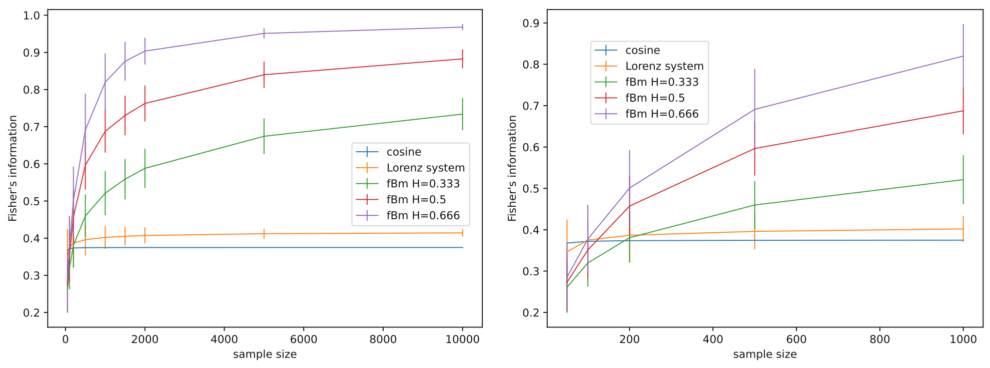


FIGURE E.2: Fisher's information for different time series data and varying input windows. Left: full range of the experiment; Right: Close up on smaller input window sizes.

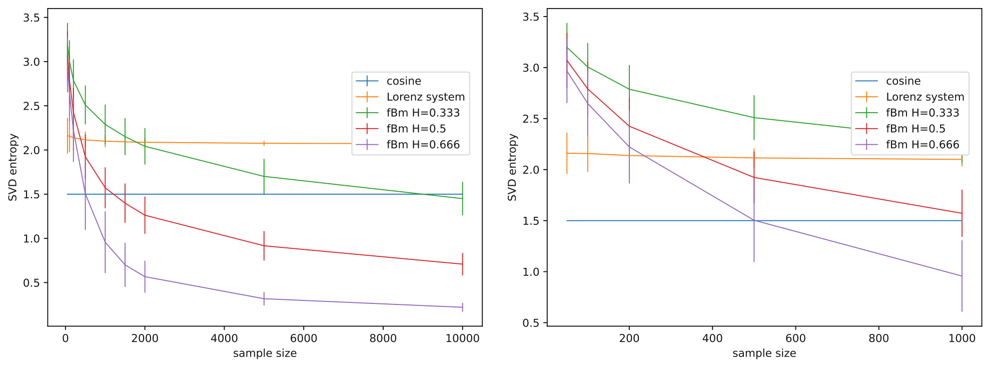


FIGURE E.3: SVD entropy for different time series data and varying input windows. Left: full range of the experiment; Right: Close up on smaller input window sizes.

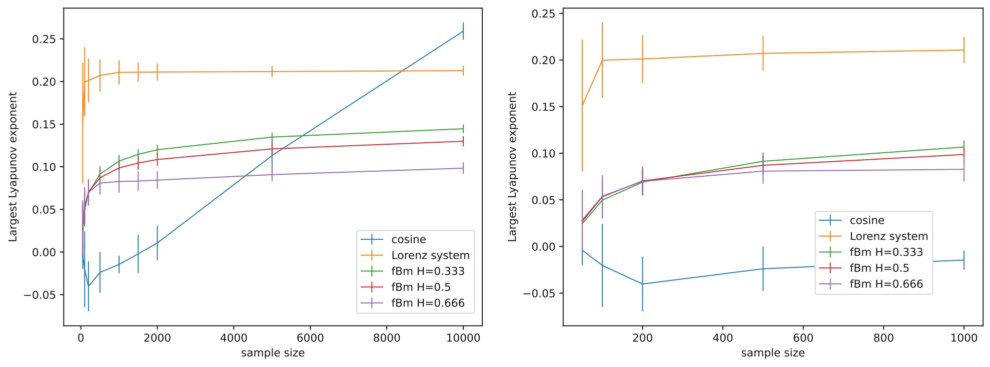


FIGURE E.4: Largest Lyapunov exponent for different time series data and varying input windows. Left: full range of the experiment; Right: Close up on smaller input window sizes.

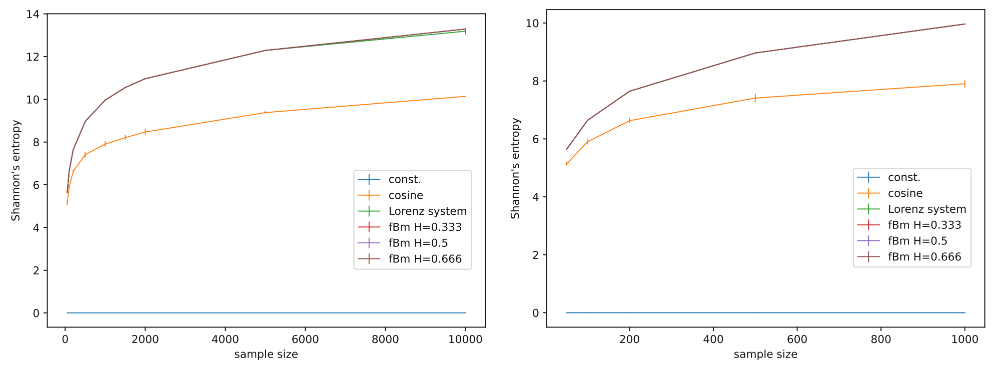


FIGURE E.5: Shannon's entropy for different time series data and varying input windows. Left: full range of the experiment; Right: Close up on smaller input window sizes.

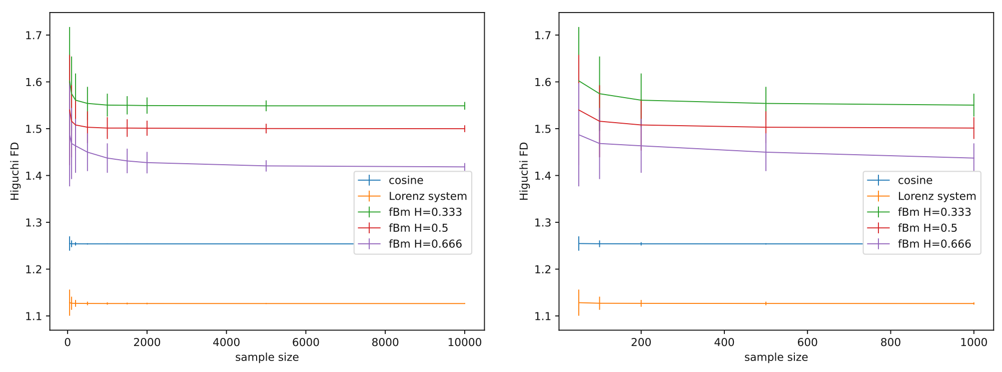


FIGURE E.6: Fractal dimension for different time series data and varying input windows. Left: full range of the experiment; Right: Close up on smaller input window sizes.

TABLE E.1: Table containing the averaged results and errors for the Hurst exponent depending on the input window size for different time series data.

Window Size	**Cosine**	**Cosine Error**	**Lorenz**	**Lorenz Error**	**fBm H=0.333**	**fBm H=0.333 Error**	**fBm H=0.5**	**fBm H=0.5 Error**	**fBm H=0.666**	**fBm H=0.666 Error**
50	0.89757	0.01182	0.84433	0.01747	0.74661	0.08133	0.76695	0.07728	0.77802	0.06766
100	0.15176	0.01845	0.63362	0.07876	0.51397	0.09819	0.61238	0.10059	0.67307	0.11288
200	0.12086	0.03307	0.50048	0.03206	0.47456	0.07239	0.59279	0.08134	0.65654	0.09961
500	0.10715	0.01370	0.39970	0.02850	0.43222	0.04861	0.57809	0.05564	0.68134	0.09415
1000	0.07050	0.01348	0.33547	0.02302	0.40724	0.03682	0.56781	0.04240	0.72125	0.08183
1500	0.05106	0.01295	0.29651	0.02137	0.39272	0.03269	0.55973	0.04003	0.75188	0.07157
2000	0.05387	0.01566	0.26636	0.01842	0.38242	0.02923	0.55438	0.03811	0.77603	0.05859
5000	0.03687	0.00684	0.22715	0.01431	0.37281	0.02294	0.54806	0.03026	0.79412	0.03004
10000	0.04341	0.01609	0.20130	0.01282	0.36887	0.01924	0.54291	0.02422	0.79474	0.02291

TABLE E.2: Table containing the averaged results and errors for Fisher's information depending on the input window size for different time series data.

Window Size	Cosine	Cosine Error	Lorenz	Lorenz Error	fBm H=0.333	fBm H=0.333 Error	fBm H=0.5	fBm H=0.5 Error	fBm H=0.666	fBm H=0.666 Error
50	0.36775	0.00000	0.34683	0.07766	0.26107	0.06176	0.27299	0.06984	0.28515	0.07981
100	0.37203	0.00000	0.37497	0.07897	0.31976	0.05728	0.35122	0.06861	0.37748	0.08276
200	0.37364	0.00000	0.38666	0.06575	0.38077	0.06000	0.45720	0.07300	0.50084	0.09153
500	0.37448	0.00000	0.39605	0.04307	0.45978	0.05801	0.59645	0.06559	0.69091	0.09803
1000	0.37474	0.00000	0.40199	0.03091	0.52111	0.05946	0.68760	0.05726	0.81998	0.07744
1500	0.37483	0.00000	0.40518	0.02555	0.55882	0.05445	0.73031	0.05303	0.87598	0.05213
2000	0.37487	0.00000	0.40729	0.02206	0.58792	0.05302	0.76259	0.04866	0.90353	0.03619
5000	0.37494	0.00000	0.41198	0.01343	0.67426	0.04801	0.83984	0.03542	0.95116	0.01366
10000	0.37497	0.00001	0.41423	0.00991	0.73391	0.04338	0.88272	0.02529	0.96800	0.00844

TABLE E.3: Table containing the averaged results and errors for SVD entropy depending on the input window size for different time series data.

Window Size	Cosine	Cosine Error	Lorenz	Lorenz Error	fBm H=0.333	fBm H=0.333 Error	fBm H=0.5	fBm H=0.5 Error	fBm H=0.666	fBm H=0.666 Error
50	1.49962	0.00000	2.16044	0.20162	3.19798	0.23869	3.06992	0.27143	2.96720	0.31438
100	1.49994	0.00000	2.15697	0.18139	3.00599	0.23571	2.79096	0.27224	2.64799	0.32158
200	1.49999	0.00000	2.13759	0.14517	2.78718	0.23740	2.42499	0.27786	2.22174	0.35704
500	1.49999	0.00000	2.11432	0.09372	2.50928	0.21942	1.92286	0.25328	1.50341	0.40878
1000	1.49999	0.00000	2.09978	0.06805	2.28893	0.22490	1.57200	0.23158	0.95685	0.34989
1500	1.49999	0.00000	2.09204	0.05616	2.15117	0.20934	1.39848	0.22300	0.70022	0.24869
2000	1.49999	0.00000	2.08695	0.04843	2.04151	0.20609	1.26222	0.21043	0.56704	0.18023
5000	1.49999	0.00000	2.07560	0.02877	1.70092	0.19886	0.91624	0.16671	0.31723	0.07638
10000	1.49999	0.00000	2.07017	0.02088	1.44970	0.18943	0.70876	0.12700	0.22031	0.05035

TABLE E.4: Table containing the averaged results and errors for the largest Lyapunov exponent depending on the input window size for different time series data.

Window Size	Cosine	Cosine Error	Lorenz	Lorenz Error	fBm H=0.333	fBm H=0.333 Error	fBm H=0.5	fBm H=0.5 Error	fBm H=0.666	fBm H=0.666 Error
50	-0.00401	0.01599	0.15121	0.07074	0.02436	0.02992	0.02679	0.03102	0.02854	0.03221
100	-0.02031	0.04452	0.19983	0.04028	0.04969	0.01971	0.05332	0.02026	0.05400	0.02246
200	-0.04022	0.02932	0.20111	0.02560	0.06900	0.01386	0.07032	0.01509	0.06990	0.01539
500	-0.02391	0.02388	0.20713	0.01889	0.09139	0.00941	0.08708	0.01050	0.08080	0.01359
1000	-0.01453	0.01028	0.21069	0.01417	0.10668	0.00719	0.09866	0.00850	0.08275	0.01315
1500	-0.00212	0.02257	0.21088	0.01124	0.11455	0.00629	0.10443	0.00786	0.08319	0.01124
2000	0.01062	0.02007	0.21112	0.01009	0.12001	0.00595	0.10850	0.00750	0.08435	0.01034
5000	0.11313	0.01970	0.21167	0.00649	0.13483	0.00516	0.12095	0.00664	0.09077	0.00770
10000	0.25905	0.01003	0.21273	0.00575	0.14459	0.00514	0.12995	0.00601	0.09838	0.00675

TABLE E.5: Table containing the averaged results and errors for Shannon's entropy depending on the input window size for different time series data.

Window Size	Const.	Const. Error	Cosine	Cosine Error	Lorenz	Lorenz Error	fBm H=0.333	fBm H=0.333 Error	fBm H=0.5	fBm H=0.5 Error	fBm H=0.666	fBm H=0.666 Error
50	0	0	5.13870	0.07511	5.64385	0	5.64385	0	5.64385	0	5.64385	0
100	0	0	5.90223	0.08951	6.64385	0	6.64385	0	6.64385	0	6.64385	0
200	0	0	6.62910	0.08842	7.64385	0	7.64385	0	7.64385	0	7.64385	0
500	0	0	7.40990	0.14190	8.96578	0	8.96578	0	8.96578	0	8.96578	0
1000	0	0	7.90332	0.13580	9.96578	0	9.96578	0	9.96578	0	9.96578	0
1500	0	0	8.20617	0.11387	10.55074	0	10.55074	0	10.55074	0	10.55074	0
2000	0	0	8.46953	0.15246	10.96578	0	10.96578	0	10.965784	0	10.9657	0
5000	0	0	9.37903	0.07216	12.28633	0.01647	12.28771	0	12.28771	0	12.28771	0
10000	0	0	10.13963	0.01500	13.18841	0.15046	13.28771	0	13.28771	0	13.28771	0

TABLE E.6: Table containing the averaged results and errors for the fractal dimension calculated using Higuchi's algorithm depending on the input window size for different time series data.

Window Size	Cosine	Cosine Error	Lorenz	Lorenz Error	fBm H=0.333	fBm H=0.333 Error	fBm H=0.5	fBm H=0.5 Error	fBm H=0.666	fBm H=0.666 Error
50	1.25470	0.01533	1.12839	0.02795	1.60218	0.11503	1.53992	0.11772	1.48706	0.11023
100	1.25416	0.00722	1.12713	0.01402	1.57471	0.07952	1.51589	0.07692	1.46833	0.07598
200	1.25400	0.00355	1.12689	0.00729	1.56085	0.05710	1.50792	0.05459	1.46341	0.05763
500	1.25395	0.00140	1.12667	0.00378	1.55395	0.03546	1.50309	0.03393	1.44979	0.04034
1000	1.25394	0.00070	1.12655	0.00246	1.55034	0.02447	1.50133	0.02330	1.43721	0.03145
1500	1.25394	0.00046	1.12655	0.00191	1.54996	0.01953	1.50132	0.01879	1.43109	0.02623
2000	1.25394	0.00035	1.12655	0.00166	1.54935	0.01738	1.50101	0.01612	1.42751	0.02296
5000	1.25394	0.00014	1.12651	0.00104	1.54882	0.01158	1.50026	0.01029	1.42051	0.01209
10000	1.25394	0.00003	1.12650	0.00082	1.54886	0.00824	1.49994	0.00735	1.41839	0.00819

E.1 Variance of Second Derivatives

We also test the variance of second derivatives along a reconstructed phase space trajectory in the experimental setup from the previous section. We also try different phase space embeddings to show how the variance of second derivatives can differentiate between various signals depending on the chosen phase space embedding. We chose five different phase space embeddings, i.e., pairs of embedding dimension and time delay as $(d_E; \tau) = \{(1;0), (3;1), (3;5), (5;1), (5;5)\}$.

The results (Tables E.7 - E.11) are depicted in Figures E.7 - E.11.

Similar to the previous section's results, we observe that the variance of second derivatives converges towards certain values for the different types of time series data under study for increasing signal length. Fractional Brownian motion with Hurst exponents of ≥ 0.5 seem difficult to distinguishable, given our results. However, our signal with a Hurst exponent of 0.333 can be distinguished quite well from all other fractional Brownian motions for all phase space embeddings. The reason here is that signals with a Hurst exponent of below 0.5 are considered antipersistent, thus, do not provide linear behavior for long or at all. Therefore these signals produce an increased variance of the second derivatives in reconstructed phase space, compared to a signal with an increased Hurst exponent of, e.g., 0.9, which will provide a very linear signal. The const. function has a zero variance of second derivatives for all phase space embeddings because it's not changing. The variance of second derivatives for the cosine function, though staying mostly constant for all phase space embeddings, shifts its relative position with respect to the other data for different phase space embeddings. We see the best separation for $(d_E = 3; \tau = 5)$. The cosine shows a similar behavior as the fractional Brownian motions with different Hurst exponents and negligible errors for $(d_E = 5, \tau = 1)$, as the curve is decreasing and converging towards a certain value, in this case very close to zero (We didn't print out values lower than 10^{-5}). Thus, we assume that one can choose an embedding that is beneficial for separating good and bad predictions. In the case of the cosine, the reason is that one can choose a phase space embedding such that the reconstructed phase space of the cosine is (almost) a circle, thus we observe a vanishing variance of second derivatives. This is due to the periodicity/autocorrelations of the cosine function. One can exploit this to find an optimal separation between different signals, i.e., by taking into account a signal's autocorrelations and using this for a phase space embedding that provides an optimal separation from, e.g., chaotic or random signals. Thus this applies to some degree to the Lorenz system as well. Given that one can choose a phase space embedding where the variance of second derivatives is increased or decreased (We show this for the Lorenz system in Section C.3), one might achieve an optimal separation from other signals. This should be considered for future research regarding the filtering of forecasts.

Overall the results show that the variance of second derivatives is capable of differentiating between different types of time series data. Still, for short signals, this is a difficult task due to the large errors and/or artifacts of the phase space embedding.

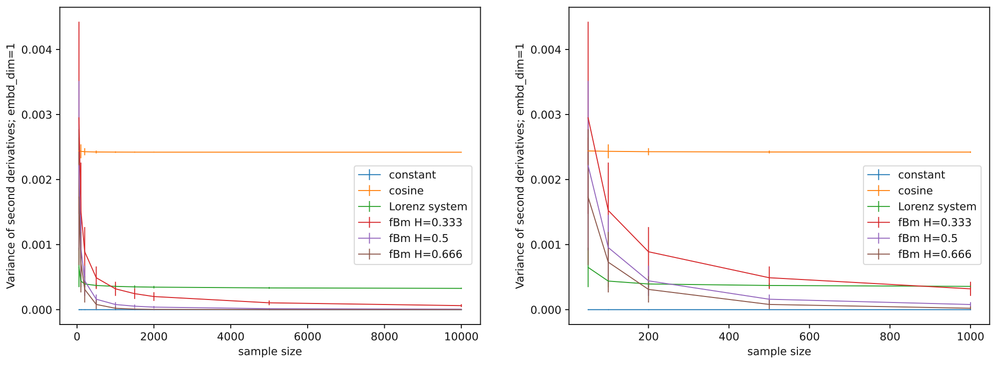


FIGURE E.7: Variance of second derivatives along a reconstructed phase space trajectory for different time series data and varying input windows. The time delay embedding was chosen as $\tau = 1, d_E = 1$ Left: full range of the experiment; Right: Close up on smaller input window sizes.

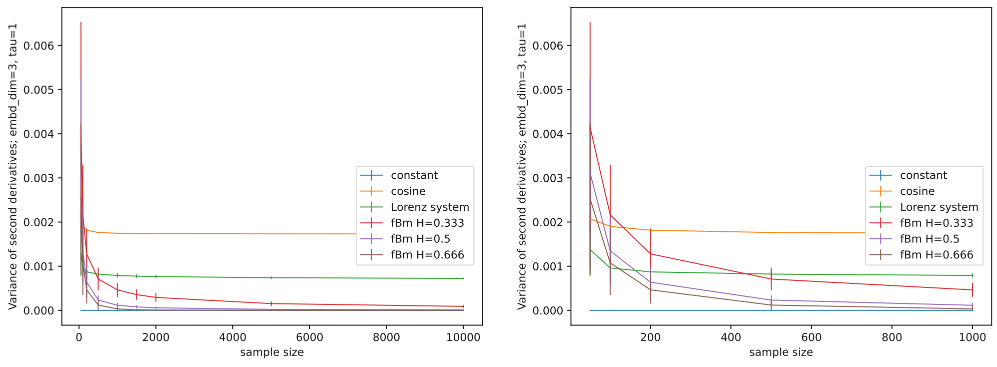


FIGURE E.8: Variance of second derivatives along a reconstructed phase space trajectory for different time series data and varying input windows. The time delay embedding was chosen as $d_E = 3, \tau = 1$ Left: full range of the experiment; Right: Close up on smaller input window sizes.

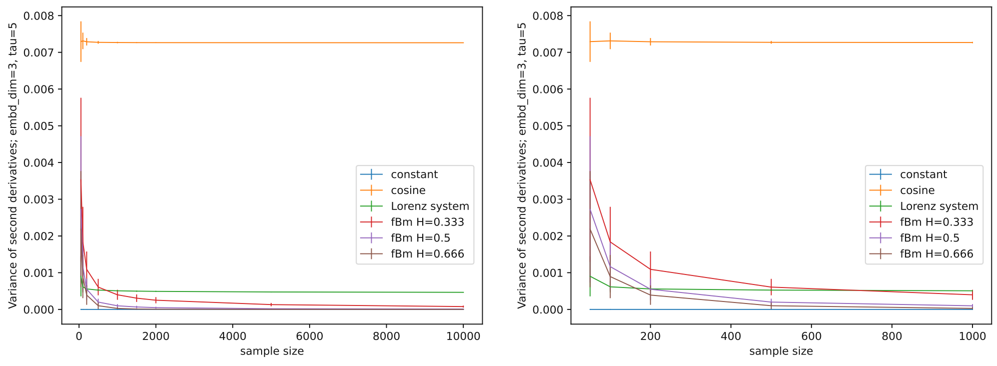


FIGURE E.9: Variance of second derivatives along a reconstructed phase space trajectory for different time series data and varying input windows. The time delay embedding was chosen as $d_E = 3, \tau = 5$ Left: full range of the experiment; Right: Close up on smaller input window sizes.

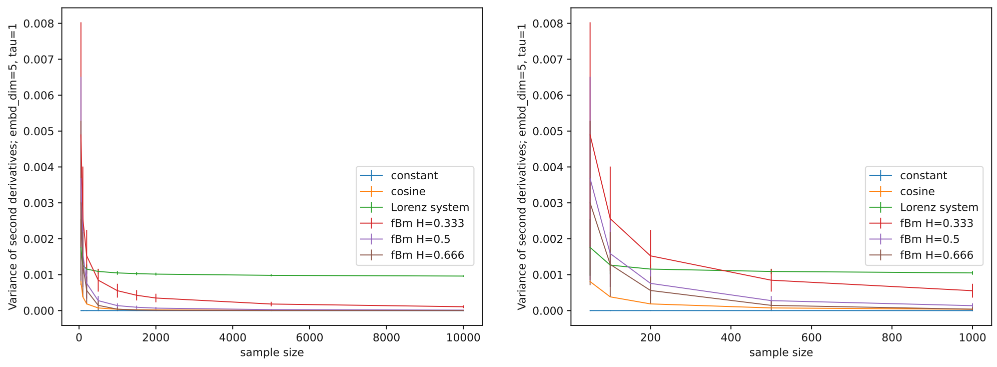


FIGURE E.10: Variance of second derivatives along a reconstructed phase space trajectory for different time series data and varying input windows. The time delay embedding was chosen as $d_E = 5, \tau = 1$ Left: full range of the experiment; Right: Close up on smaller input window sizes.

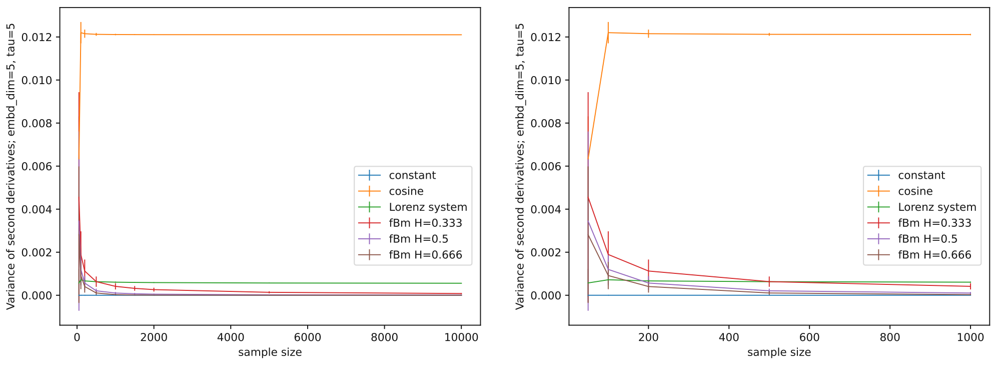


FIGURE E.11: Variance of second derivatives along a reconstructed phase space trajectory for different time series data and varying input windows. The time delay embedding was chosen as $d_E = 5, \tau = 5$ Left: full range of the experiment; Right: Close up on smaller input window sizes.

TABLE E.7: Table containing the averaged results and errors for the variance of second derivatives along the curve, depending on the input window size for different time series data. No embedding, $d_E = 1$.

Window Size	**Const.**	**Const. Error**	**Cosine**	**Cosine Error**	**Lorenz**	**Lorenz Error**	**fBm H=0.333**	**fBm H=0.333 Error**	**fBm H=0.5**	**fBm H=0.5 Error**	**fBm H=0.666**	**fBm H=0.666 Error**
50	0	0	0.00244	0.00021	0.00065	0.00030	0.00295	0.00148	0.00221	0.00130	0.00173	0.00104
100	0	0	0.00243	0.00011	0.00044	0.00009	0.00153	0.00074	0.00096	0.00050	0.00073	0.00046
200	0	0	0.00243	0.00005	0.00040	0.00005	0.00089	0.00038	0.00044	0.00023	0.00031	0.00020
500	0	0	0.00242	0.00002	0.00037	0.00003	0.00049	0.00017	0.00016	0.00007	0.00008	0.00007
1000	0	0	0.00242	0.00001	0.00036	0.00003	0.00032	0.00011	0.00008	0.00004	0.00002	0.00003
1500	0	0	0.00242	0.00001	0.00035	0.00002	0.00025	0.00008	0.00005	0.00002	0.00001	0.00001
2000	0	0	0.00242	0.00001	0.00035	0.00002	0.00020	0.00007	0.00004	0.00002	0.00000	0.00001
5000	0	0	0.00242	0.00000	0.00033	0.00001	0.00011	0.00004	0.00002	0.00001	0.00000	0.00000
10000	0	0	0.00242	0.00000	0.00033	0.00001	0.00006	0.00002	0.00001	0.00000	0.00000	0.00000

TABLE E.8: Table containing the averaged results and errors for the variance of second derivatives along the curve in reconstructed phase space, depending on the input window size for different time series data. Embedding: $d_E = 3$, $\tau = 1$.

Window Size	**Const.**	**Const. Error**	**Cosine**	**Cosine Error**	**Lorenz**	**Lorenz Error**	**fBm H=0.333**	**fBm H=0.333 Error**	**fBm H=0.5**	**fBm H=0.5 Error**	**fBm H=0.666**	**fBm H=0.666 Error**
50	0	0	0.00207	0.00012	0.00138	0.00060	0.00415	0.00238	0.00311	0.00213	0.00253	0.00171
100	0	0	0.00190	0.00007	0.00096	0.00019	0.00216	0.00113	0.00135	0.00076	0.00107	0.00073
200	0	0	0.00181	0.00003	0.00087	0.00010	0.00128	0.00058	0.00064	0.00034	0.00046	0.00031
500	0	0	0.00176	0.00001	0.00082	0.00006	0.00071	0.00026	0.00023	0.00011	0.00012	0.00011
1000	0	0	0.00175	0.00001	0.00079	0.00005	0.00046	0.00016	0.00011	0.00005	0.00003	0.00004
1500	0	0	0.00174	0.00000	0.00077	0.00004	0.00036	0.00012	0.00008	0.00004	0.00001	0.00002
2000	0	0	0.00174	0.00000	0.00076	0.00004	0.00029	0.00010	0.00006	0.00003	0.00001	0.00001
5000	0	0	0.00173	0.00000	0.00074	0.00003	0.00015	0.00005	0.00002	0.00001	0.00000	0.00000
10000	0	0	0.00173	0.00000	0.00072	0.00002	0.00009	0.00003	0.00001	0.00000	0.00000	0.00000

Table E.9: Table containing the averaged results and errors for the variance of second derivatives along the curve in reconstructed phase space, depending on the input window size for different time series data. Embedding: $d_E = 3$, $\tau = 5$.

Window Size	**Const.**	**Const. Error**	**Cosine**	**Cosine Error**	**Lorenz**	**Lorenz Error**	**fBm H=0.333**	**fBm H=0.333 Error**	**fBm H=0.5**	**fBm H=0.5 Error**	**fBm H=0.666**	**fBm H=0.666 Error**
50	0	0	0.00729	0.00055	0.00090	0.00055	0.00353	0.00223	0.00272	0.00199	0.00219	0.00158
100	0	0	0.00731	0.00023	0.00062	0.00015	0.00184	0.00095	0.00117	0.00066	0.00089	0.00059
200	0	0	0.00729	0.00010	0.00056	0.00005	0.00109	0.00049	0.00055	0.00030	0.00039	0.00027
500	0	0	0.00727	0.00004	0.00053	0.00003	0.00061	0.00022	0.00020	0.00009	0.00010	0.00009
1000	0	0	0.00727	0.00002	0.00051	0.00002	0.00040	0.00014	0.00010	0.00005	0.00003	0.00003
1500	0	0	0.00727	0.00001	0.00050	0.00002	0.00031	0.00010	0.00007	0.00003	0.00001	0.00002
2000	0	0	0.00726	0.00001	0.00049	0.00001	0.00025	0.00008	0.00005	0.00002	0.00001	0.00001
5000	0	0	0.00726	0.00000	0.00047	0.00001	0.00013	0.00005	0.00002	0.00001	0.00000	0.00000
10000	0	0	0.00726	0.00000	0.00047	0.00001	0.00008	0.00003	0.00001	0.00000	0.00000	0.00000

Table E.10: Table containing the averaged results and errors for the variance of second derivatives along the curve in reconstructed phase space, depending on the input window size for different time series data. Embedding: $d_E = 5$, $\tau = 1$

Window Size	**Const.**	**Const. Error**	**Cosine**	**Cosine Error**	**Lorenz**	**Lorenz Error**	**fBm H=0.333**	**fBm H=0.333 Error**	**fBm H=0.5**	**fBm H=0.5 Error**	**fBm H=0.666**	**fBm H=0.666 Error**
50	0	0	0.00081	0.00000	0.00176	0.00079	0.00489	0.00314	0.00368	0.00283	0.00300	0.00229
100	0	0	0.00038	0.00000	0.00127	0.00025	0.00256	0.00145	0.00159	0.00093	0.00129	0.00090
200	0	0	0.00019	0.00000	0.00116	0.00013	0.00152	0.00072	0.00076	0.00042	0.00056	0.00039
500	0	0	0.00007	0.00000	0.00109	0.00007	0.00085	0.00032	0.00028	0.00013	0.00014	0.00013
1000	0	0	0.00004	0.00000	0.00105	0.00005	0.00055	0.00019	0.00014	0.00006	0.00004	0.00005
1500	0	0	0.00002	0.00000	0.00103	0.00004	0.00043	0.00014	0.00009	0.00004	0.00002	0.00002
2000	0	0	0.00002	0.00000	0.00102	0.00004	0.00035	0.00012	0.00007	0.00003	0.00001	0.00001
5000	0	0	0.00001	0.00000	0.00098	0.00003	0.00018	0.00006	0.00003	0.00001	0.00000	0.00000
10000	0	0	0.00000	0.00000	0.00096	0.00002	0.00011	0.00004	0.00001	0.00001	0.00000	0.00000

Table E.11: Table containing the averaged results and errors for the variance of second derivatives along the curve in reconstructed phase space, depending on the input window size for different time series data. Embedding: $d_E = 5$, $\tau = 5$

Window Size	**Const.**	**Const. Error**	**Cosine**	**Cosine Error**	**Lorenz**	**Lorenz Error**	**fBm H=0.333**	**fBm H=0.333 Error**	**fBm H=0.5**	**fBm H=0.5 Error**	**fBm H=0.666**	**fBm H=0.666 Error**
50	0	0	0.00634	0.00197	0.00057	0.00064	0.00455	0.00488	0.00344	0.00416	0.0028	0.0032
100	0	0	0.01220	0.00049	0.00072	0.00023	0.00189	0.00108	0.00120	0.00075	0.0009	0.0006
200	0	0	0.01215	0.00019	0.00067	0.00008	0.00113	0.00053	0.00057	0.00032	0.0004	0.0003
500	0	0	0.01212	0.00007	0.00063	0.00004	0.00063	0.00024	0.00021	0.00010	0.0001	0.0001
1000	0	0	0.01211	0.00003	0.00061	0.00002	0.00041	0.00014	0.00010	0.00005	0.0000	0.0000
1500	0	0	0.01211	0.00002	0.00060	0.00002	0.00032	0.00011	0.00007	0.00003	0.0000	0.0000
2000	0	0	0.01211	0.00002	0.00059	0.00002	0.00026	0.00009	0.00005	0.00002	0.0000	0.0000
5000	0	0	0.01210	0.00001	0.00057	0.00001	0.00014	0.00005	0.00002	0.00001	0.0000	0.0000
10000	0	0	0.01210	0.00000	0.00056	0.00001	0.00008	0.00003	0.00001	0.00000	0.0000	0.0000

Bibliography

[1] Jonathan Schmidt, Mário R. G. Marques, Silvana Botti, and Miguel A. L. Marques. Recent advances and applications of machine learning in solid-state materials science. npj Computational Materials, 5(1):83, August 2019. ISSN 2057-3960. doi: 10.1038/s41524-019-0221-0. URL https://www.nature.com/articles/s41524-019-0221-0. Visited on 2023-04-20.

[2] Cyril Voyant, Gilles Notton, Soteris Kalogirou, Marie-Laure Nivet, Christophe Paoli, Fabrice Motte, and Alexis Fouilloy. Machine learning methods for solar radiation forecasting: A review. Renewable Energy, 105:569–582, May 2017. ISSN 0960-1481. doi: 10.1016/j.renene.2016.12.095. URL http://www.sciencedirect.com/science/article/pii/S0960148116311648. Visited on 2023-04-20.

[3] Pedro Larrañaga, Borja Calvo, Roberto Sanatana, Concha Bielza, Josu Galdiano, Inaki Inza, José A. Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, and Victor Robles. Machine learning in bioinformatics. Briefings in Bioinformatics, 7(1):86–112, 2005. doi: 10.1093/bib/bbk007.

[4] Anna Chlingaryan, Salah Sukkarieh, and Brett Whelan. Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. Computers and Electronics in Agriculture, 151:61–69, 2018. ISSN 0168-1699. doi: 10.1016/j.compag.2018.05.012. URL https://www.sciencedirect.com/science/article/pii/S0168169917314710. Visited on 2023-04-20.

[5] Alvin Rajkomar, Jeffrey Dean, and Isaac Kohane. Machine Learning in Medicine. The New England Journal of Medicine, 380:1347–1358, 2019. doi: 10.1056/NEJMra1814259.

[6] J. B. Heaton, N. G. Polson, and J. H. Witte. Deep learning in finance, 2016. URL https://arxiv.org/abs/1602.06561. Visited on 2023-04-20.

[7] H.E. Hurst, R. Black, and Y.M. Sinaika. Long-term Storage in Reservoirs: An experimental Study. Constable: London, 1965.

[8] Bo Qian and Khaled Rasheed. Hurst exponent and financial market predictability. Proceedings of the 2nd IASTED International Conference on Financial Engineering and Applications, Cambridge, MA:203–209, 2004.

[9] Sunkyu Yu, Xianji Piao, Jiho Hong, and Namkyoo Park. Bloch-like waves in random-walk potentials based on supersymmetry. Nature Communications, 6(1): 8269, September 2015. ISSN 2041-1723. doi: 10.1038/ncomms9269.

[10] Shuying Chen, Liping Yu, Jingli Ren, Xie Xie, Xueping Li, Ying Xu, Guangfeng Zhao, Peizhen Li, Fuqian Yang, Yang Ren, and Peter K. Liaw. Self-Similar Random Process and Chaotic Behavior In Serrated Flow of High Entropy Alloys. Scientific Reports, 6(1):29798, July 2016. ISSN 2045-2322. doi: 10.1038/srep29798. URL https://www.nature.com/articles/srep29798. Visited on 2023-04-20.

[11] Wang-Kun Chen and Ping Wang. Fuzzy Forecasting with Fractal Analysis for the Time Series of Environmental Pollution, pages 199–213. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-33439-9. doi: 10.1007/978-3-642-33439-9_9. URL https://link.springer.com/chapter/10.1007/978-3-642-33439-9_9. Visited on 2023-04-20.

[12] S M Pincus. Approximate entropy as a measure of system complexity. Proceedings of the National Academy of Sciences, 88(6):2297–2301, 1991. doi: 10.1073/pnas.88.6.2297. URL https://www.pnas.org/doi/abs/10.1073/pnas.88.6.2297. Visited on 2023-04-20.

[13] O. Castillo and P. Melin. Hybrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory. IEEE Transactions on Neural Networks, 13(6):1395–1408, 2002. doi: 10.1109/TNN.2002.804316. URL https://ieeexplore.ieee.org/document/1058075. Visited on 2023-04-20.

[14] Indranil Ghosh and Tamal Datta Chaudhuri. Fractal Investigation and Maximal Overlap Discrete Wavelet Transformation (MODWT)-based Machine Learning Framework for Forecasting Exchange Rates. Studies in Microeconomics, 5(2): 1–27, 2017. doi: 10.1177/2321022217724978. URL https://journals.sagepub.com/doi/abs/10.1177/2321022217724978?journalCode=mica. Visited on 2023-04-20.

[15] Eugen Diaconescu. The Use of NARX Neural Networks to Predict Chaotic Time Series. WSEAS Trans. Comp. Res., 3(3):182–191, mar 2008. ISSN 1991-8755. URL https://dl.acm.org/doi/10.5555/1466884.1466892. Visited on 2023-04-20.

[16] William Gilpin. Deep reconstruction of strange attractors from time series. In Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[17] Floris Takens. Detecting strange attractors in turbulence. In David Rand and Lai-Sang Young, editors, Dynamical Systems and Turbulence, Warwick 1980, Lecture Notes in Mathematics, volume 898, pages 366–381. Springer-Verlag, Berlin Heidelberg, 1981. ISBN 978-3-540-11171-9. URL http://www.springerlink.com/index/10.1007/BFb0091924. Visited on 2023-04-20.

[18] F. Yakuwa, Y. Dote, M. Yoneyama, and S. Uzurabashi. Novel time series analysis and prediction of stock trading using fractal theory and time delayed neural network. In SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483), volume 1, pages 134–141 vol.1, 2003. doi: 10.1109/ICSMC.2003.1243804. URL https://ieeexplore.ieee.org/document/1243804. Visited on 2023-04-20.

[19] Wei Sun and Yuwei Wang. Short-term wind speed forecasting based on fast ensemble empirical mode decomposition, phase space reconstruction, sample entropy and improved back-propagation neural network. Energy Conversion and Management, 157:1–12, February 2018. ISSN 0196-8904. doi: 10.1016/j.enconman.2017.11.067. URL https://www.sciencedirect.com/science/article/pii/S0196890417311196. Visited on 2023-04-20.

[20] Sang-Hong Lee, Kyung-Yong Chung, and Joon S. Lim. Detection of ventricular fibrillation using hilbert transforms, phase-space reconstruction, and time-domain analysis. Personal and Ubiquitous Computing, 18(6):1315–1324, Aug 2014. ISSN 1617-4917. doi: 10.1007/s00779-013-0735-2. URL https://link.springer.com/article/10.1007/s00779-013-0735-2. Visited on 2023-04-20.

[21] Gerhard Moitzi, Reinhard W. Neugschwandtner, Hans-Peter Kaul, and Helmut Wagentristl. Energy Efficiency of Continuous Rye, Rotational Rye and Barley in

Different Fertilization Systems in a Long-Term Field Experiment. Agronomy, 11 (2), 2021. ISSN 2073-4395. doi: 10.3390/agronomy11020229. URL https://www.mdpi.com/2073-4395/11/2/229. Visited on 2023-04-20.

[22] James Gleick. Chaos: Making a New Science. Penguin Books USA, New York, 1987. ISBN 0-14-009250-1.

[23] M. Mitchell Waldrop. Complexity: the emerging science at the edge of order and chaos. Simon & Schuster, New York, 1992. ISBN 0671767895. URL https://www.bibsonomy.org/bibtex/29277b73e5affb35edd5707bd556993c5/jrennstich. Visited on 2023-04-20.

[24] Daniel Toker, Friedrich T. Sommer, and Mark D'Esposito. A simple method for detecting chaos in nature. Communications Biology, 3(1):11, Jan 2020. ISSN 2399-3642. doi: 10.1038/s42003-019-0715-9. URL https://doi.org/10.1038/s42003-019-0715-9. Visited on 2023-04-20.

[25] Tanya L. Rogers, Bethany J. Johnson, and Stephan B. Munch. Chaos is not rare in natural ecosystems. Nature Ecology & Evolution, Jun 2022. ISSN 2397-334X. doi: 10.1038/s41559-022-01787-y. URL https://doi.org/10.1038/s41559-022-01787-y. Visited on 2023-04-20.

[26] Artemios-Anargyros Semenoglou, Evangelos Spiliotis, and Vassilios Assimakopoulos. Data augmentation for univariate time series forecasting with neural networks. Pattern Recognition, 134:109132, 2023. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2022.109132. URL https://www.sciencedirect.com/science/article/pii/S0031320322006124. Visited on 2023-04-20.

[27] Polychronis Manousopoulos, Vassileios Drakopoulos, and Theoharis Theoharis. Curve Fitting by Fractal Interpolation, pages 85–103. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-79299-4. doi: 10.1007/978-3-540-79299-4_4. URL https://link.springer.com/chapter/10.1007/978-3-540-79299-4_4. Visited on 2023-04-20.

[28] Sebastian Raubitzek and Thomas Neubauer. A fractal interpolation approach to improve neural network predictions for difficult time series data. Expert Systems with Applications, 169:114474, 2021. ISSN 0957-4174. doi: 10.1016/j.eswa.2020.114474. URL http://www.sciencedirect.com/science/article/pii/S0957417420311234. Visited on 2023-04-20.

[29] Jan Friedrich, Sebastian Gallon, Alain Pumir, and Rainer Grauer. Stochastic Interpolation of Sparsely Sampled Time Series via Multipoint Fractional Brownian Bridges. Physical Review Letters, 125(17):170602, 2020. Publisher: APS.

[30] Sebastian Raubitzek, Thomas Neubauer, Jan Friedrich, and Andreas Rauber. Interpolating strange attractors via fractional brownian bridges. Entropy, 24(5), 2022. ISSN 1099-4300. doi: 10.3390/e24050718. URL https://www.mdpi.com/1099-4300/24/5/718. Visited on 2023-04-20.

[31] Sebastian Raubitzek and Thomas Neubauer. Reconstructed phase spaces and lstm neural network ensemble predictions. Engineering Proceedings, 18(1), 2022. ISSN 2673-4591. doi: 10.3390/engproc2022018040. URL https://www.mdpi.com/2673-4591/18/1/40. Visited on 2023-04-20.

[32] Sebastian Raubitzek and Thomas Neubauer. Taming the Chaos in Neural Network Time Series Predictions. Entropy, 23(11), 2021. ISSN 1099-4300. doi: 10.3390/e23111424. URL https://www.mdpi.com/1099-4300/23/11/1424. Visited on 2023-04-20.

[33] María Antonia Navascués, Arya Kumar Bedabrata Chand, Viswanathan Puthan Veedu, and María Victoria Sebastián. Fractal Interpolation Functions: A Short Survey. Applied Mathematics, 5:1834–1841, 2014. doi: 10.4236/am.2014.512176.

[34] Joseph L. McCauley, Gemunu H. Gunaratne, and Kevin E. Bassler. Hurst exponents, markov processes, and fractional brownian motion. Physica A: Statistical Mechanics and its Applications, 379(1):1–9, 2007. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2006.12.028. URL https://www.sciencedirect.com/science/article/pii/S0378437106013483. Visited on 2023-04-20.

[35] Valeria Bondarenko, Victor Bondarenko, and Kyryl Truskovskyi. Forecasting of time data with using fractional brownian motion. Chaos, Solitons & Fractals, 97: 44–50, 2017. ISSN 0960-0779. doi: https://doi.org/10.1016/j.chaos.2017.01.013. URL https://www.sciencedirect.com/science/article/pii/S096007791730019X. Visited on 2023-04-20.

[36] Jingtao Yao, Chew Lim Tan, and Hean-Lee Poh. Neural Networks for Technical Analysis: A Study on KLCI. International Journal of Theoretical and Applied Finance, 2(2):221–241, 1999. doi: 10.1142/S0219024999000145. URL https://www.worldscientific.com/doi/abs/10.1142/S0219024999000145. Visited on 2023-04-20.

[37] Somesh Selvaratnam and Michael Kirley. Predicting stock market time series using evolutionary artificial neural networks with hurst exponent input windows. In Proceedings of the 19th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence, AI'06, page 617–626, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3540497870. doi: 10.1007/11941439_66.

[38] Bo Qian and Khaled Rasheed. Stock market prediction with multiple classifiers. Applied Intelligence, 26(1):25–33, feb 2007. ISSN 0924-669X. doi: 10.1007/s10489-006-0001-7.

[39] Indranil Ghosh, Manas K. Sanyal, and R. K. Jana. Fractal Inspection and Machine Learning-Based Predictive Modelling Framework for Financial Markets. Arabian Journal for Science and Engineering, page 15, November 2017. ISSN 2191-4281. doi: 10.1007/s13369-017-2922-3.

[40] João Nunes De Mendonça Neto, Luiz Paulo Lopes Fávero, and Renata Turola Takamatsu. Hurst exponent, fractals and neural networks for forecasting financial asset returns in Brazil. International Journal of Data Science and Analytics, 3(1), 2018. doi: 10.1504/IJDS.2018.10011821.

[41] C.L. Chang, S.L. Lo, and S.L. Yu. Applying fuzzy theory and genetic algorithm to interpolate precipitation. Journal of Hydrology, 314(1):92–104, November 2005. ISSN 0022-1694. doi: 10.1016/j.jhydrol.2005.03.034. URL https://www.sciencedirect.com/science/article/pii/S0022169405001484. Visited on 2023-04-20.

[42] Yeliz Karaca and Dumitru Baleanu. A Novel R/S Fractal Analysis and Wavelet Entropy Characterization Approach for Robust Forecasting Based on Self-Similar Time Series Modelling. Fractals, 28, May 2020. doi: 10.1142/S0218348X20400320.

[43] Yeliz Karaca, Yu-Dong Zhang, and Khan Muhammad. Characterizing Complexity and Self-Similarity Based on Fractal and Entropy Analyses for Stock Market Forecast Modelling. Expert Systems with Applications, 144:113098, April 2020. ISSN 0957-4174. doi: 10.1016/j.eswa.2019.113098. URL https://www.sciencedirect.com/science/article/pii/S0957417419308152. Visited on 2023-04-20.

[44] Li-Ping Ni, Zhi-Wei Ni, and Ya-Zhuo Gao. Stock trend prediction based on fractal feature selection and support vector machine. Expert Systems with Applications, 38(5):5569–5576, 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2010.10.079. URL

https://www.sciencedirect.com/science/article/pii/S0957417410012236. Visited on 2023-04-20.

[45] T. Di Matteo. Multi-scaling in finance. Quantitative Finance, 7(1):21–36, 2007. doi: 10.1080/14697680600969727.

[46] T. Higuchi. Approach to an irregular time series on the basis of the fractal theory. Physica D: Nonlinear Phenomena, 31(2):277–283, 1988. ISSN 0167-2789. doi: 10.1016/0167-2789(88)90081-4. URL https://www.sciencedirect.com/science/article/pii/0167278988900814. Visited on 2023-04-20.

[47] A. Petrosian. Kolmogorov complexity of finite sequences and recognition of different preictal EEG patterns. In Proceedings Eighth IEEE Symposium on Computer-Based Medical Systems, pages 212–217, 1995. doi: 10.1109/CBMS.1995.465426. URL https://ieeexplore.ieee.org/document/465426. Visited on 2023-04-20.

[48] Michael J. Katz. Fractals and the analysis of waveforms. Computers in Biology and Medicine, 18(3):145–156, January 1988. ISSN 0010-4825. doi: 10.1016/0010-4825(88)90041-8. URL https://www.sciencedirect.com/science/article/pii/0010482588900418. Visited on 2023-04-20.

[49] Jean-Pierre Eckmann, Sylvie Kamphorst, and Sergio Ciliberto. Liapunov exponents from time series. Physical review. A, 34:4971–4979, January 1987. doi: 10.1103/PhysRevA.34.4971.

[50] Teresa Henriques, Maria Ribeiro, Andreia Teixeira, Luísa Castro, Luís Antunes, and Cristina Costa-Santos. Nonlinear methods most applied to heart-rate time series: A review. Entropy, 22(3), 2020. ISSN 1099-4300. doi: 10.3390/e22030309. URL https://www.mdpi.com/1099-4300/22/3/309. Visited on 2023-04-20.

[51] Liu Zengrong, Chen Liqun, and Yang Ling. On properties of hyperchaos: Case study. Acta Mechanica Sinica, 15(4):366–370, November 1999. ISSN 1614-3116. doi: 10.1007/BF02487934. URL https://link.springer.com/article/10.1007/BF02487934. Visited on 2023-04-20.

[52] O.E. Rossler. An equation for hyperchaos. Physics Letters A, 71(2):155–157, 1979. ISSN 0375-9601. doi: https://doi.org/10.1016/0375-9601(79)90150-6. URL https://www.sciencedirect.com/science/article/pii/0375960179901506. Visited on 2023-04-20.

[53] Claude E Shannon. A mathematical theory of communication. The Bell system technical journal, 27(3):379–423, 1948.

[54] Oldrich Zmeskal, Petr Dzik, and Michal Vesely. Entropy of fractal systems. Computers & Mathematics with Applications, 66(2):135–146, 2013. ISSN 0898-1221. doi: 10.1016/j.camwa.2013.01.017. URL https://www.sciencedirect.com/science/article/pii/S0898122113000345. Visited on 2023-04-20.

[55] Dominique Makowski, Tam Pham, Zen J. Lau, Jan C. Brammer, François Lespinasse, Hung Pham, Christopher Schölzel, and Annabel S H Chen. Neurokit2: A python toolbox for neurophysiological signal processing, 2020. URL https://github.com/neuropsychology/NeuroKit. Visited on 2023-04-20.

[56] Akira Ishikawa and Hiroshi Mieno. The fuzzy entropy concept and its application. Fuzzy Sets and Systems, 2(2):113–123, 1979. ISSN 0165-0114. doi: https://doi.org/10.1016/0165-0114(79)90020-4. URL https://www.sciencedirect.com/science/article/pii/0165011479900204. Visited on 2023-04-20.

[57] Petre Caraiani. The predictive power of singular value decomposition entropy for stock market dynamics. Physica A: Statistical Mechanics and its Applications, 393:571–578, 2014. ISSN 0378-4371. doi: 10.1016/j.physa.2013.08.071. URL https://www.sciencedirect.com/science/article/pii/S0378437113008212. Visited on 2023-04-20.

[58] Rongbao Gu and Yanmin Shao. How long the singular value decomposed entropy predicts the stock market? — evidence from the dow jones industrial average index. Physica A: Statistical Mechanics and its Applications, 453:150–161, 2016. ISSN 0378-4371. doi: 10.1016/j.physa.2016.02.030. URL https://www.sciencedirect.com/science/article/pii/S0378437116001965. Visited on 2023-04-20.

[59] G. Espinosa-Paredes, E. Rodriguez, and J. Alvarez-Ramirez. A singular value decomposition entropy approach to assess the impact of covid-19 on the informational efficiency of the wti crude oil market. Chaos, Solitons & Fractals, 160:112238, 2022. ISSN 0960-0779. doi: https://doi.org/10.1016/j.chaos.2022.112238. URL https://www.sciencedirect.com/science/article/pii/S0960077922004489. Visited on 2023-04-20.

[60] S. J. Roberts, W. Penny, and I. Rezek. Temporal and spatial complexity measures for electroencephalogram based brain-computer interfacing. Medical &

Biological Engineering & Computing, 37(1):93–98, January 1999. ISSN 1741-0444. doi: 10.1007/BF02513272.

[61] Cleve B. Moler. Numerical Computing with Matlab. Society for Industrial and Applied Mathematics, 2004. doi: 10.1137/1.9780898717952. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898717952. Visited on 2023-04-20.

[62] Audrey L. Mayer, Christopher W. Pawlowski, and Heriberto Cabezas. Fisher Information and dynamic regime changes in ecological systems. Ecological Modelling, 195(1):72 – 82, 2006. ISSN 0304-3800. doi: 10.1016/j.ecolmodel.2005.11.011. URL http://www.sciencedirect.com/science/article/pii/S030438000500579X. Visited on 2023-04-20.

[63] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a Time Series. Phys. Rev. Lett., 45(9):712–716, September 1980. doi: 10.1103/PhysRevLett.45.712. URL https://link.aps.org/doi/10.1103/PhysRevLett.45.712. Visited on 2023-04-20.

[64] Andrew M. Fraser and Harry L. Swinney. Independent coordinates for strange attractors from mutual information. Phys. Rev. A, 33(2):1134–1140, February 1986. doi: 10.1103/PhysRevA.33.1134. URL https://link.aps.org/doi/10.1103/PhysRevA.33.1134. Visited on 2023-04-20.

[65] Sebastian Wallot and Dan Mønster. Calculation of Average Mutual Information (AMI) and False-Nearest Neighbors (FNN) for the Estimation of Embedding Parameters of Multidimensional Time Series in Matlab. Frontiers in Psychology, 9, 2018. ISSN 1664-1078. doi: 10.3389/fpsyg.2018.01679. URL https://www.frontiersin.org/article/10.3389/fpsyg.2018.01679. Visited on 2023-04-20.

[66] C. J. Cellucci, A. M. Albano, and P. E. Rapp. Statistical validation of mutual information calculations: Comparison of alternative numerical algorithms. Phys. Rev. E, 71:066208, Jun 2005. doi: 10.1103/PhysRevE.71.066208. URL https://link.aps.org/doi/10.1103/PhysRevE.71.066208. Visited on 2023-04-20.

[67] Patrick F. Dunn. Measurement and Data Analysis for Engineering and Science. CRC Press, Boca Raton, 3 edition, May 2014. ISBN 978-0-429-16916-8. doi: 10.1201/b16918.

[68] Matthew B. Kennel, Reggie Brown, and Henry D. I. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. Phys. Rev. A, 45:3403–3411, Mar 1992. doi: 10.1103/PhysRevA.45.3403. URL https://link.aps.org/doi/10.1103/PhysRevA.45.3403. Visited on 2023-04-20.

[69] Carl Rhodes and Manfred Morari. The false nearest neighbors algorithm: An overview. Computers & Chemical Engineering, 21:S1149–S1154, 1997. ISSN 0098-1354. doi: 10.1016/S0098-1354(97)87657-0. URL https://www.sciencedirect.com/science/article/pii/S0098135497876570. Visited on 2023-04-20.

[70] Tero Aittokallio, M Gyllenberg, Jarmo Hietarinta, T Kuusela, and T Multamäki. Improving the false nearest neighbors method with graphical analysis. Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics, 60:416–21, August 1999. doi: 10.1103/PhysRevE.60.416.

[71] Michael F. Barnsley, Stephen Demko, and Michael James David Powell. Iterated function systems and the global construction of fractals. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 399(1817): 243–275, June 1985. doi: 10.1098/rspa.1985.0057. URL https://royalsocietypublishing.org/doi/10.1098/rspa.1985.0057. Visited on 2023-04-20.

[72] D.S. Mazel and M.H. Hayes. Using iterated function systems to model discrete sequences. IEEE Transactions on Signal Processing, 40(7):1724–1734, 1992. doi: 10.1109/78.143444. URL https://ieeexplore.ieee.org/document/143444. Visited on 2023-04-20.

[73] L. Dalla and V. Drakopoulos. Regular article: On the parameter identification problem in the plane and the polar fractal interpolation functions. J. Approx. Theory, 101(2):289–302, dec 1999. ISSN 0021-9045. doi: 10.1006/jath.1999.3380.

[74] Jan W. Kantelhardt. Fractal and Multifractal Time Series, 2008. doi: 10.48550/arXiv.0804.0747. URL https://arxiv.org/abs/0804.0747. Visited on 2023-04-20.

[75] Maria Antonia Navascués. Fractal Polynomial Interpolation. ZEITSCHRIFT FÜR ANALYSIS UND IHRE ANWENDUNGEN, 24(2):401–418, 2005. doi: 10/dfcdhd.

[76] B. B. Mandelbrot. The fractal geometry of nature. W. H. Freeman and Comp., New York, 3 edition, 1983.

[77] Jens Feder. Fractals. Physics of Solids and Liquids. Springer US, New York, 1988. ISBN 0-306-42851-2.

[78] Michael F. Barnsley and Mathematics. Fractals Everywhere. Dover Publications, Inc., USA, 2012. ISBN 0486488705.

[79] Sebastian Raubitzek. Fractal Interpolation 2023, 3 2023. URL https://github.com/Raubkatz/fractal_interpolation_2023. Visited on 2023-04-20.

[80] Sebastian Raubitzek. PhaSpaSto Interpolation 2023, 3 2023. URL https://github.com/Raubkatz/PhaSpaSto_interpolation_2023. Visited on 2023-04-20.

[81] J.L. Pech-Pacheco, G. Cristobal, J. Chamorro-Martinez, and J. Fernandez-Valdivia. Diatom autofocusing in brightfield microscopy: a comparative study. In Proceedings 15th International Conference on Pattern Recognition. ICPR-2000, volume 3, pages 314–317 vol.3, 2000. doi: 10.1109/ICPR.2000.903548.

[82] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. Numerical Mathematics, volume 37. Springer Berlin, Heidelberg, 2 edition, January 2007. ISBN 978-1-4757-7394-1. doi: 10.1007/b98885.

[83] A.K. Jain, Jianchang Mao, and K.M. Mohiuddin. Artificial neural networks: a tutorial. Computer, 29(3):31–44, 1996. doi: 10.1109/2.485891.

[84] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. Heliyon, 4(11):e00938, November 2018. ISSN 2405-8440. doi: 10.1016/j.heliyon.2018.e00938. URL https://www.sciencedirect.com/science/article/pii/S2405844018332067. Visited on 2023-04-20.

[85] Henry J. Kelley. Gradient theory of optimal flight paths. ARS Journal, 30(10): 947–954, 1960. doi: 10.2514/8.5282.

[86] Ahmed Tealab. Time series forecasting using artificial neural networks methodologies: A systematic review. Future Computing and Informatics Journal, 3(2):334–340, December 2018. ISSN 2314-7288. doi: 10.1016/j.fcij.2018.10.003. URL

https://www.sciencedirect.com/science/article/pii/S2314728817300715. Visited on 2023-04-20.

[87] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. Neural Computation, 9:1735–80, December 1997. doi: 10.1162/neco.1997.9.8.1735.

[88] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL https://aclanthology.org/W14-4012. Visited on 2023-04-20.

[89] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, 2014. doi: 10.48550/arXiv.1412.3555. URL https://arxiv.org/abs/1412.3555. Visited on 2023-04-20.

[90] Nicole Gruber and Alfred Jockisch. Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text? Frontiers in Artificial Intelligence, 3, 06 2020. doi: 10.3389/frai.2020.00040.

[91] Christopher Olah. Understanding lstm networks, 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Visited on 2023-04-20.

[92] Steven Brunton, Bingni Brunton, Joshua Proctor, Eurika Kaiser, and J. Kutz. Chaos as an Intermittently Forced Linear System. Nature Communications, 8, August 2016. doi: 10.1038/s41467-017-00030-8.

[93] Domingos S. de O. Santos Júnior, João F.L. de Oliveira, and Paulo S.G. de Mattos Neto. An intelligent hybridization of arima with machine learning models for time series forecasting. Knowledge-Based Systems, 175:72–86, 2019. ISSN 0950-7051. doi: 10.1016/j.knosys.2019.03.011. URL https://www.sciencedirect.com/science/article/pii/S0950705119301327. Visited on 2023-04-20.

[94] R.J. Hyndman and Yangzhuoran Yang. Time Series Data Library v0.1.0., 2018. URL https://pkg.yangzhuoranyang.com/tsdl/. Visited on 2023-04-20.

[95] Wayne P. London and James A. Yorke. Recurrent Outbreaks of Measles, Chickenpox and Mumps: I. Seasonal Variation in Contact Rates. American Journal of Epidemiology, 98(6):453–468, 12 1973. ISSN 0002-9262. doi: 10.1093/oxfordjournals.aje.a121575.

[96] Steven Wheelwright, Spyros Makridakis, and Rob J Hyndman. Forecasting: methods and applications. John Wiley & Sons, Hoboken, New Jersey, 1998. ISBN 0471532339.

[97] U.S. dollars to U.K. pound Sterling Spot Exchange Rate. Federal Reserve Bank of St. Louis, 2022. URL https://fred.stlouisfed.org/series/DEXUSUK. Visited on 2023-04-20.

[98] The Global Runoff Data Centre, 56068 Koblenz, Germany. Federal Institute of Hydrology (BfG), 2022. URL https://www.bafg.de/GRDC/EN/Home/homepage_node.html. Visited on 2023-04-20.

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

[100] Peter Albrecht. The runge-kutta theory in a nutshell. SIAM J. Numer. Anal., 33 (5):1712–1735, oct 1996. ISSN 0036-1429. doi: 10.1137/S0036142994260872.

[101] Charles A Hall and W.Weston Meyer. Optimal error bounds for cubic spline interpolation. Journal of Approximation Theory, 16(2):105–122, 1976. ISSN 0021-9045. doi: 10.1016/0021-9045(76)90040-X. URL https://www.sciencedirect.com/science/article/pii/002190457690040X. Visited on 2023-04-20.

[102] Mathieu Lepot, Jean-Baptiste Aubin, and François H.L.R. Clemens. Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. Water, 9(10), 2017. ISSN 2073-4441. doi: 10.3390/w9100796. URL https://www.mdpi.com/2073-4441/9/10/796. Visited on 2023-04-20.

[103] Tilmann Gneiting and Martin Schlather. Stochastic models that separate fractal dimension and the hurst effect. SIAM Review, 46(2):269–282, 2004. ISSN 00361445. URL http://www.jstor.org/stable/20453506. Visited on 2023-04-20.

[104] Sumit Kumar, Lasani Hussain, Sekhar Banarjee, and Motahar Reza. Energy load forecasting using deep learning approach-lstm and gru in spark cluster. In 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT), pages 1–4, 2018. doi: 10.1109/EAIT.2018.8470406.

[105] V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. IEEE Transactions on Automatic Control, 25(2):164–176, April 1980. ISSN 1558-2523. doi: 10.1109/TAC.1980.1102314.

[106] Sondo Kim, Seungmo Ku, Woojin Chang, and Jae Wook Song. Predicting the direction of us stock prices using effective transfer entropy and machine learning techniques. IEEE Access, 8:111660–111682, 2020. doi: 10.1109/ACCESS.2020.3002174.

[107] SciPy. Interpolation (scipy.interpolate) — SciPy v1.8.0 Manual, 2022. URL https://docs.scipy.org/doc/scipy/tutorial/interpolate.html. Visited on 2023-04-20.

[108] Carl de Boor. A Practical Guide to Splines, volume 27. Springer, New York, 01 1978. doi: 10.2307/2006241.

[109] Pierre Dubois, Thomas Gomez, Laurent Planckaert, and Laurent Perret. Data-driven predictions of the lorenz system. Physica D: Nonlinear Phenomena, 408:132495, 2020. ISSN 0167-2789. doi: 10.1016/j.physd.2020.132495. URL https://www.sciencedirect.com/science/article/pii/S0167278919307080. Visited on 2023-04-20.

[110] Vishvesh Shah. A comparative study of univariate time-series methods for sales forecasting. Master's thesis, University of Waterloo, 2020. URL https://uwspace.uwaterloo.ca/handle/10012/15488. Visited on 2023-04-20.

[111] G.Peter Zhang. Time series forecasting using a hybrid arima and neural network model. Neurocomputing, 50:159–175, 2003. ISSN 0925-2312. doi: 10.1016/S0925-2312(01)00702-0. URL https://www.sciencedirect.com/science/article/pii/S0925231201007020. Visited on 2023-04-20.

[112] Mehdi Khashei and Mehdi Bijari. An artificial neural network (p,d,q) model for timeseries forecasting. Expert Systems with Applications, 37(1):479–489, 2010. ISSN 0957-4174. doi: 10.1016/j.eswa.2009.05.044. URL https://www.sciencedirect.com/science/article/pii/S0957417409004850. Visited on 2023-04-20.

[113] Mehdi Khashei and Mehdi Bijari. A novel hybridization of artificial neural networks and arima models for time series forecasting. Applied Soft Computing, 11(2):2664–2675, 2011. ISSN 1568-4946. doi: 10.1016/j.asoc.2010.10.015. URL https://www.sciencedirect.com/science/article/pii/S1568494610002759. Visited on 2023-04-20.

[114] João Fausto Lorenzato de Oliveira and Teresa Bernarda Ludermir. A hybrid evolutionary system for parameter optimization and lag selection in time series

forecasting. In 2014 Brazilian Conference on Intelligent Systems, pages 73–78, 2014. doi: 10.1109/BRACIS.2014.24.

[115] C. Narendra Babu and B. Eswara Reddy. A moving-average filter based hybrid arima–ann model for forecasting time series data. Applied Soft Computing, 23: 27–38, 2014. ISSN 1568-4946. doi: 10.1016/j.asoc.2014.05.028. URL https://www.sciencedirect.com/science/article/pii/S1568494614002555. Visited on 2023-04-20.

[116] João F.L. de Oliveira and Teresa B. Ludermir. A hybrid evolutionary decomposition system for time series forecasting. Neurocomputing, 180:27–34, 2016. ISSN 0925-2312. doi: 10.1016/j.neucom.2015.07.113. URL https://www.sciencedirect.com/science/article/pii/S0925231215016057. Visited on 2023-04-20.

[117] Sibarama Panigrahi and H.S. Behera. A hybrid ets–ann model for time series forecasting. Engineering Applications of Artificial Intelligence, 66:49–59, 2017. ISSN 0952-1976. doi: 10.1016/j.engappai.2017.07.007. URL https://www.sciencedirect.com/science/article/pii/S0952197617301550. Visited on 2023-04-20.

[118] Paulo S.G. de Mattos Neto, George D.C. Cavalcanti, and Francisco Madeiro. Nonlinear combination method of forecasters applied to pm time series. Pattern Recognition Letters, 95:65–72, 2017. ISSN 0167-8655. doi: 10.1016/j.patrec.2017.06.008. URL https://www.sciencedirect.com/science/article/pii/S0167865517302143. Visited on 2023-04-20.

[119] C.-K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger. Mosaic organization of DNA nucleotides. Phys. Rev. E, 49(2): 1685–1689, February 1994. doi: 10.1103/PhysRevE.49.1685. URL https://link.aps.org/doi/10.1103/PhysRevE.49.1685. Visited on 2023-04-20.

[120] P. Abry and D. Veitch. Wavelet analysis of long-range-dependent traffic. IEEE Transactions on Information Theory, 44(1):2–15, 1998. doi: 10.1109/18.650984.

[121] Cheolhwan Oh, Seungmin Han, and Jongpil Jeong. Time-series data augmentation based on interpolation. Procedia Computer Science, 175:64–71, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.07.012. URL https://www.sciencedirect.com/science/article/pii/S1877050920316914. Visited on 2023-04-20.

[122] Nicolas Morizet, Matteo Rizzato, David Grimbert, and George Luta. A pilot study on the use of generative adversarial networks for data augmentation of time series. AI, 3(4):789–795, 2022. ISSN 2673-2688. doi: 10.3390/ai3040047. URL https://www.mdpi.com/2673-2688/3/4/47. Visited on 2023-04-20.

[123] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1):67–82, 1997. doi: 10.1109/4235.585893.

[124] Sanjay Vasant Dudul. Prediction of a lorenz chaotic attractor using two-layer perceptron neural network. Applied Soft Computing, 5(4):333–355, 2005. ISSN 1568-4946. doi: 10.1016/j.asoc.2004.07.005. URL https://www.sciencedirect.com/science/article/pii/S1568494604000730. Visited on 2023-04-20.

[125] J. S. Richman and J. R. Moorman. Physiological time-series analysis using approximate entropy and sample entropy. American journal of physiology. Heart and circulatory physiology, 278(6):H2039–H2049, 2000. doi: 10.0000/PMID10843903.

[126] Amir Omidvarnia, Mostefa Mesbah, Mangor Pedersen, and Graeme Jackson. Range Entropy: A Bridge between Signal Complexity and Self-Similarity. Entropy, 20(962), 2018. doi: 10/ggtqpt.

[127] Ladislav Kristoufek and Miloslav Vosvrda. Measuring capital market efficiency: long term memory, fractal dimension and approximate entropy. The European Physical Journal B, 87(7), 2014. doi: 10/ggtqpz.

[128] Sebastian Raubitzek and Thomas Neubauer. Machine Learning and Chaos Theory in Agriculture. ERCIM News, 122, July 2020.

[129] Sebastian Raubitzek and Thomas Neubauer. Combining measures of signal complexity and machine learning for time series analyis: A review. Entropy, 23 (12), 2021. ISSN 1099-4300. doi: 10.3390/e23121672. URL https://www.mdpi.com/1099-4300/23/12/1672. Visited on 2023-04-20.

[130] Sebastian Raubitzek and Thomas Neubauer. An exploratory study on the complexity and machine learning predictability of stock market data. Entropy, 24(3), 2022. ISSN 1099-4300. doi: 10.3390/e24030332. URL https://www.mdpi.com/1099-4300/24/3/332. Visited on 2023-04-20.

[131] Kevin Mallinger, Sebastian Raubitzek, Thomas Neubauer, and Steven Lade. Potentials and limitations of complexity metrics for the sustainable transition to

Farming 4.0. Current Opinion in Environmental Sustainability, 2022. Accepted, not yet published.

[132] Sebastian Raubitzek, Kevin Mallinger, and Thomas Neubauer. Combining fractional derivatives and machine learning: A review. Entropy, 25(1), 2023. ISSN 1099-4300. doi: 10.3390/e25010035. URL https://www.mdpi.com/1099-4300/25/1/35. Visited on 2023-04-20.