Universidad Michoacana de San Nicolás de Hidalgo

División de Estudios de Posgrado de la
Facultad de Ingeniería Eléctrica

NEAREST NEIGHBORS ALGORITHMS FOR NOISY AND CHAOTIC TIME SERIES FORECASTING

TESIS

Que para obtener el grado de

DOCTOR EN CIENCIAS EN INGENIERÍA ELÉCTRICA

Presenta
M.C. José Rafael Cedeño González
jose.cedeno@umich.mx

Director de Tesis
Dr. Juan José Flores Romero
juanf@umich.mx

Morelia, Michoacán. Septiembre 2019.

Agradezco al CONACYT por el apoyo otorgado a través de la beca No. 516226/290379.
Agradezco a la División de Estudios de Posgrado de La Universidad Michoacana de San Nicolás de Hidalgo por haberme aceptado en su programa de doctorado.
Esta tesis está dedicada a mi hija por ser mi inspiración para conseguir este y muchos logros. La dedico también a mi madre por habérmelo dado todo.

II

Resumen

Esta tesis presenta un nuevo método de pronóstico basado en el algoritmo de Vecinos Cercanos el cual, usando Evolución Diferencial, es capaz de obtener los mejores parámetros para producir pronósticos precisos de series de tiempo caóticas y en presencia de ruido. Este nuevo método se llamó NNDE. Se analizaron varios casos de estudio comparando diferentes métodos de pronóstico y en la mayoría de los casos NNDE obtuvo resultados sustancialmente mejores que el resto de los métodos.

Palabras clave: predicción de series de tiempo, algoritmo de vecinos cercanos, algoritmos evolutivos

Abstract

This thesis presents a new forecasting method based on the Nearest Neighbors algorithm that, with the use of Differential Evolution, is able to obtain the best parameters to produce accurate forecasts of chaotic time series with the presence of noise. This new method is called NNDE. Several case studies were analyzed comparing different forecasting methods and in most cases NNDE obtains substantially better results than the rest of the methods.

Keywords: time series forecasting, nearest neighbors algorithm, evolutionary algorithms

Publications

In This Thesis

Some of the research leading to this thesis has appeared previously in the following publications.

Journal Articles

- Juan J. Flores, José R. Cedeño González, Héctor Rodríguez, Mario Graff, Rodrigo Lopez-Farias, Felix Calderon: **Soft Computing Methods with Phase Space Reconstruction for Wind Speed Forecasting - A Performance Comparison**. – *Energies*, September 2019.

- Juan J. Flores, José R. Cedeño González, Rodrigo Lopez-Farias, Felix Calderon: **Evolving nearest neighbor time series forecasters**. – *Soft Computing*, September 2017.

Conference Papers

- Juan J. Flores, Felix Calderon, Elisa Espinoza, José R. Cedeño González, Adan Garnica, Georgina Flores: **Fuzzy Nearest Neighbor Time Series Forecasting - Computational Complexity**. – *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, December 2016, Las Vegas, U.S.A.

- Juan J. Flores, Felix Calderon, José R. Cedeño González, Jose Ortiz, Rodrigo Lopez-Farias: **Comparison of time series forecasting techniques with respect to tolerance to noise**. – *2016 IEEE*

International Autumn Meeting on Power, Electronics and Computing (ROPEC), November 2016, Ixtapa, México.

- Juan J. Flores, Jose Ortiz, José R. Cedeño González, Carlos Lara, Rodrigo Lopez-Farias: **FNN a fuzzy version of the nearest neighbor time series forecasting technique**. – *2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, November 2015, Ixtapa, México.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ACF Auto-correlation Function. 17, 18, 49, 50, 52, 54, 55, 57, 76, 89, 90, 103, 104

ADF Augmented Dickey-Fuller. 21, 50, 53, 55, 58, 75, 89, 105

ANFIS Adaptive Neuro-Fuzzy Inference System. 70

ANN Artificial Neural Network. 12–15, 47, 48, 59–64, 70, 71, 77, 81, 86–88, 102

AR Auto-regressive. 47, 71

ARIMA Auto-regressive Integrated Moving Average. 47, 48, 59–61, 64, 70, 71, 77, 86, 91, 98

ARMA Auto-regressive Moving Average. 71, 86

CENACE Centro Nacional de Control de Energía. 101, 107, 108, 113

cGA Compact Genetic Algorithm. 71, 77

DA-RNN Dual-Stage Attention-Based Recurrent Neural Network. 107, 108, 110, 112

DE Differential Evolution. 30–33, 35, 36, 42–44, 48, 60, 63, 78, 92, 108

EA Evolutionary Algorithms. 30, 48

EvoDAG Evolving Directed Acyclic Graph. 71, 77, 78

FF Fuzzy Forecast. 71, 77, 78, 81

GPU Graphic Processing Unit. 117

GWPPT Generalized Wind Power Prediction Tool. 70, 71

LSTM Long Short-term Memory. 47, 61, 86, 97, 98

MA Moving Average. 71

MAE Mean Absolute Error. 26, 27, 108

MAPE Mean Absolute Percentage Error. 26–28, 31, 33, 60–64, 70, 102, 103, 116

MLE Maximum Lyapunov Exponent. 51, 53, 55, 56, 58, 76, 90, 91, 100, 106, 116

MLP Multi-layer Perceptron. 78, 107–111, 113

MSE Mean Squared Error. 26, 27, 31, 33, 84

NARX Nonlinear Autoregressive Exogenous Artificial Neural Networks. 70

NN Nearest Neighbors. 12, 15, 29–33, 35, 36, 42, 48, 59–61, 64, 71, 77, 78, 91

NNDE Nearest Neighbors by Differential Evolution. 33, 47, 59–64, 68, 69, 72, 77, 78, 81, 83, 84, 93, 96–98, 100, 101, 107–111, 113, 115–118

NSR Noise-to-signal Ratio. 64

NSS N-Step Simultaneous. 4, 30, 31, 59, 61, 62, 64

ODA One Day Ahead. 5, 77, 78, 84, 91, 92, 96

ODE Ordinary Differential Equation. 22

OSA One Step Ahead. 4, 30, 31, 59–62, 64, 78, 81, 84, 91, 92

RBF Radial Basis Function. 47, 59–64

RMSD Root Mean Squared Deviation. 27

RMSE Root Mean Squared Error. 27

RNN Recurrent Neural Network. 47, 59, 61–64

rRMSE Relative Root Mean Squared Error. 87

SENER Secretaría de Energía. 101

SIN Sistema Eléctrico Nacional. 103, 106–108

SMAPE Symmetric Mean Absolute Percentage Error. 26, 28, 31, 78, 81, 82, 84, 92, 93, 116

SNR Signal-to-noise Ratio. 63, 64

SOM Self Organizing Map. 102

SVM Support Vector Machine. 102

UMSNH Universidad Michoacana de San Nicolás de Hidalgo. 101

VAR Vector Autoregressive. 71

WPPT Wind Power Prediction Tool. 70, 71

Chapter 1

Introduction

This chapter introduces the problem of time series forecasting by describing what a time series is, some of its characteristics and what kind of times series will be dealt with in this work. Then a brief review of the state of the art is presented. A series of objectives to be accomplished in this work will be discussed. And finally a description of the chapters is enumerated.

1.1 Problem Definition

In engineering and statistics, a time series is described as a time-dependent sequence of measurements taken at equidistant time intervals of a process or natural phenomenon. Examples of time series are the atmospheric temperature at a certain place, the stock market, wind speed at certain height and location, number of gallons of water passing through a valve, etc. Because of their importance in their respective areas, there is a need to analyze and forecast time series. However several challenges appear when dealing with time series.

Since time is continuous, to model a time series is necessary to define a discrete time interval. This time interval is dependent of the particular phenomenon or process being observed. A very frequent or very infrequent time interval can be detrimental for its analysis. Measuring the water level of a dam every second results in long data sets that with an almost constant value of water level. On the other side, measuring the data usage in a computer network every hour can make you think that the network is almost unused, while during the hour long window several gigabytes of data were consumed.

Time series can be classified in stationary and non-stationary. The stationary time series are those that their mean and variance fluctuates around certain value, while the non-stationary time series tend to increase or decrease its mean value. Stationary time series is required by several time series forecasting methods. In this thesis, all of the time series analyzed are stationary.

A time series can present seasonality for a certain time lag. This is, by using a metric called autocorrelation (the correlation between the time series and itself), if for a certain displacement value of the time series the correlation is close or equal to 1, this means that the time series is following a previously exhibited pattern.

Time series can be contaminated with noise. Noise has diverse origins; instrumental errors and data resolution limits are the most common types of noise. One must consider that noise is always present in time series data.

Furthermore, a time series can be considered chaotic. This is, that a dynamical system of deterministic nature produced a sequence of never repeating values. Chaos should not be confused with noise, since noise is of stochastic nature and generally is not bounded, while a chaotic system is bounded and is controlled by the rules that conform it. In Chapter 2 we will broaden the discussion about these characteristics of time series.

When a time series contains noise and is also chaotic, it becomes very difficult to forecast. Almost all of the known forecasting methods fail to produce adequate results. In this thesis, we will propose a time series forecasting algorithm capable to generate low error forecasts for chaotic and noisy time series. By using the Takens theorem of phase space reconstruction, it is possible to form delay vectors that can be feed to a learning algorithm. This algorithm will capture the patterns present in the time series and allow it to produce forecasts based on the data.

1.2 Time Series Forecasting Concepts

This section describes a few concepts pertained to time series forecasting that will allow to define the problem to solve and the objectives of this thesis.

1.2.1 Modeling

In time series forecasting (and machine learning in general), a model is a program that allows to produce forecasts. These models require certain parameters to be adjusted in order to produce accurate forecasts.

In the classic computational theory, the user feeds a program some input and the program produces data that is useful for the user. However in these days, programs generate an abysmal quantity of data of high complexity. The requirements of the users also have changed; instead of data, the users require to know what value or type the data is going to be.

So, in this new paradigm there exists two phases: a training phase and a forecast phase.

In the training phase, the user feeds the computer a set of data and a model to a learning process. The learning process initializes the model with certain parameters and generates forecasts from instances that their desired value is known. The output of the model is compared to the desired output by using an error metric. The process is repeated with a new set of model parameters in the hope of reducing the error obtained by the previous iteration. This process will adjust the model to make it able to replicate the exhibited data to a certain extent.

Then, in the forecast phase, the user will feed some data to the resulting program (the model), where this program will produce a forecast. For testing and validation purposes, the desired output is also known and the same error metric is used to compare the forecast made against the actual values. In a production environment, the model produces forecasts and the actual output of the underlying process is unknown.

1.2.2 Training, Validation, and Forecast Sets

The training and validation sets are sub-sets of a time series that are used in separated phases of a given experiment.

The training set is used to expose this sub-set of values from the times series to a forecasting method. The forecasting method is aided by a training algorithm to learn the features present in the data and be able to produce forecasts.

The validation set contains the immediate values following the training set. This is, the whole data set is sliced into these two sets (and if required, these sets can be manipulated like shuffling or making further sub-sets). This

sub-set of a time series is used to validate the forecasts produced by a given forecaster. This set cannot be seen or modified by the model being trained until the experiment allows it.

The forecast set is a container where the produced forecasts are stored. This set is the one to be compared with the validation set once the forecasting process is complete. The length of this set may be the same as the validation set.

The length of these sets depends on the number of forecasts to produce. In this thesis the lengths will vary depending of the number of samples available in the time series and the requirements of the experiments; both the validation and forecast set will have the same length as the number of forecasts to be produced.

1.2.3 Forecasting Horizon

The forecasting horizon is the number of time steps in the future for which forecasts are to be produced. Time is considered discrete in time series forecasting. In this thesis the forecasting horizons are equal to 1, this is, the forecast are for the immediate time instant after the end of the training set.

1.2.4 Forecasting Scheme

A forecasting scheme refers to the way the forecasting method will produce their forecasts, which could be simultaneously, iteratively, or a combination of both. Simultaneous forecast are done in a single call of the forecaster. Iterative forecast are made with intermediate forecasts until the desired number of forecasts are obtained.

In this thesis we define three forecasting schemes. One Step Ahead, N-steps simultaneous, and One Day Ahead.

One Step Ahead (OSA) calculates one forecast per call and this prediction is incorporated to the forecast set. This process repeats by putting the corresponding values from the validation set for the forecasted time instant into the training set. Notice that the model only sees the values from the validation set until it has done the predictions and the OSA scheme is simply extending the training set one value at a time.

N-steps simultaneous (NSS) calculate all the desired forecasts in one call. These forecasts are incorporated into the test set.

One Day Ahead (ODA) calculates simultaneously a set of forecasts that comprise a day of observations in the time series. This is, if the time series is sampled hourly, a day of observations contains 24 values. These forecasts are incorporated into the forecast set, but the same amount of samples from the validation set (corresponding to the time period forecasted) are incorporated into the training set. This process repeats until the desired number of days are forecasted.

1.2.5 Forecasting Task

A forecasting task is the sum of the components that participate in the forecasting process (forecasting horizon, forecasting scheme and number of forecasts).

1.3 Problem Definition

The problem to solve is, for a given time series $\mathbb{S} = \{s_1, s_2, \ldots, s_N\}$ that is considered to be noisy and with chaotic features and using a forecasting technique $f_m(\mathbb{S})$ where m is a given forecaster, find the lowest forecasting error ϵ^*_{ft} and the method that produce the lowest forecasting error m^*.

First, to produce a training set, $\text{train}_{\mathbb{S}}$, and a validation set, $\text{val}_{\mathbb{S}}$, the full time series is sliced considering the desired number of forecasts n to make, as shown in Equation (1.1).

$$\text{train}_{\mathbb{S}}, \text{val}_{\mathbb{S}} = \text{slice}(\mathbb{S}, n) \tag{1.1}$$

The validation set $\text{val}_{\mathbb{S}}$ will have a length n and the training set $\text{train}_{\mathbb{S}}$ will have a length of $\text{length}(\mathbb{S}) - n$.

With the training set, a fully trained forecaster $f_m(\cdot)$ will produce a forecast set $\text{forecast}_{\mathbb{S}}$ (Equation (1.2)).

$$\text{forecast}_{\mathbb{S},m} = f_m(\text{train}_{\mathbb{S}}) \tag{1.2}$$

This forecast set will be compared against the validation set using an error metric $\text{error}(\cdot)$ to produce the error ϵ_m achieved by forecaster $f_m(\cdot)$ (Equation (1.3)).

$$\epsilon_m = \text{error}(\text{val}_{\mathbb{S}}, \text{forecast}_{\mathbb{S},m}) \tag{1.3}$$

Then, by using different forecasting methods that compose the forecasting methods set M, a set of forecasting errors E will be produced (Equation (1.4)).

$$E = \{\epsilon_m \mid m \in M\} \tag{1.4}$$

Finally, The lowest forecasting error produced by a forecasting method and the method that produce the lowest forecasting error are obtained in Equation (1.5).

$$\begin{aligned} \epsilon^* &= min(E) \\ m^* &= argmin(E) \end{aligned} \tag{1.5}$$

1.4 Objectives

The objective proposed for this thesis is to present a forecasting method that can deal with noisy and chaotic time series, and compare it with the most used forecasting methods.

1.4.1 General Objective

Design and develop a computer software capable of generating competitive forecasts for chaotic time series in the presence of noise.

1.4.2 Specific Objectives

In order to fulfill the general objective it is necessary to:

- Determine the tools and metrics to analyze time series.
- Implement the algorithms that will compose the proposed forecasting method, optimizing their operations.
- Compare the forecasting method against the state of the art in time series forecasting.

1.5 Justification

We consider that time series forecasting can be improved by proposing alternative methods, such as the ones presented in this thesis. Also, we consider that phase space reconstruction methods can be enhanced with the inclusion of Differential Evolution as parameter optimization, instead of the deterministic approach that has not been modified in the almost quarter of century since its proposal.

Also, as mentioned in Section 1.1, time series are present in many aspects of every day. The correct forecast of these processes and phenomena is critical for some industries such as the generation of electrical energy from wind generators or by photo-voltaic panels, and at least desirable in the rest of the cases. However, many of these time series observe noise and chaos and traditional methods are not able to deal with this kind of time series.

1.6 Chapter Description

This thesis is composed of eight chapters. Chapter 2 describes the algorithms that comprises the theoretical framework of this work. Chapter 3 discusses the technical implementation of the theoretical framework. Chapter 4 covers a case study using synthetic time series. Chapter 5 describes an application of the system developed in wind speed forecasting. Chapter 6 shows the use of the method to forecast solar radiance and temperature. Chapter 7 presents a case study of electrical demand forecasting. Finally Chapter 8 contains a discussion of the results obtained by this method and our thoughts about this method and future improvements.

Chapter 2

Time Series Analysis

This chapter presents a set of basic time series concepts that will help to describe the data to further analyze it. It also presents the algorithms that integrates the forecasting system developed in this doctoral program. This chapter contains parts published in [24].

2.1 Background

Time series forecasting and analysis is a long and vastly studied aspect of science and engineering, where at least four methodologies arise as the most used and with the best results: linear and non-linear regression, exponential smoothing methods, auto-regressive models, and machine learning methods.

In this section we will discuss the three latter, since linear and non-linear regression simply assume that by adjusting the coefficients of the terms that compose a regression curve it will recreate the structure of a time series. However, this approach does not consider noise nor chaos, which are a fundamental part of this thesis.

2.1.1 Exponential Smoothing Methods

Exponential Smoothing is a technique that smooths time series data by using the exponential window function (smoothing refers to the removal of noise or undesirable behavior from a signal). While the moving average window function weighs the samples inside of the window equally, exponential smoothing weights the samples exponentially decreasing in time.

An approach to obtain a smoother signal that will react to process changes faster assigns geometrically decreasing weights to the previous observations. Hence an exponentially weighted smoother is obtained by introducing a discount factor Θ as

$$\sum_{t=0}^{T-1} \Theta^t y_{T-t} = y_T + \Theta y_{T-1} + \Theta^2 y_{T-2} + \ldots + \Theta^{T-1} y_1 \tag{2.1}$$

Note that if the previous observations are to be considered in a geometrically decreasing manner, then we must have $\mid \Theta \mid < 1$. However, the smoothing coefficient in Equation (2.1) is not an average since the sum of the weights is

$$\sum_{t=0}^{T-1} \Theta^t = \frac{1-\Theta^T}{1-\Theta} \tag{2.2}$$

and hence (2.2) does not necessarily add up to 1. For that we can adjust the smoother in Equation (2.1) by multiplying it by $^{(1-\Theta)}/_{(1-\Theta^T)}$. However, for large T values, Θ^T goes to zero and so the exponentially weighted average will have the following form:

$$\tilde{y}_T = (1-\Theta)\sum_{t=0}^{T-1} \Theta^t y_{T-t} = (1-\Theta)(y_T + \Theta y_{T-1} + \Theta^2 y_{T-2} + \ldots + \Theta^{T-1} y_1) \tag{2.3}$$

This is called a simple or first-order exponential smoother [55].

There exist many exponential smoothing approaches, being the most popular the Holt-Winters Exponential Smoothing technique. Holt-Winters incorporates periodic seasonality into the model. For this we decompose the smoothed values $\tilde{y}_t$ into the sum of a trend component ℓ_t and a seasonal component s_t,

$$\tilde{y}_t = \ell_t + s_t \tag{2.4}$$

The trend – consisting of the local level ℓ_t and the local slope b_t – and the seasonal component s_t are smoothed separately by the recursions

$$\begin{aligned} \ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta) b_{t-1} \\ s_t &= \gamma(y_t - \ell_t) + (1-\gamma) s_{t-m}, \end{aligned} \tag{2.5}$$

where m is the seasonality period. Note that ℓ_t is calculated from the observations corrected from seasonality. h-step ahead forecasts are obtained by means of

$$\hat{y}_{t+h|t} = \ell_t + hb_t + s_{t-m+i}, \qquad h = jm+i, i = 1, 2, \ldots, m, \quad j = 0, 1, \ldots \quad (2.6)$$

The parameters of all these exponential smoothing techniques depend on the smoothing parameters α, β, and γ, which are sometimes set arbitrarily to values between 0.05 and 0.3. Alternatively, they can be chosen by minimizing the sum of the squared 1-step ahead prediction errors $\sum_{t=k+1}^{n} (y_t - \hat{y}_{t-1})^2$ or another error criterion [25].

2.1.2 Auto-regressive Models

While exponential smoothing methods incorporate a very meaningful distinction of separating noise from the actual behavior of a time series, the consideration that noise comes from an isolated process at each sample is not suitable in some scenarios.

Auto-regressive models make the assumption that successive observations in a time series show serial dependence between the signal and noise. Under these circumstances, forecasting methods based on exponential smoothing may be inefficient and sometimes inappropriate because they do not take advantage of the serial dependence in the observations in the most effective way.

Given a time series of data y_t, an ARMA(p', q) model is given by

$$y_t - \alpha_1 y_{t-1} - \ldots - \alpha_{p'} y_{t-p'} = \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_q \epsilon_{t-q} \quad (2.7)$$

or equivalently by

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) y_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \epsilon_t \quad (2.8)$$

where L is the lag operator, the α_i are the parameters of the auto-regressive part of the model, the θ_i are the parameters of the moving average part and the ϵ_t are error terms. The error terms are generally assumed to be independent, identically distributed variables sampled from a normal distribution with zero mean.

Assume now that the polynomial $\left(1 - \sum_{i=0}^{p'} \alpha_i L^i\right)$ has a unit root[1] (a factor $(1 - L)$) of multiplicity d. Then it can be rewritten as:

$$\left(1 - \sum_{i=1}^{p'} \alpha_i L^i\right) = \left(1 - \sum_{i=1}^{p'-d} \phi_i L^i\right)(1 - L)^d \tag{2.9}$$

An ARIMA(p, d, q) process expresses this polynomial factorization property with $p = p' - d$ and is given by:

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right)(1 - L)^d y_t = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\epsilon_t \tag{2.10}$$

and thus can be thought as a particular case of an ARMA$(p + d, q)$ process having the auto-regressive polynomial with d unit roots. The above can be generalized as follows

$$\left(1 - \sum_{i=1}^{p} \phi_i L^i\right)(1 - L)^d y_t = \delta + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\epsilon_t \tag{2.11}$$

This defines an ARIMA(p, d, q) process [4].

ARIMA and its variants have been vastly used to forecast diverse time series, from electrical demand [13; 14], wind speed [43; 7], to house pricing [15] and many others with fairly good results. ARIMA is based on statistical techniques, and is often used when certain properties have been identified in a time series.

2.1.3 Machine Learning Methods

Machine learning is an interdisciplinary field that uses statistical techniques to give computer systems the ability to learn (i.e., progressively improve performance on a specific task) from data, without being explicitly programmed. Overall, machine learning refers to methods that capture more general patterns, while statistical methods are tailored to detect particular patterns.

This thesis we will focus on two machine learning methods: Artificial Neural Networks (ANN) and Nearest Neighbors (NN).

[1] A unit root is a stochastic trend in a time series, sometimes called a "random walk with a drift".

Artificial Neural Networks

ANN's are computing systems vaguely inspired by the biological neural networks that constitute animal brains [87]. Figure 2.1 shows the basic structure of an artificial neuron.

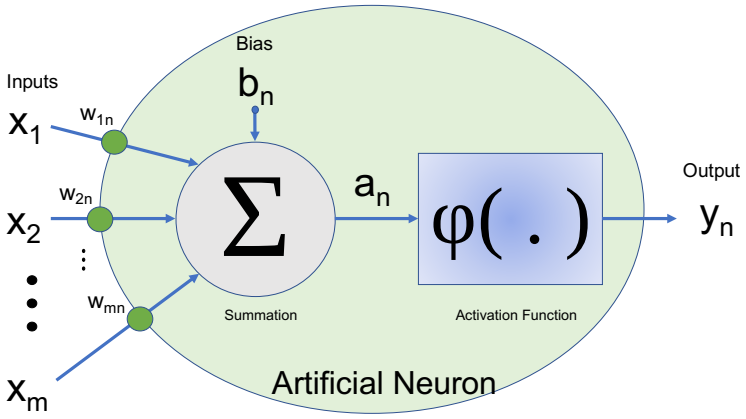


Figure 2.1: Artificial Neuron

A neuron n can receive multiple inputs $(x_1, x_2, \ldots, x_m)$, which are weighed individually $(w_{1n}, w_{2n}, \ldots, w_{mn})$ to make an intermediate output a_n, this is

$$a_n = \sum_{i=1}^{m} w_i x_i + b_n \tag{2.12}$$

where b_n is a bias that acts like a weight. IN this linear expression, the bias is an independent term that allows the neuron to shift its behavior and have a better fit with functions displaced from the origin.

This output a_n is passed to a $\varphi(\cdot)$, the activation function, which yields the neuron output y_n

$$y_n = \varphi(a_n) \tag{2.13}$$

By piling many neurons operating in parallel it is possible to form a layer. ANN's are usually composed of one input layer of length m, one output layer (of a single or many outputs) and a varying number of hidden layers which in turn can have a varying number of neurons [30]. Figure 2.2 shows a typical ANN topology.

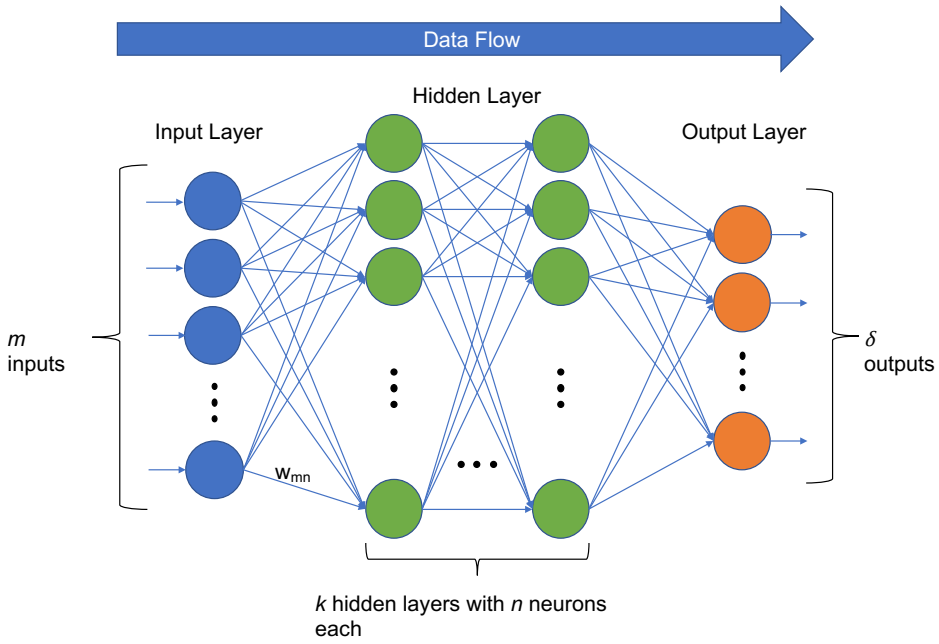


Figure 2.2: Artificial Neural Network

Generally speaking, ANN's are universal approximators. Theorem 1 states the Universal Approximation Theorem.

Theorem 1 *Let $\varphi(\cdot)$ be a non-constant, bounded, and monotone-increasing continuous function. Let I_{m_0} denote the m_0-dimensional unit hypercube $[0,1]^{m_0}$. The space of continuous functions on I_{m_0} is denoted by $C(I_{m_0})$. Then, given any function $f \ni C(I_{m_0})$ and $\epsilon > 0$, there exists an integer m_1 and sets of real constants α_i, b_i, and w_{ij} where $i = 1, \ldots, m_1$ and $j = 1, \ldots, m_0$ such that we may define*

$$F(x_1, \ldots, x_{m_0}) = \sum_{i=0}^{m_1} \alpha_i \varphi \left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right) \tag{2.14}$$

as an approximate realization of the function $f(\cdot)$; that is,

$$| F(x_1, \ldots, x_{m_0}) - f(x_1, \ldots, x_{m_0}) | < \epsilon$$

for all $x_1, x_2, \ldots, x_{m_0}$ that lie in the input space.

The proof of this theorem can be found in [31].

In time series forecasting, an ANN can identify patterns and tendencies that are contained in the data and adjust the weights of each neuron to match the desired output. ANN's have been used successfully in time series forecasting, in works such as forecasting water demand [66], foreign exchange market [92], market volatility [44], and more.

2.1.4 Nearest Neighbors

Nearest Neighbors is one of the oldest and simplest machine learning algorithms. It is based on the idea that similar sequences (measured by the distance between the evaluation sequence and the rest of the set) or instances will generate similar results. The algorithm will be fully described in Chapter 2. While the use of nearest neighbors in classification problems is ample, in time series forecasting the interest over it declined in recent years. [22] present a forecaster for chaotic time series based on NN. In [41] Nearest Neighbors is described as a simple non-linear time series forecaster used to forecast diverse problems that present chaotic structure.

2.2 Statistics

To describe a time series it is necessary to extract features and apply statistical techniques to it. This section describes the most used analysis techniques in time series forecasting.

2.2.1 Time Series Plotting

The first and most important tool in time series analysis is plotting. Time series plots can reveal patterns such as random, trends, level shifts, periods or cycles, unusual observations, or a combination of patterns [55].

Usually, time series are plotted with time as the x-axis and the measured variable as the y-axis. Time is considered to be discrete, it should reflect the sampling frequency used, and must be equidistant (this is, there should not be gaps between measures and all the samples should be taken at the same sampling time – not earlier and not before time).

2.2.2 Descriptive Statistics

A time series can be seen as a set of samples, much like a statistical variable extracted from a population. Based on this concept, a time series posses the main statistical moments: mean and variance. Also, the samples contained in a time series can exhibit certain frequency in a range of values that can be modeled to fit a probabilistic distribution such as the normal distribution.

2.2.3 Stationary Time Series

A very important type of time series is a stationary time series. A time series is said to be strictly stationary if its properties are not affected by a change in the time origin. That is, if the joint probability distribution of the observations $y_t, y_{t+1}, \ldots, y_{t+n}$ is exactly the same as the joint probability distribution of the observations $y_{t+k}, y_{t+k+1}, \ldots, y_{t+k+n}$ then the time series is strictly stationary. When $n = 0$ the stationarity assumption means that the probability distribution of y_t is the same for all time periods and can be written as $f(y)$ [55].

Stationarity implies a type of statistical equilibrium or stability in the data. Consequently, the time series has a constant mean defined in the usual way as

$$\mu_y = E(y) = \int_{-\infty}^{\infty} y f(y) dy \tag{2.15}$$

and constant variance, defined as

$$\sigma_y^2 = Var(y) = \int_{-\infty}^{\infty} (y - \mu_y)^2 f(y) dy \tag{2.16}$$

The sample mean and sample variance are used to estimate these parameters. If the observations in the time series are $y_1, y_2, \ldots, y_t$ then the sample mean is

$$\bar{y} = \hat{\mu}_y = \frac{1}{T} \sum_{t=1}^{T} y_t \tag{2.17}$$

and the sample variance is

$$s^2 = \hat{\sigma}_y^2 = \frac{1}{T} \sum_{t=1}^{T} (y_t - \bar{y})^2 \tag{2.18}$$

In order to find if a time series is stationary, we have at our disposal several tools, such as the histogram of the time series, the auto-correlation plot, and the Dickey-Fuller test. This section describes the auto-correlation function and the Dickey-Fuller test.

Auto-covariance and Auto-correlation Functions

A time series is stationary if the joint probability distribution of any two observations is the same for any two time periods t and $t+k$ that are separated by the same interval k. Useful information about this joint distribution, and hence about the nature of the time series, can be obtained by plotting a scatter diagram of all of the data pairs y_t, y_{t+k} that are separated by the same interval k. The interval k is called the lag [55].

The covariance between y_t and its value at another time period, say y_{t+k} is called the auto-covariance at lag k, defined by

$$\gamma_k = Cov(y_t, y_{t+k}) = E[(y_t - \mu)(y_{t+k} - \mu)] \tag{2.19}$$

where E is the expected value and $Cov(\cdot)$ is the covariance function.

The collection of the values of $\gamma_k, k = 0, 1, 2, \ldots$ is called the auto-covariance function. Note that the auto-covariance at lag $k = 0$ is just the variance of the time series; that is, $\gamma_0 = \sigma_y^2$. The auto-correlation coefficient at lag k is

$$\rho_k = \frac{E[(y_t - \mu)(y_{t+k} - \mu)]}{\sqrt{E[(y_t - \mu)^2]E[(y_{t+k} - \mu)^2]}} = \frac{Cov(y_t, y_{t+k})}{Var(y_t)} = \frac{\gamma k}{\gamma_0} \tag{2.20}$$

The collection of the values of $\rho_k, k = 0, 1, 2, \ldots$ is called the auto-correlation function (ACF). Note that by definition $\rho_0 = 1$. Also, the ACF is independent of the scale of measurement of the time series, so it is a dimensionless quantity. Furthermore, $\rho_k = \rho{-}k$; that is, the auto-correlation function is symmetric around zero, so it is only necessary to compute the positive (or negative) half.

To estimate the auto-covariance and auto-correlation functions from a time series of finite length, the usual estimate of the auto-covariance function is

$$c_k = \hat{\gamma}_k = \frac{1}{T}\sum_{t=1}^{T-k}(y_t - \bar{y})(y_{t+k} - \bar{y}), k = 0, 1, 2, \ldots, K \tag{2.21}$$

and the auto-correlation function is estimated by the sample auto-correlation function (or sample ACF)

$$r_k = \hat{\rho}_k = \frac{c_k}{c_0}, k = 0, 1, 2, \ldots, K \tag{2.22}$$

[55] recommend at least 50 observations to calculate a reliable estimate of the ACF and the lag K be about $T/4$ (where T is the length of the time series). However, it has been shown that K should be around 2 times the suspected seasonality lag of the time series, and length T as large as possible. That information allows to observe the points where the auto-correlation function contains a local maximum in order to infer the seasonality of a time series.

Usually, a confidence interval is calculated along the ACF plot. This confidence interval is calculated by Equation (2.23).

$$\pm z_{1-\alpha/2}\sqrt{\frac{1}{N}(1 + 2\sum_{i=1}^{k} y_i^2)} \tag{2.23}$$

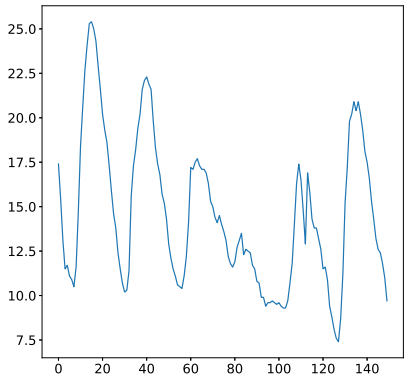
where k is the lag, N is the sample size, z is the cumulative distribution function of the standard normal distribution and α is the significance level. In this case, the confidence bands increase as the lag increases.

The ACF plot is a tool to detect randomness in a time series. This randomness is ascertained by computing autocorrelations for data values at varying time lags. If random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero.

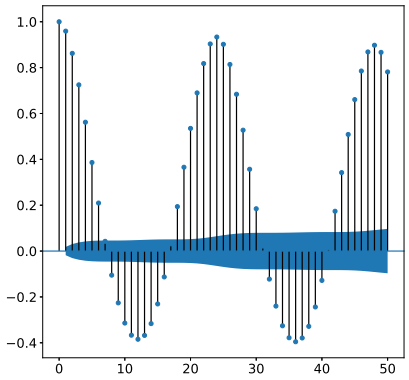
Figure 2.3a shows the hourly ambient temperature measurement in the city of Morelia Michoacán and 2.3b its ACF plot. Usually, an ACF is plotted as vertical lines instead of a continuous line, to clearly indicate the value of the auto-correlation at each time lag. Included in the plot is the confidence interval (the blue translucent area that surrounds the x axis), which traditionally is produced by a 95% confidence interval.

From the example we can infer that the time series is not produced by a random process, since each auto-correlation value is non zero and exceeds the confidence interval. Also, the time series presents a strong seasonal component every 24th lag.

However, having an auto-correlation value exceeding the confidence interval sometimes can indicate something else. Figure 2.4a shows a section of a synthetic and chaotic time series and Figure 2.4b its ACF.

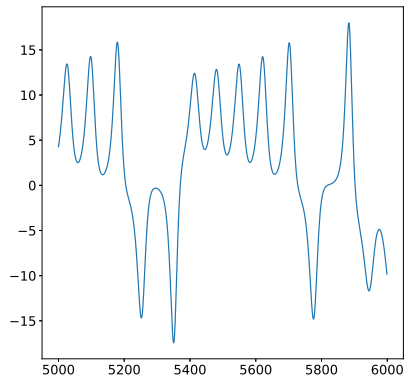


(a) Hourly Temperature Measures

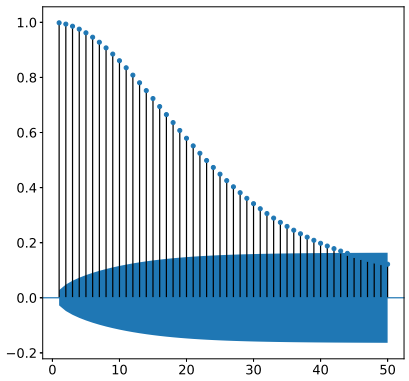


(b) ACF Plot of Hourly Temperature Measures

Figure 2.3: Temperature Measures and its ACF Plot



(a) Synthetic Time Series



(b) ACF Plot of Synthetic Time Series

Figure 2.4: Synthetic Time Series and its ACF Plot

From the figure we see that the auto-correlation values are outside the 95% confidence interval. Also, it has a somewhat monotonically descending behavior. This case demonstrate an scenario where the auto-correlation function does not serve well to detect stationarity. In order to determine stationarity is necessary to use more robust tests, such as the Dickey-Fuller test. However, it is important to note that the auto-correlation function is useful to detect seasonality.

Dickey-Fuller Test

In statistics, the Dickey–Fuller test tests the null hypothesis that a unit root is present in an autoregressive model. The alternative hypothesis is different depending on which version of the test is used, but it is usually stationarity or trend-stationarity. It is named after the statisticians David Dickey and Wayne Fuller, who developed the test in 1979 [20].

A simple AR(1) model is shown in Equation (2.24)

$$y_t = \rho y_{t-1} + u_t \tag{2.24}$$

where y_t is the variable of interest, t is the time index, ρ is a coefficient, and u_t is the error term. A unit root is present if $\rho = 1$. The model would be non-stationary in this case.

The regression model can be written as (Equation (2.25))

$$\Delta y_t = (\rho - 1) y_{t-1} + u_t = \delta y_{t-1} + u_t \tag{2.25}$$

where Δ is the first difference operator. This model can be estimated and testing for a unit root is equivalent to testing $\delta = 0$ ($\delta \equiv \rho - 1$). Since the test is done over the residual term rather than raw data, it is not possible to use standard t-distribution to provide critical values. Therefore, this statistic t has a specific distribution simply known as the Dickey–Fuller table.

There are three main versions of the test:

- Test for a unit root (Equation (2.26)):

$$\Delta y_t = \delta y_{t-1} + u_t \tag{2.26}$$

- Test for a unit root with drift (Equation (2.27)):

$$\Delta y_t = a_0 + \delta y_{t-1} + u_t \tag{2.27}$$

- Test for a unit root with drift and deterministic time trend:

$$\Delta y_t = a_0 + a_1 t + \delta y_{t-1} + u_t \tag{2.28}$$

Each version of the test has its own critical value which depends on the size of the sample. In each case, the null hypothesis is that there is a unit root, $\delta = 0$. The tests have low statistical power in the sense that they often cannot distinguish between true unit-root processes ($\delta = 0$) and near unit-root processes (δ is close to zero). This is called the "near observation equivalence" problem.

The intuition behind the test is as follows. If the series y is stationary (or trend stationary), then it has a tendency to return to a constant (or deterministically trending) mean. Therefore, large values will tend to be followed by smaller values (negative changes), and small values by larger values (positive changes). Accordingly, the level of the series will be a significant predictor of next period's change, and will have a negative coefficient. If, on the other hand, the series is integrated, then positive changes and negative changes will occur with probabilities that do not depend on the current level of the series; in a random walk, the current position does not affect which way you will go next [20].

In this thesis is used the Augmented Dickey-Fuller (ADF) test, which is more robust and it is implemented in the statsmodels library[2].

2.3 Maximum Lyapunov Exponent of a Time Series

Time series can also be measured qualitatively, and one of those qualities is the level of chaos present in the data. Chaos is considered to be the aperiodic, long-term behavior of a bounded, deterministic system that exhibits sensitive dependence to initial conditions [81]. The assumption of determinism is difficult to prove in many real world time series, however, it is more difficult to establish sensitive dependence on initial conditions. For that purpose we must quantify this sensitivity.

The Lyapunov exponent (or more specifically, the spectrum of Lyapunov exponents) of a dynamical system is a quantity that characterizes the rate

[2]https://www.statsmodels.org/stable/index.html

of separation of infinitesimally close trajectories. Quantitatively, two trajectories in phase space with initial separation $\delta\mathbf{Z}_0$ diverge (provided that the divergence can be treated within the linearized approximation) at a rate given by Equation (2.29):

$$|\delta\mathbf{Z}(t)| \approx e^{\lambda t}|\delta\mathbf{Z}_0| \tag{2.29}$$

where λ is the Lyapunov exponent [2].

A dynamic system has as many Lyapunov exponents as its order. The spectrum of Lyapunov exponents is denoted with a subindex $\lambda_i(i = 1, 2, \ldots, n)$, where n is the number of differential equations or number of state variables in the system [69].

The largest Lyapunov exponent can be defined using Equation (2.30)

$$d(t) = Ce^{\lambda_1 t} \tag{2.30}$$

where $d(t)$ is the average divergence at time t and C is a constant that normalizes the initial separation.

Since Lyapunov exponents must be calculated numerically (since the system that generates the data is unknown) and the largest exponent is the most important, it is useful to develop a general numerical technique that works for any system in any dimension and that does not require explicit evaluation of the monodromy matrix[3]. You can apply this method to any system without having to calculate the matrix of partial derivatives. In essence, you numerically evaluate the derivative along the direction of maximum expansion (or minimum contraction) and average its logarithm over the orbit [81]. There exist many algorithms to numerically calculate the Lyapunov exponents, but the one used on this thesis is described in [69], known as Rosenstein's algorithm.

The algorithm goes as follows. The first step consists in reconstructing the attractor dynamics from a single time series. By using the method of delays[4], the reconstructed trajectory, $\boldsymbol{X}$, can be expressed as a matrix where

[3]A monodromy matrix is the fundamental matrix of a system of ordinary differential equations (ODE's) evaluated at the period of the coefficients of the system. It is used for the analysis of periodic solutions of ODE's in Floquet theory.

[4]the basic idea of this method is that the state of an m-dimensional dynamical system can be uniquely characterized by m independent quantities. One such set of independent quantities are, of course, the phase space coordinates (more exact: the coordinates in some m-dimensional basis which spans M) [60]

each row is a phase-space vector. That is,

$$\boldsymbol{X} = (X_1, X_2, \ldots, X_M)^T \tag{2.31}$$

where X_i is the state of the system at discrete time i. For a time series $\{x_1, x_2, \ldots, x_N\}$ of length N, each X_i is given by

$$X_i = (x_i, x_{i+J}, \ldots, x_{i+(m-1)J}) \tag{2.32}$$

where J is the lag or reconstruction delay and m is the embedding dimension. Thus $\boldsymbol{X}$ is an $M \times m$ matrix, and the constants m, M, J, and N are related as by Equation (2.33).

$$M = N - (m-1)J \tag{2.33}$$

The embedding dimension is usually estimated according to Takens' theorem, i.e. $m > 2n$, although this algorithm often works well when m is below the Takens criterion [69]. Determining the proper lag is done in this thesis with the Mutual Information method. At the time Rosenstein's algorithm was published and still nowadays the determination of the embedding dimension remains an open problem.

After reconstructing the dynamics, the algorithm locates the nearest neighbor of each point on the trajectory. The nearest neighbor $\boldsymbol{X}_{\hat{j}}$, is found by searching for the point that minimizes the distance to the particular reference point, $\boldsymbol{X}_j$. This is expressed as

$$d_j(0) = \min_{\boldsymbol{X}_{\hat{j}}} \|\boldsymbol{X}_j - \boldsymbol{X}_{\hat{j}}\| \tag{2.34}$$

where $d_j(0)$ is the initial distance from the j-th point to its nearest neighbor, and $\|\ \|$ denotes the Euclidean norm. An additional constrain to determining $\hat{j}$ is that nearest neighbors have a temporal separation greater than the mean period[5] of the time series

$$\|j - \hat{j}\| > \text{mean period} \tag{2.35}$$

this allows the algorithm to consider each pair of neighbors as nearby initial conditions for different trajectories. The largest Lyapunov exponent is then estimated as the mean rate of separation of the nearest neighbors [69].

[5]The mean period is estimated as the reciprocal of the mean frequency of the power spectrum.

A first approach to estimate λ_1 is

$$\lambda_1(i) = \frac{1}{i\Delta t}\frac{1}{(M-i)}\sum_{j=1}^{M-i} ln\frac{d_j(i)}{d_j(0)} \tag{2.36}$$

where Δt is the sampling period of the time series, and $d_j(i)$ is the distance between the jth pair of nearest neighbors after i discrete-time steps. In order to improve convergence (with respect to i), the alternate form of Equation (2.36) is

$$\lambda_1(i,k) = \frac{1}{k\Delta t}\frac{1}{(M-k)}\sum_{j=1}^{M-k} ln\frac{d_j(i+k)}{d_j(i)} \tag{2.37}$$

In Equation (2.37), k is held constant and λ_1 is extracted by locating the plateau of $\lambda_1(i,k)$ with respect to i [69].

From the definition of λ_1 given in Equation (2.30), Rosenstein's algorithm assumes that the j-th pair of nearest neighbors diverge approximately at a rate given by the largest Lyapunov exponent (Equation (2.38)).

$$d_j(i) \approx C_j e^{\lambda_1(i\Delta t)} \tag{2.38}$$

where C_j is the initial separation. By taking the logarithm of both sides of Equation (2.38) we obtain

$$ln\ d_j(i) \approx ln\ C_j + \lambda_1(i\Delta t) \tag{2.39}$$

Equation (2.39) represents a set of approximately parallel lines (for $j = 1, 2, \ldots, M$), each with a slope roughly proportional to λ_1. The largest Lyapunov exponent is easily and accurately calculated using a least-squares fit to the "average" line defined by

$$y(i) = \frac{1}{\Delta t}\langle ln\ d_j(i)\rangle \tag{2.40}$$

where $\langle\ \rangle$ denotes the average over all points of j in the time series [69].

Figure 2.5 shows the curve of the log-divergence of the Lorenz attractor time series (blue dots) and the regression fit (red line) for an embedding dimension $m = 3$.

Is important to note that, the regression line is adjusted from the fifth to the thirteenth point. This is done since the divergence tends to increase

(which makes the first points unusable), but then stabilizes if the series is not chaotic. The slope of the curve is positive, which indicates chaotic behavior.

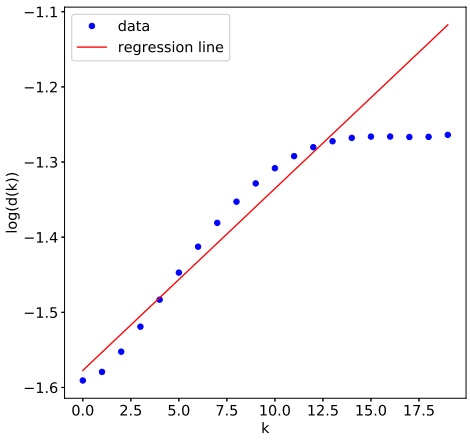


Figure 2.5: Maximum Lyapunov Exponent of Lorenz Attractor $m = 3$

Figure 2.6 shows the log-divergence plot of hourly temperature measures for an embedding dimension of $m = 2$ (in a forecasting setting, this m needs to be iterated to capture the possible chaotic behavior present in the time series).

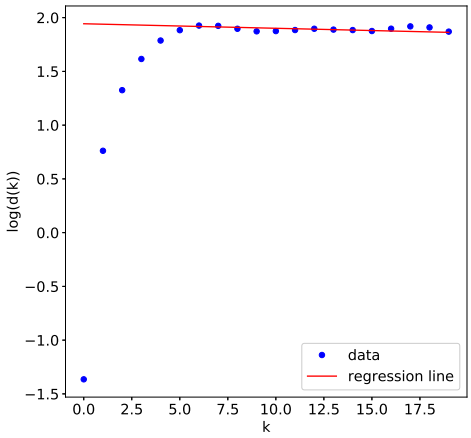


Figure 2.6: Maximum Lyapunov Exponent of Hourly Temperature $m = 2$

It is possible to see that the curve stabilizes rapidly, giving a slimly slight negative slope. In this cases it is almost certain that the time series can not be considered chaotic.

2.4 Forecast Evaluation

In order to validate the results of a forecasting technique it is necessary to use a metric (or set of metrics) that measures the error between the actual values of the time series and the forecasts. Error metrics tend to focus on different aspects of the quality of the forecasts. In this thesis we will use four error metrics: Mean Absolute Error (MAE) Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and Symmetric Mean Absolute Percentage Error (SMAPE).

2.4.1 Mean Absolute Error

The Mean Absolute Error Mean Absolute Error (MAE) is the average distance between each forecasted point on a scatter plot. MAE is also the average horizontal distance between each point and the identity line. It is always non-negative and values closer to zero are better.

MAE is given by Equation (2.41).

$$\text{MAE} = \frac{1}{n}\sum_{i=0}^{n} \mid y_i - \hat{y}_i \mid \tag{2.41}$$

where n is the length of the samples, y_i is the i-th actual value, and $\hat{y}_i$ is the i-th predicted value.

MAE has a clear interpretation as the average absolute difference between y_i and $\hat{y}_i$.

2.4.2 Mean Squared Error

The Mean Squared Error (MSE) of a forecaster measures the average squared difference between the estimated values and what is estimated. It is always non-negative, and values closer to zero are better.

The MSE is the second moment (about the origin) of the error, and thus incorporates both the variance of the forecaster (how widely spread the estimates are from the actual values w.r.t. the forecast) and its bias (how far off the average estimated value is from the truth). For an unbiased forecaster, the MSE is the variance of the estimator. Like the variance, MSE has the same units of measurement as the square of the quantity being estimated. In an analogy to standard deviation, taking the square root of MSE yields the root-mean-squared error or root-mean-squared deviation (RMSE or RMSD), which has the same units as the quantity being estimated; for an unbiased predictor, the RMSE is the square root of the variance, known as the standard error.

MSE is defined in Equation (2.42):

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2 \tag{2.42}$$

where n is the length of the samples, y_i is the ith actual value, and $\hat{y}_i$ is the ith predicted value.

2.4.3 Mean Absolute Percentage Error

The Mean Absolute Percentage Error (MAPE) of a forecaster measures the average absolute error as percentage between observed and forecast values. It is always non-negative and values closer to zero are better.

MAPE is defined in Equation (2.43):

$$\text{MAPE} = \frac{100}{n} \sum_{i=0}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{2.43}$$

where n is the length of the samples, y_i is the ith actual value, and $\hat{y}_i$ is the ith predicted value. MAPE allows to measure the effectiveness of a given forecaster between different time series, since it reports the error as percentage and is dimensionless. However, the main disadvantage of MAPE is that, if the data to predict contains zeros it will lead to indeterminate values.

2.4.4 Symmetric Mean Absolute Percentage Error

The Symmetric Mean Absolute Percentage Error (SMAPE) measures the the average absolute error weighing both the actual value and the forecast simultaneously as percentage with reference to the mean between the real and forecast values. It is always non-negative, it is bounded between 0 and 200% and values closer to zero are better.

SMAPE is defined in Equation (2.44)

$$\text{SMAPE} = \frac{100}{n} \sum_{i=0}^{n} \frac{\mid y_i - \hat{y}_i \mid}{(\mid y_i \mid + \mid \hat{y}_i \mid)/2} \tag{2.44}$$

where n is the length of the samples, y_i is the i-th actual value, and $\hat{y}_i$ is the i-th predicted value. SMAPE posses the same advantages of MAPE and minimizes (but does not eliminate entirely) the problem of having zeros in the data. In our implementation of SMAPE, if both y_i and $\hat{y}_i$ are equal to zero, the i-th forecast does not contribute to the error measure.

Both MAPE and SMAPE have a bias towards big values in the $\hat{y}$ set, lowering their contribution to the error. This makes MAPE symmetric and SMAPE asymmetric in the percentage scale, but the inverse happens in the unit scale (see [11]). However, both MAPE and SMAPE are useful to compare forecasts errors in a relative scale.

2.5 Nearest Neighbors Algorithm

The idea behind the Nearest Neighbors (NN) algorithm is simple and intuitive. Given the most recent observations of the time series, we search the time series for other observation sequences that are similar to the current one (this similarity is calculated by using a distance function). Then, we take the next value of each of them and obtain the mean of those values; this result is used as the forecast for the evaluated set of samples.

Let $\mathbb{S} = [s_1, s_2, \ldots, s_t, \ldots, s_N]$ be a time series, where s_t is the value of variable s at time t. It is desired to obtain the forecast of Δn consecutive values, this is $[s_{N+1}, s_{N+2}, \ldots, s_{N+\Delta n}]$ by employing any observation available in $\mathbb{S}$.

By using a time delay τ and an embedding dimension m, it is possible to build delay vectors of the form $S_t = [s_{t-(m-1)\tau}, s_{t-(m-2)\tau}, \ldots, s_{t-\tau}, s_t]$, where $m, \tau \in \mathbb{Z}^+$. These vectors represent a reconstruction of the m-dimensional phase space that defines the dynamics of the system that produced the time series (see [41]).

The time series is traversed with a sliding window S_N of size m over each delay vector. To obtain the neighbors closer to S_N an ϵ parameter is used and the distance to each delay vector S_t is calculated. The nearest neighbors are those S_t whose distance to S_N is at most ϵ.

$$\|S_N - S_t\| \leq \epsilon \qquad \forall t \in [m, N-1] \tag{2.45}$$

Each of these vectors S_t form the neighborhood $\upsilon_\epsilon(S_N)$ with radius ϵ around the vector S_N in an m-dimensional space. We can use any vector distance function to measure the distance between possible neighbors. In [17] the distance function used was the Absolute-value norm. However, [41] suggest that, when $m > 20$, the Euclidean distance function provides a better measurement of the neighbor for a given delay vector.

For each vector S_t that satisfies Equation (2.45) the individual values $s_{t+\Delta n}$ are retrieved. The forecast is the mean of the next values (i.e. $s_{t+\Delta n}$) of every delay vector in S_N's neighborhood, as expressed in Equation (2.46).

$$\hat{s}_{N+\Delta_n} = \frac{1}{\mid \upsilon_\epsilon(S_N) \mid} \sum_{S_t \in \upsilon_\epsilon(S_N)} \mathrm{nv}(S_t, \Delta n) \tag{2.46}$$

where $\mathrm{nv}(S_t, \Delta n) = s_{t+\Delta n}$. In the case $\Delta n = 1$ Equation (2.46) produces a forecast for the next instant of time. This forecasting scenario is known as

One Step Ahead (OSA) forecasting.

In the event of not finding any near neighbor, the value of ϵ is increased until we find at least one neighbor. The deterministic approach [41] suggests to start with $\epsilon = 1 \times 10^{-3}$ with updates of $\epsilon \leftarrow \epsilon \times 1.2$.

However, in many situations it is necessary to predict more than one value at a time. For this purpose we employ an N-Step Simultaneous (NSS) forecasting strategy [78]. The N-Step Simultaneous strategy is described in Equation (2.47).

$$\hat{s}_{N+r\Delta n} = \frac{1}{\mid \upsilon_\epsilon(S_N) \mid} \sum_{S_t \in \upsilon_\epsilon(S_n)} \mathrm{nv}(S_t, r\Delta n) \qquad r \in [1, \ldots, n_f] \tag{2.47}$$

where n_f is the required number of forecast steps.

In the simultaneous strategy, no forecasts are added to the forecast set; furthermore, no cumulative error is introduced through the inputs, since only the original data is used for the prediction of future values. It is quite important to control the error when we deal with chaotic systems since it is very difficult to distinguish it from actual data; otherwise, the error grows exponentially [41].

The precise implementation of Equation (2.47), and the rest of the algorithms that comprise this thesis will be shown in Chapter 3.

2.6 Differential Evolution

The NN forecasting method requires three parameters: m, τ, and ϵ (defined in the previous section). The values of those parameters affect dramatically the forecaster's performance, so it is necessary to determine their optimal values. In this work we propose the use of Differential Evolution to optimize those parameters.

Differential Evolution (DE) was presented in 1997 by Storn & Price [84; 63] as a meta-heuristic capable to optimize functions of arbitrary complexity. Similar to most evolutionary algorithms (EA), DE is a population-based optimizer. At each iteration DE tries to improve each individual's fitness, by slight alterations to them. After a number of iterations, the best individual found is returned as an approximation to the problem solution. The DE implementation used for this work is the traditional version, known as DE/rand/1/bin.

The parameters of the NN method are integers ($[m, \tau] \in \mathbb{Z}^+$), while ϵ is a real number ($\epsilon \in \mathbb{R}^+$). Since we use DE as an optimizer, we have two constrains over these parameters: type and limits.

The task of optimizing the parameters of NN is a mixed value problem, because it has both continuous and discrete values. This problem is solved by considering every variable as continuous for the evolutionary process, and round down the variables considered as discrete [45]. For this case, variables m and τ get rounded down in order to transform its continuous values to discrete. When working with discrete values, the objective function is evaluated once the continuous values have been calculated during DE. In this DE implementation, the individuals represent the NN parameters in the form $x^{(g,i)} = [m, \tau, \epsilon]$.

To determine the fitness of the individuals that constitute the population, it is necessary to compare the forecasts made by each individual against a subset of the time series unknown to the forecasters. This unknown subset is often called validation set and is usually comprised of the n last values of the time series (this set is denoted as $\mathbb{S}_{val}$). To make this comparison, any prediction accuracy measure function can be used.

Equation (2.48) shows the fitness function used to evaluate each DE individual.

$$\text{fitness}(x^{(g,i)}) = \text{error}(\mathbb{S}_{val}, \text{NN}(\mathbb{S}, \Delta n, x^{(g,i)})) \tag{2.48}$$

where $x^{(g,i)}$ is the i-th individual of generation g. The *error* function can be any function that measures the precision of the NN process when the genetic information of individual $x^{(g,i)}$ is used as parameters against the validation set $\mathbb{S}_{val}$. This function can be the Mean Absolute Percentage Error, Mean Squared Error, Symmetric Mean Absolute Percentage Error, etc. NN stands for the Nearest Neighbors forecasting algorithm solving one of the forecasting problems OSA or NSS.

To minimize the result of Equation (2.48), each individual $x^{g,i}$ is bound by the constraint $x_j^L \leq x_j^{(g,i)} \leq x_j^U$, where L is the vector that contains the lower limits of the j values that compose the individuals and U is the vector that contains the upper limits of the j values that compose the individuals. Formally, the problem to solve is given by Equation (2.6).

$$\begin{gathered} \min \text{fitness}(x^{(g,i)}) \\ \text{s.t.} \\ x_j^L \leq x_j^{(g,i)} \leq x_j^U \\ 1 \leq j \leq D \end{gathered} \tag{2.49}$$

where D is the dimension of the search space.

However, to satisfy the constraints of the NN parameters, there is a consideration we need to have over the mutation function of DE. In DE, every individual is crossed over with a mutant individual randomly generated. The mutation process is calculated by using Equation (2.50).

$$\begin{gathered} v^{(g,i)} = x^{(g,r_0)} + F(x^{(g,r_1)} - x^{(g,r_2)}) \\ \forall i \in [1, N_{pop}] \end{gathered} \tag{2.50}$$

where r_0, r_1 y r_2 are the indexes of randomly selected individuals, F is a positive real number that controls the speed in which the population evolves, and N_{pop} is the population size.

Because the current population satisfies every limit restriction, only the contributions of mutant vector can violate the limits of the parameters. Consequentially, the limits must be checked only when a mutant parameter is selected as test individual.

A reset method known as bounce-back [63], replaces the individual that has exceeded one or more of its limits with a valid individual that satisfies every constraint. The bounce-back strategy considers the progress towards the optimum, by selecting a value of the parameter that is between the base value of the parameter and the limit that is being violated.

When a mutant individual violates the parameter limits, Equations (2.51) and (2.52) replace this parameters in order to satisfy both lower and upper limits.

$$v_j^{(g,i)} = x_j^{(g,r_0)} + w(x_j^L - x_j^{(g,r_0)}) \tag{2.51}$$

$$v_j^{(g,i)} = x_j^{(g,r_0)} + w(x_j^U - x_j^{(g,r_0)}) \tag{2.52}$$

where $w = U(0, 1)$.

The NN algorithm requires fine tuning of its parameters in order to generate accurate forecasts. By adding DE as the parameter selection process, it makes it possible to obtain the best parameters given a specific fitness function. The resulting method is called Nearest Neighbors by Differential Evolution (NNDE).

What NNDE does is, for a randomly generated population of $[m, \tau, \epsilon]$ individuals, a forecast for a given time series is obtained per each individual. Then, each forecast is compared to the actual values of the time series by using an error measure such as MAPE or MSE. The resulting value is the fitness of each individual, and the individual with the best fitness is the one used to evolve the population. Once the evolutionary process is completed, the individual with the best fitness is retrieved and becomes the set of parameters to use to forecast the time series being evaluated.

2.7 Chapter Conclusions

This chapter described the foundational theoretical concepts in which this thesis is based, outlining the tools, metrics and methods that will serve to analyze, forecast, and asses the results that will be shown in the later chapters.

Chapter 3

Design and Implementation

This chapter describes the implementation developed for this thesis, the Nearest Neighbors algorithm, and the required functions to produce a computing system that optimizes the NN parameters for the given forecasting task. The system was designed as a library, where the invocation of the tasks is made by instantiating an object that contains both the data and the methods to process a time series.

As discussed in the previous chapter, the system is composed of two parts: the Nearest Neighbors time series forecaster, and the Differential Evolution parameters optimizer. The way they interact is, DE produces parameters that are introduced into the Nearest Neighbors component, where NN produces a forecast and an error value that DE will use to find the optimum set of parameters. Also, since DE is stochastic, this interaction needs to be repeated independently at least 30 times to assert that the best combination of parameters was found.

In other words, NN is a black box function that ultimately produces a value that is desired to minimize and DE will find a minimum in the given search space. However, since this minimum is not guaranteed to be the lowest minimum in the search space, it is necessary to repeat the process with a different population in order to find the lowest of minimums.

3.1 Description of the Optimization Process

The system was designed modularly to easily add and remove components, since it is designed to deal with many types of forecasting tasks.

The main focus of the library is to make it possible to run the forecasting task inside a Differential Evolution process and in a parallel.

Figure 3.1 shows a simple representation of the optimization process. Basically it is a wrapper for the DE process that will invoke a problem function (in this case a composition of the forecasting method, the time series, and the parameters of the NN algorithm) and returns a forecasting error along with the parameters that produced that error.

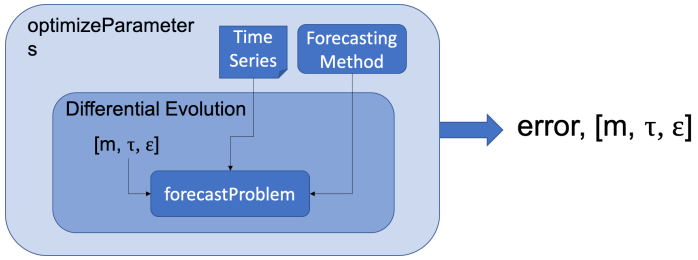


Figure 3.1: Diagram of the optimizeParameters Function

This process needs to be repeated since DE is of stochastic nature. Because making this process sequentially would take several hours to finish, it is convenient to run it in parallel, with each process running independently from each other.

Figure 3.2 shows a representation of the run process. This process will launch multiple optimize processes with the same parameters excepting a seeding parameter that will initialize the random number generator of the DE function.

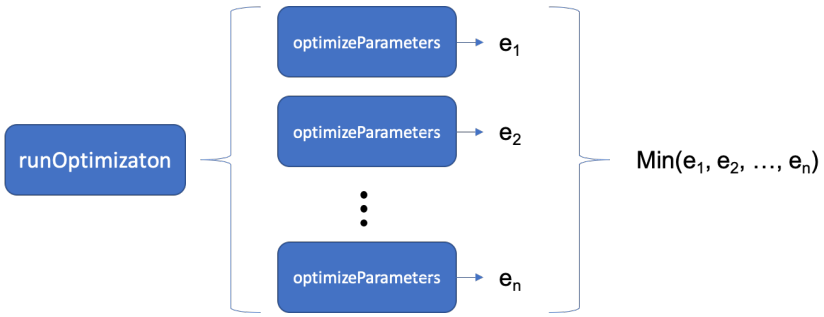


Figure 3.2: Diagram of the runOptimization Function

Once all instances of optimize are finalized, the minimum of all is selected as the best error and parameters.

3.2 Description of the Algorithms

The first step to make predictions is to convert the time series into a list of delay vectors. Algorithm 3.1 describes the operations required to make this list.

Algorithm 3.1 Form List of Delay Vectors

```
1: function formDVList(ts: array, m: int, tau: float, h: int,
       delta: int)
2:     dvi ← dvIndex(m, tau)
3:     now ← dvi[−1]
4:     fi ← formFutureIndex(now, h, delta)
5:     indDv ← [dvi + i for i in range(len(ts − fi[−1]))]
6:     indNv ← [fi + i for i in range(len(ts − fi[−1]))]
7:     DVList ← np.array([ts[r] for r in indDv])
8:     NVList ← np.array([ts[r] for r in indNv])
9:     return DVList, NVList
10: end function
```

The function receives *ts* which is an array that contains the values of the time series, m the length of the embedding dimension, *tau* the time delay between the delay vector dimensions, h the forecasting horizon, and *delta* the separation between the last dimension of the delay vector and the target values. First, an array of indexes is formed with the m and *tau* parameters. This array represents the sliding window that will traverse the time series (line 2). Then, the last value of the *dvi* array is assigned to the variable *now*(line 4), to then invoke the form future index function. This function will determine up to which index the list will comprise. Two matrices are formed, *indDv* and *indNv* (lines 5 and 6). This matrices store the indexes of each time step of the slide of the *dvi* window and the *fi* index, which will allow to extract the values from the time series (lines 7 and 8). The resulting delay vector list (*DVList*) and next values list (*NVList*) are returned by the function.

The delay vector index function (Algorithm 3.2) receives the length of the embedding dimension (m), and the time delay between the delay vector dimensions (*tau*). It creates and returns an array in the form $[\tau \times 0, \tau \times 1, \ldots, \tau \times (m-1), \tau \times m]$ which is used to traverse the time series and create the delay vectors.

Algorithm 3.2 Delay Vector Index

```
1: function DvIndex(m: integer, tau: integer)
2:     return np.array(list(range(0, m*tau, tau)))
3: end function
```

Algorithm 3.3 receives as parameters *now* which is the index of the last observation, the forecasting horizon *h*, the separation between the las dimension of the delay vector and the target values *delta* and it returns an array with the indexes of the target values of the first delay vector. The algorithm obtains the indexes of the target values for each delay vector given a forecasting horizon *h* and a *delta* which represents the number of future values to predict.

Algorithm 3.3 Form Future Index

```
1: function formFutureIndex(now: integer, h: integer,
       delta: integer)
2:     return np.array(range(now + h, now + h + delta*i, i))
3: end function
```

With the lists formed is possible to use any forecasting method based on phase space reconstruction for any kind of forecasting technique. In Algorithm 3.4 the Nearest Neighbors algorithm is used to produce one step ahead forecasts.

The algorithm receives the time series *ts*, the Nearest Neighbors parameters (*m*, *tau*, and *epsilon*), and the number of forecasts to make *ncasts*. The *delta* parameter is omitted since it is assumed to be 1. It separates the time series into a validation set and a training set (lines 2 and 4) and makes an empty list where the forecasts are going to be stored (line3). It is important to remember that the Nearest Neighbors algorithm does not require training, but to maintain the conventions of machine learning the variables are named in their usual form. The lists of delay vectors and next values are stored in *DVList* and *NVList*, respectively (line 5). Then, the forecasts are produced one by one by instantiating a NearestNeighbors object and passing the lists to it (lines 7 and 8). The last vector from the time series is obtained in line 9 and the prediction using this vector is appended to the forecast list (line 10). The new vector is appended to the list in line 11 and the actual value is obtained from the validation set and added to the next values list (line 12).

Algorithm 3.4 One Step Ahead Forecasting

```
1: function OSA(ts: array, m: integer, tau: integer, epsilon: float,
       ncasts: integer)
2:     valSet ← ts[−ncasts :]
3:     forecast ← LIST()
4:     tsTrain ← ts[: −ncasts]
5:     DVList, NVList ← FORMDVLIST(tsTrain, m, tau)
6:     for i in RANGE(ncasts, 0, −1) do
7:         nn ← NEARESTNEIGHBORS(epsilon)
8:         nn.FIT(X, y)
9:         newVector ← LASTVECTOR(ts[: −i], m, tau, dates)
10:        APPEND(forecast, nn.PREDICT(newVector))
11:        X ← npVSTACK(X, newVector)
12:        y ← np.VSTACK(y, [valSet[np.ABS(i − ncasts)])]
13:    end for
14:    forecast ← np.ARRAY(forecast)
15:    return ERRORFUNCTION(valSet, forecast)
16: end function
```

The result of this function is the error obtained by the Nearest Neighbors algorithm compared against the validation set (line 15). This error function can be any of the metrics exposed in Chapter 2.

The predictions are done by the predict function (Algorithm 3.5). The function receives *evalVector* which is the last delay vector in the time series and returns the predicted value. This function simply looks for all the neighbors of the vector being evaluated, obtains their indexes and calculates the forecast.

Algorithm 3.5 Predict

```
1: function PREDICT(evalVector: array)
2:     neighborhood ← OBTAINNEIGHBORSINDEXES(evalVector)
3:     return CALCULATEFORECAST(neighborhood, evalVector)
4: end function
```

The find neighbors function makes an exhaustive search for all of the delay vectors that are near the evaluation vector (Algorithm 3.6).

The function receives as parameters an array that contains the delay vec-

Algorithm 3.6 Find Neighbors

```
 1: function findNeighbors(DVList: array,
        evalVector: array, epsilon: float)
 2:     N ← len(DVList)
 3:     m ← len(evalVector)
 4:     dist ← 0.0
 5:     Sn ← list()
 6:     if m < 20 then
 7:         dist ← maxNorm(evalVector, DVList[i])
 8:     else
 9:         dist ← euclideanDistance(evalVector, DVList[i])
10:     end if
11:     if dist ≤ epsilon then
12:         append(Sn, i)
13:     end if
14:     return Sn
15: end function
```

tors extracted from the time series (*DVList*), the last delay vector (*evalVector*, and *epsilon* which is the radius of the neighborhood. For each delay vector, Algorithm 3.6 calculates the distance between it and the evaluation vector. If the length of the vectors is below 20 it used the max norm function (line 7) or the Euclidean distance (line). This condition exists since [41] says that, for delay vectors with less than 20 dimensions the max norm produces better results than the Euclidean distance. If this distance is lower or equal than the specified *epsilon*, the vector is considered a neighbor and its index is added to the *Sn* list (line 12). The function returns the list of indexes of all the neighbors of the evaluation vector (line 14).

Algorithm 3.7 receives as parameters *neighborhood* which is an array that contains the indexes of the neighbors of the current delay vector, the last delay vector *evalVector*, and the list with the target values of each delay vector extracted from the time series *NVList*. To calculate the forecast, Algorithm 3.7 looks in the next values list and retrieves the target values of all of the neighbors using their index. Depending if the length of the targets is greater than one, the function selects which function to call to calculate the mean of all of the targets. This is done since, for one case the list of next values is one-dimensional, while for the other case the list is two-dimensional.

Finally, it returns the mean value of the neighboring target values to the last delay vector.

Algorithm 3.7 Calculate Forecast

```
1: function CALCULATEFORECAST(neighborhood: array,
       evalVector: array, NVList: array)
2:     deltas ← [NVList[i] for i in neighborhood]
3:     if SHAPE(NVList, 1) > 1 then
4:         return CALCULATEMEANNSS(deltas, evalVector)
5:     else
6:         return CALCULATEMEAN(deltas, evalVector)
7:     end if
8: end function
```

Algorithm 3.8 obtains the forecast by determining the number of neighbors of the evaluation vector and by using the indicated measure of central tendency.

Algorithm 3.8 Calculate Mean

```
1: function CALCULATEMEAN(deltas: array, evalVector: array)
2:     f ← 0
3:     if LEN(deltas) > minNeighSize then
4:         f ← np.MEAN(deltas)
5:     else
6:         if useNaive then
7:             f ← evalVector[−1]
8:         else
9:             f ← ∞
10:        end if
11:    end if
12:    return f
13: end function
```

Algorithm 3.8 receives a *deltas* array that contains the target values, and *evalVector* which is the last delay vector. If the length of next values is greater than the *minNeighSize* variable, Algorithm 3.8 computes the mean of the *deltas* list (line 3). If the length of *deltas* is less than *minNeighSize*,

the function checks if the *useNaive* flag is true. If it is true, the forecast is the last known value of the time series (line 7). In this thesis, the *useNaive* flag is always true. The algorithm returns the forecast *f*.

To generate an N-step simultaneous forecast, the system uses Algorithm 3.9. This algorithm is quite similar to Algorithm 3.8, but instead of calculating the mean and median of singular vectors, it does it for many vectors. It receives the same parameters but *deltas* is a two-dimensional array. It returns an array of forecasts instead of a single value.

Algorithm 3.9 Calculate Mean NSS

```
1: function CALCULATEMEANNSS(dt: array, evalVector: array)
2:     dt ← np.ARRAY(dt)
3:     f ← None
4:     if LEN(dt) > minNeighSize then
5:         f ← [np.MEAN(dt[:, i]) for i in RANGE(SHAPE(dt, 1)])
6:     else
7:         if useNaive then
8:             f ← [evalVector[−1]] * SHAPE(dt, 1)
9:         else
10:            f ← [∞] * SHAPE(dt, 1)
11:        end if
12:    end if
13:    return f
14: end function
```

These functions are capable to generate forecasts of a time series when the NN parameters are known or at least estimated. However, in order to obtain these parameters, it is necessary to implement a method to produce the NN parameters.

As mentioned in Chapter 2, the proposed system incorporates Differential Evolution as the optimization process to obtain the optimal parameters of NN for a particular forecasting task. DE individual is a vector of length 3, where each dimension corresponds to the m, τ, and ϵ parameters, respectively.

each independent *optimizeParameters* process is initialized with a different seed, in order to improve the search of the optimal parameters.

Algorithm 3.10 receives the time series *ts*, *nf* number of forecasts to make, d which is the separation between the last dimension of the delay vector and

the target values, a forecasting horizon h, a number of simultaneous executions nj, mi maximum number of DE iterations, and ps which is the DE population size. This procedure initializes a processes pool with the number of desired processes to run (line 2). Then, a list of tuples is created where each tuple is the set of arguments that will be passed to every pooled process. Notice that all of the parameters are the same except the seed (s variable) that will initialize the random number generator of the *optimizeParameters* process (line 3). Finally, the required number of *optimizeParameters* processes are put to run with the specified arguments.

Algorithm 3.10 RunOptimization

```
1: procedure RUNOPTIMIZATION(ts: array, nf: integer, me: string,
       d: integer, h: integer, nj: integer, mi: integer, ps: integer)
2:     p ← POOL(nj)
3:     params ← [(s, ts, nf, me, d, h, mi, ps) for s in RANGE(nj)]
4:     p.RUN(optimize, params)
5: end procedure
```

The *optimizeParameters* function is described in Algorithm 3.11. This function is a wrapper for the Differential Evolution function (in this implementation, the differential evolution function is part of the `scipy.optimize` library – [40]).

Algorithm 3.11 OptimizeParameters

```
1: function OPTIMIZEPARAMETERS(s: integer, ts: array, nf: integer,
       me: string, d: integer, h: integer, dt: array,
       mi: integer, ps: integer)
2:     bounds ← [(1, 100), (1, 50), (0, MAXEPSILON(ts, nf))]
3:     problemArgs ← (bounds, mi, ps, (ts, nf, ef, me, d, h), s)
4:     result ← DE(forecastProblem, problemArgs, 0.75, False, False)
5:     return result.x, result.fun
6: end function
```

In line 2 the *bounds* variable define the minimum and maximum values that the evolutionary population can have. The m and τ parameters are bounded from 1 to 100 and 1 to 50, respectively, while the ϵ parameter is bounded from 0 to the return value of the *maxEpsilon* function. *maxEpsilon*

returns the range of the time series. Then, the *differentialEvolution* function is invoked and its result is assigned to the *result* variable. *differentialEvolution* receives as parameters the *forecastProblem* function (which is described in Algorithm 3.12), the bounds, the maximum iterations for the DE process, the population size, the arguments for the objective function which are the time series *ts*, the number of forecasts *nf*, the error function *ef*, the forecasting method *me*, the separation *d*, the forecasting horizon h (m, τ and ϵ are not considered arguments for the function but an special argument called x), the recombination probability, a display partial results flag, and a polish flag which is used by this implementation of the DE algorithm to further explore the search space (line 4). Once the differential evolution process is terminated, the function returns the best individual and its score (line 5).

The *forecastProblem* function (Algorithm 3.12) is a selector that initializes a Forecast object, and, depending of the desired forecasting scheme, it will select it and produce the forecasting score of the parameters sent to this function.

Algorithm 3.12 receives the DE individual x and a *args* tuple that contains the additional parameters for the function. This parameters structure is used since the *DE* function requires it. The *args* tuple contains the time series, the number of forecasts, the separation between the last element of the delay vectors and its targets, the forecasting horizon, and the error function. This function returns the error of the forecast made with the given parameters.

3.3 Chapter Conclusions

The resulting computing system is capable of running multiple instances of DE that will look for the optimum parameters for a given time series and forecasting task.

It also resulted very advantageous to modularize most of the aspects of the system, since it allowed to change forecasters without rewriting code.

Algorithm 3.12 forecastProblem

```
1: function forecastProblem(x: array , args: tuple)
2:     (m, tau, eps) ← extractParameters(x)
3:     fm ← Forecast(args[0], m, tau, eps, args[1], args[2], args[6])
4:     result ← ∞
5:     if find(args[3], ''osa'') ≥ 0 then
6:         result ← fm.osa()
7:     else
8:         if find(args[3], ''oha'') ≥ 0 then
9:             result ← fm.oneHourAhead(args[4], args[5])
10:        else
11:            if find(args[3], ''nss'') ≥ 0 then
12:                result ← fm.nss()
13:            else
14:                if find(args[3], ''oda'') ≥ 0 then
15:                    result ← fm.oneDayAhead(args[4], args[5])
16:                end if
17:            end if
18:        end if
19:    end if
20:    return result
21: end function
```

Chapter 4

Case Study: Synthetic Time Series Forecasting

NNDE was tested with four synthetic chaotic time series: the Hénon Map (henon – [32]), the Lorenz system (lorenz – [48]), Mackey-Glass (mackey – [51]), and the Rössler attractor (rossler – [70]). Every synthetic time series comprises 50,000 samples.

To compare the forecasts made by NNDE, three other methods were used to predict the synthetic time series: ARIMA, Radial Basis Function (RBF) ANN [6], and Recurrent Neural Networks (RNN) in the form of Long Short-term Memory (LSTM) Artificial Neural Network [35].

Most of the content of this chapter was first published in [24].

4.1 Related Work

One of the most used methodologies is the Auto-regresive Integrated Moving Average (ARIMA), which is used to obtain further information of the data and to forecast future points in the time series [4]. ARIMA has been used in many scientific efforts; in a similar work to this thesis, which is the volatility of stock market, ARIMA comes as the forecasting technique that obtains the best results in a prediction horizon of 10 days [44]. However, we have found contradictory literature, since it has been tested that, for intra-day stock market operations, AR models perform poorly compared to non-linear models [53].

Another methodology that has shown good results in time series fore-

casting is Artificial Neural Networks (ANN). ANN's have demonstrated that are capable of adjust to high dimensionality chaotic processes, generating more accurate predictions than ARIMA and some bayesian methods [71]. Nonetheless, ANN's have several restrictions with their implementation as we are going to show, being the most problematic for this kind of effort their high use of computational resources when dealing with huge amounts of data.

Considering the downsides of these methods, we focused on a rather simple method that has been successful in different areas of pattern recognition. The Nearest Neighbors algorithm (NN) [41] can be employed in time series forecasting due to its simplicity and intuitiveness. NN searches for similar instances recovering them from a large dimensional feature space with possibly incomplete data [22]. The NN algorithm assumes that sub-sequences of the time series that emerged in the past are likely to have a resemblance to the future sub-sequences and can be used for generating NN-based forecasts.

One significant drawback of the NN forecasting method is its sensitivity to changes in the input parameters (i.e. the length of the radius of the neighborhood, the embedding dimension and the delay between measurements). If the input parameters are not appropriately selected, the accuracy of the forecasts can decrease (as shown in [58]).

One way to select the best possible input parameters for the NN method is using an Evolutionary Algorithm (EA) [61]. EA's are computational algorithms made up of a collection of randomized global search paradigms for finding the optimal solutions to a given problem. In particular, Differential Evolution (DE), is a population-based search strategy that has recently proven to be a valuable method for optimizing real valued multi-modal objective functions [63] [84]. DE is a parallel direct search method exhibiting good convergence properties and simplicity in implementations.

The combination of the Nearest Neighbor algorithm with Differential Evolution optimization has been addressed in [17]. However, in this thesis we introduce a new comparison against Artificial Neural Networks, some improvements over the algorithm and a test with noise added to chaotic time series.

4.2 Data Analysis

The time series tested in this chapter are known to be chaotic; data were produced from their respective dynamic systems by means of integration.

Since data do not come from a natural process, they are called synthetic time series. However, for completeness we did an analysis of their seasonal components (via their ACF plot), tested for stationarity using the Dickey-Fuller test, and obtained their maximum Lyapunov exponents numerically. This section analyzes the chaotic time series in alphabetical order.

4.2.1 Henon Map

Henon map is a dissipative map (a map is a discrete dynamical system and dissipative means it includes dampening dynamics – [33]). The system that produces the Henon Map is shown in Equation (4.1).

$$\begin{aligned} x_{n+1} &= 1 - \alpha x_n^2 + y_n \\ y_{n+1} &= \beta x_n \end{aligned} \tag{4.1}$$

It has been shown that the Henon Map exhibits chaotic behavior with $\alpha = 1.4$ and $\beta = 0.3$ [81].

The strange attractor produced by the Henon map is shown in Figure 4.1.

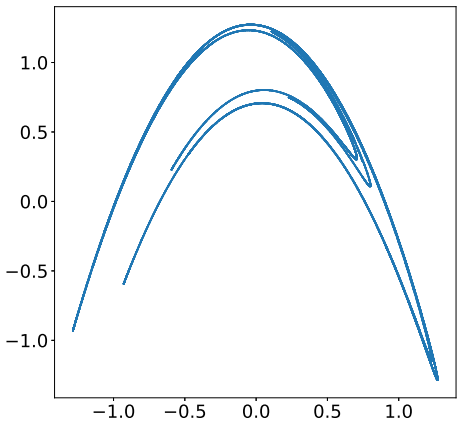
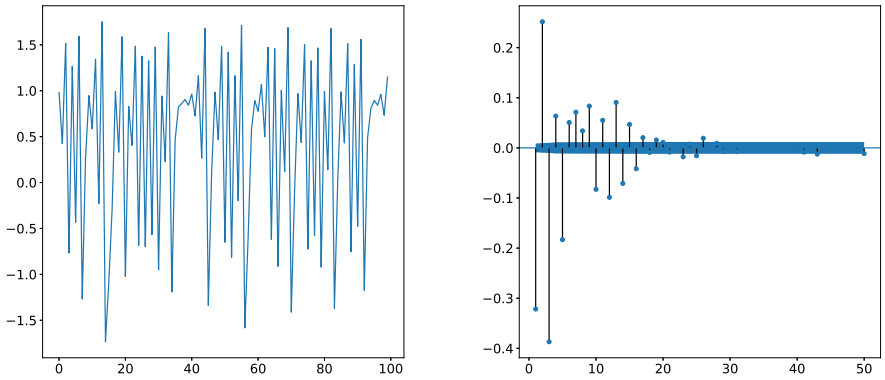


Figure 4.1: Henon Map Strange Attractor

Figure 4.2a shows a segment of the Henon Map time series for variable x and 4.2b its ACF.



(a) Synthetic Time Series (b) ACF plot

Figure 4.2: Henon Map Time Series

Henon's ACF shows no apparent structure with a very low auto-correlation values for time lags 28 onwards. This ACF tells us that, no seasonal component exists in the time series. In fact, it resembles the ACF of a time series with a constant mean and random values around that mean.

Table 4.1 shows the results of the Dickey-Fuller test of the Henon time series.

Table 4.1: Dickey-Fuller Results for Henon Map

ADF Statistic	**p-value**	**Critical values**
-54.862870	0.000000	1%: -3.430, 5%: -2.862, 10%: -2.567

The Dickey-Fuller test shows that the ADF statistic is much less than the 1% critical value ($-54.862870 \leq -3.430$) and the p-value is ≤ 0, which rejects the null hypothesis that the data has a unit root. This is, the time series is stationary.

Henon map has two Lyapunov exponents $\lambda \simeq 0.41922, -1.62319$[1]. By using Rosenstein's method [69], we calculated numerically the Maximum

[1]These exponents were calculated by Sprott in [81] with the method described in Chapter 5 of his book.

Lyapunov Exponents (MLE's) for the embedding dimensions $m \in [2, 10]$ (shown in Table 4.2).

Table 4.2: Henon MLE's calculated Numerically

Embedding Dimension	**MLE**	**% of Actual MLE**
$m = 2$	0.3387	80.8
$m = 3$	0.3337	79.6
$m = 4$	0.3235	77.2
$m = 5$	0.3115	74.3
$m = 6$	0.2974	70.9
$m = 7$	0.2822	67.3
$m = 8$	0.2661	63.5
$m = 9$	0.2763	65.9
$m = 10$	0.2577	61.5

The calculated MLE ranges from 80.8-61.5% of the actual largest exponent. According to the above analysis, the Henon Map is chaotic and does not exhibit a regular stationarity.

4.2.2 Lorenz System

The Lorenz System is an autonomous (does not explicitly depend on the independent variable) dissipative flow (a flow is a continuous dynamical system [80]). The Lorenz equation system is shown in Equation (4.2).

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z \end{aligned} \tag{4.2}$$

It has been shown that with $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ with initial conditions $x_0 = 0$, $y_0 = -0.01$, and $z_0 = 9$ the Lorenz system exhibits chaotic behavior.

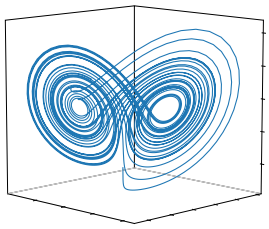
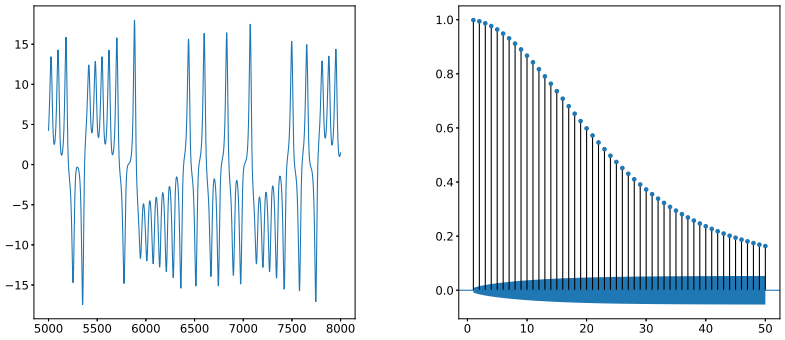


Figure 4.3: Lorenz System Strange Attractor

The strange attractor produced by the Lorenz system is shown in Figure 4.3.

Figure 4.4a shows a segment of the Lorenz system time series for variable x and 4.4b its ACF.



(a) Synthetic Time Series (b) ACF plot

Figure 4.4: Lorenz System

From the ACF plot it is possible to see that no apparent seasonal component is present and resembles entirely the ACF of a non stationary time series, which is not the case, since Lorenz is bounded within the limits of a strange attractor.

Table 4.3 shows the results of the Dickey-Fuller test of the Lorenz System time series.

Table 4.3: Dickey-Fuller Results for Lorenz System

ADF Statistic	**p-value**	**Critical values**
-18.511888	0.000000	1%: -3.430, 5%: -2.862, 10%: -2.567

The Dickey-Fuller test shows that the ADF statistic is much less than the 1% critical value (-18.511888) and the p-value is ≤ 0, which rejects the null hypothesis.

Lorenz system possesses three Lyapunov exponents $\lambda \simeq 0.9056, 0, -14.5723$. Table 4.4 shows the numerically calculated MLE's with $m \in [2, 10]$.

Table 4.4: Lorenz MLE's calculated Numerically

Embedding Dimension	**MLE**	**% of Actual MLE**
$m = 2$	0.01556	1.7
$m = 3$	0.01286	1.4
$m = 4$	0.01101	1.2
$m = 5$	0.00852	0.9
$m = 6$	0.00807	0.9
$m = 7$	0.00784	0.9
$m = 8$	0.00663	0.7
$m = 9$	0.00632	0.7
$m = 10$	0.00859	0.9

From the table it is possible to see that the numerically calculated MLE diverges from 1.7 - 0.7 % of the actual MLE. According to the above analysis, the Lorenz system is chaotic and does not exhibit a regular stationarity.

4.2.3 Mackey-Glass Equation

The Mackey-Glass equation is a non-linear time delay differential equation. Equation (4.3) shows the Mackey-Glass equation.

$$\frac{dx}{dt} = \beta x(t) + \frac{\alpha x(t-\tau)}{1 + x(t-\tau)^{10}} \tag{4.3}$$

where x is the series in time t and τ the time delay. Parameters $\alpha = 0.2$, $\beta = 0.1$ and $x(0) = 1.2$. With $\tau \geq 17$ the equation makes the system exhibit a chaotic behavior.

The strange attractor produced by the Mackey-Glass equation is shown in Figure 4.5. To construct this attractor a shift has to be applied to the time series and plot the shifted time series as coordinates.

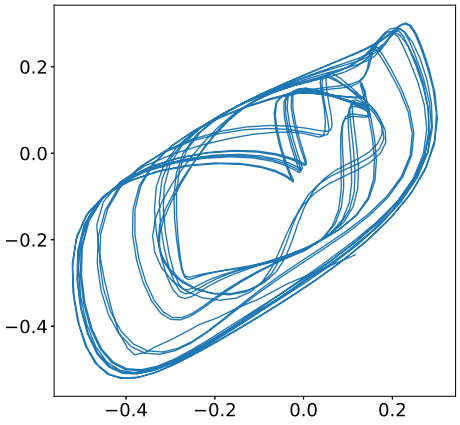
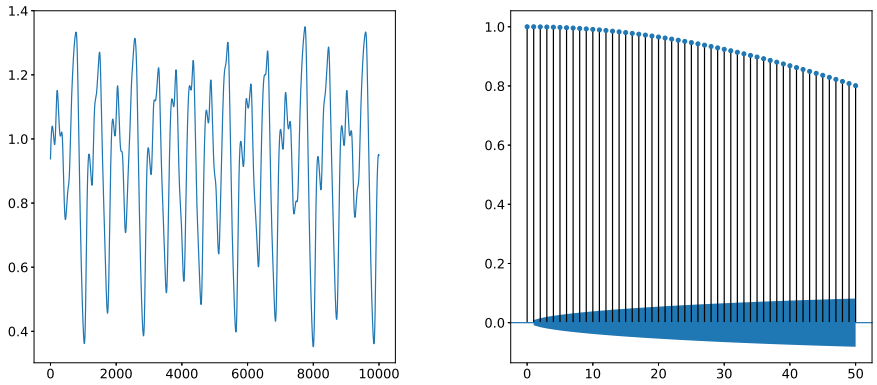


Figure 4.5: Mackey-Glass Attractor

Figure 4.6 shows the time series and ACF plot for the Mackey-Glass Equation.

Figure 4.6a shows a segment of the Mackey-Glass Equation time series and 4.6b its ACF.



(a) SyntheticTime Series (b) ACF plot

Figure 4.6: Mackey-Glass Equation

The ACF of the Mackey-Glass time series shows a non-stationary behavior, with a very slowly descending curve. There is no apparent structure which would indicate seasonal components. However, as in the case of the Lorenz system, the Mackey-Glass equation is bounded by a strange attractor, which indicates that it is stationary.

Table 4.5 shows the results of the Dickey-Fuller test of the Mackey-Glass time series.

Table 4.5: Dickey-Fuller Results for Mackey-Glass Equation

ADF Statistic	**p-value**	**Critical values**
		1%: -3.430,
-23.897187	0.000000	5%: -2.862,
		10%: -2.567

The Dickey-Fuller test shows that the ADF statistic is much less than the 1% critical value (-23.897187) and the p-value is ≤ 0, which rejects the null hypothesis that the time series has a unit root.

The Mackey-Glass equation has four Lyapunov exponents $\lambda \simeq 0.00860$, 0.00100, -0.03950, -0.05050. Table 4.6 shows the numerically calculated MLE's with $m \in [2, 10]$.

Table 4.6: Mackey-Glass MLE's calculated Numerically

Embedding Dimension	**MLE**	**% of Actual MLE**
$m = 2$	0.004772	55.5
$m = 3$	0.001098	12.8
$m = 4$	0.000688	8.0
$m = 5$	0.000302	3.5
$m = 6$	0.000216	2.5
$m = 7$	0.000292	3.4
$m = 8$	0.000227	2.6
$m = 9$	7.7e-05	0.9
$m = 10$	0.000226	2.6

We can see from the table that the numerically calculated MLE diverges from 55.5 - 0.9 % of the actual MLE.

4.2.4 Rössler Attractor

The Rössler Attractor is an autonomous dissipative flow. The solution to the attractor is shown in Equation (4.4).

$$\begin{aligned} \dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c) \end{aligned} \tag{4.4}$$

$(x, y, z) \in \mathbb{R}^3$ are dynamical variables and $(a, b, c) \in \mathbb{R}^3$ are parameters. the system exhibit a chaotic behavior with parameter values $a = b = 0.2$, and $c = 5.7$ with initial conditions $x = -9$, $y = 0$, and $z = 0$.

The strange attractor produced by the Rössler attractor is shown in Figure 4.7.

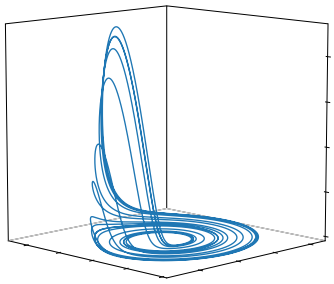
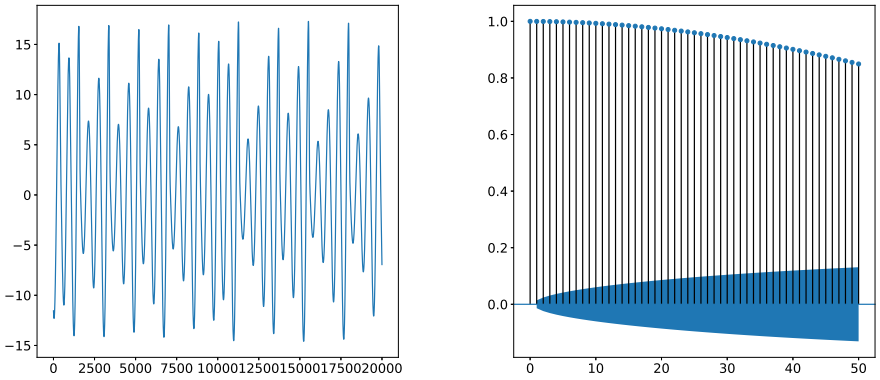


Figure 4.7: Rössler Attractor

Figure 4.8a shows a segment of the Rössler Attractor time series and 4.8b its ACF.



(a) Time Series (b) ACF plot

Figure 4.8: Rössler Attractor

The ACF of the Rössler time series shows a non-stationary behavior, with a very slowly descending curve. Again, this is similar to the Lorenz and

Mackey-Glass time series, which is known that they are stationary time series. There is no apparent structure that would indicate seasonal components.

Table 4.7 shows the results of the Dickey-Fuller test of the Rössler time series.

Table 4.7: Dickey-Fuller Results for Rössler Attractor

ADF Statistic	**p-value**	**Critical values**
-28.704474	0.000000	1%: -3.430, 5%: -2.862, 10%: -2.567

The test reveals an ADF statistic value of -28.704474, which is lower than the 1% critical value (-3.430). Also, the p-value is ≤ 0. These results refute the null hypothesis of the test.

The Rössler Attractor has three Lyapunov exponents $\lambda \simeq 0.0714,\ 0,\ -5.3943$. Table 4.8 shows the numerically calculated MLE's with $m \in [2, 10]$.

Table 4.8: Rössler Time Series MLE's calculated Numerically

Embedding Dimension	**MLE**	**% of Actual MLE**
$m = 2$	0.00682	9.6
$m = 3$	0.00234	3.3
$m = 4$	0.00154	2.2
$m = 5$	0.001402	2.0
$m = 6$	0.001196	1.7
$m = 7$	0.001092	1.5
$m = 8$	0.001042	1.5
$m = 9$	0.000982	1.4
$m = 10$	0.000809	1.1

The table with the MLE's for the Rössler time series indicates that the numerically calculated MLE diverges from 9.6 - 1.1 % of the actual MLE.

From this analysis we can conclude that the time series contains stationary but no seasonal components. Their MLE's are positive, which indicate chaotic behavior. However, the numerically calculated MLE's diverge significantly from the real (theoretical) MLE's.

4.3 Experiments and Results

Five methods were used to forecast the synthetic time series. NNDE (the method proposed in this thesis), ARIMA [4], Deterministic Nearest Neighbors (NN – [41]), RBF ANN's [6], and RNN's [35]. The methods were employed in two forecasting schemes: OSA and NSS. The experiments are divided in three aspects: short-term forecasts where the last 50 values are forecasted; long-term forecasts where the last 10,000 values are forecasted; and noise added forecasting, where noise is added to the synthetic time series.

4.3.1 Forecasting Tasks

Table 4.9 shows the proposed forecasting tasks and their parameters. Each of the tasks use both OSA and NSS forecasting schemes, varying only in the length of the training and validation sets. Also, in the case of the noisy forecast, the time series have noise added to them.

Table 4.9: Forecasting Tasks

Forecasting Task	Forecasting Schemes	Parameters
Short-term Forecast	OSA NSS	Training set length: 49,950 Validation set length: 50
Long-term Forecast		Training set length: 40,000 Validation set length: 10,000
Noisy Forecast		Training set length: 49,950 Validation set length: 50

4.3.2 Forecasting Methods Settings

Each forecaster require certain settings in order to be used. This section describes the setup of each of them.

NNDE

To obtain the embedding dimension (m), time delay (τ), and the neighborhood radius (ϵ) for the NNDE method, we generated several independent differential evolution runs. The parameters of these runs are described in Table 4.10.

Table 4.10: Differential Evolution Parameters

Parameter	**Value**
Population size	30
Maximum iterations	20
Scale factor	0.50
Recombination probability	0.75
Number of independent executions	30

The fitness function employed to measure the population is the MAPE (Mean Absolute Percentage Error) of the individual's $[m, \tau, \epsilon]$ parameters.

In this work we use 30 independent DE processes in order to provide statistical significance to the experiments. The best individual of all processes is retrieved along with its fitness.

Deterministic NN

The deterministic method of NN determines the embedding dimension and time delay using the False Neighbor algorithm and the Mutual Information algorithms, described by [41].

A slight modification was done for the Deterministic method. Instead of increment ϵ by a certain factor, the value obtained by the best OSA NNDE process is used, since this value comes from an optimization process and not by a trial and error heuristic.

ARIMA

For ARIMA we obtained the model parameters with the `auto.arima` function provided by R [65].

RBF ANN

The RBF network architecture implemented in this comparison had m input neurons, a single hidden layer (with $2m$ neurons) and a single output neuron, which produces the forecast. This architecture was selected since it has been previously used to forecast chaotic time series with fairly good results [85; 46; 92]. The implementation of ANN employed in this work was trained using the last 20,000 samples of the time series, since it caused the program to end abruptly when using the whole training set.

RNN

The RNN implemented has m input neurons, a single hidden layer (with 5 neurons), a simple LSTM loop for each neuron, and a single output neuron, which produces the forecast. This type of ANN allows the network to have loops in the connections, adding feedback and memory to the networks over time. This kind of ANN has been used to forecast chaotic time series [94] under the assumption that, an RNN can describe a dynamic system, precisely the kind of system that produces the chaotic time series used in this forecasting study.

4.3.3 Short-term Forecasting

Table 4.11 shows the parameters used by the forecasters to forecast the selected time series. The obtained parameters for the different methods change for each time series. The first column denotes the time series, the following two columns include the proposed NNDE OSA and NNDE NSS (which correspond to the forecasting methods described in Equations 2.46 and 2.47, respectively) the last three columns correspond to the Deterministic NN, ARIMA, and ANN methods. Note that both RBF and RNN share the same parameters.

Table 4.11: Forecasters Parameters for OSA and NSS

TS	NNDE OSA (m, τ, ϵ)	NNDE NSS (m, τ, ϵ)	NN (m, τ, ϵ)	ARIMA (p, d, q)	ANN RBF & RNN (m, τ)
henon	[4, 1, 0.061]	[10, 47, 3.581]	[2, 1, 0.061]	[4, 0, 5]	[2, 14]
lorenz	[47, 1, 3.138]	[2, 44, 31.773]	[3, 23, 3.138]	[0, 0, 0]	[3, 23]
mackey	[51, 23, 0.033]	[3, 50, 0.878]	[4, 60, 0.033]	[0, 0, 0]	[4, 60]
rossler	[17, 1, 1.892]	[4, 50, 31.891]	[4, 16, 3.506]	[0, 0, 0]	[4, 16]

Table 4.12 shows the MAPE scores of the forecasters when solving the OSA forecasting task. The table is organized with each line showing the scores for the indicated time series, and the columns correspond to the MAPE for each of the methods included in the comparison; the best result for each time series is highlighted in bold face. As the results show, NNDE obtains the best MAPE score in all of the time series. The difference of results between the rest of the forecasters is quite ample for this forecasting technique for the synthetic time series, with no other forecaster obtaining results similar to those produced by NNDE.

Table 4.12: MAPE Results for OSA Forecasting

TS	NNDE	NN	ARIMA	ANN RBF	RNN
henon	**0.319**	113.989	169.678	79.123	7.506
lorenz	**0.448**	65.605	4.896	57.180	1.353
mackey	**0.007**	3.113	2.081	0.032	6.798
rossler	**0.159**	8.168	103.522	42.040	15.761

Table 4.13 shows the results when solving the N-Step Simultaneous forecasting problem. The MAPE scores when using this technique increased drastically, compared to the results when using OSA. This behavior can be attributed to the inherent error propagation in chaotic systems. Nonetheless, the NNDE method shows better scores than the rest of the forecasters in henon and rossler.

Notice that some of the results of the ANN RBF, compared to Table 4.12, show an improvement over their OSA counterpart. This behavior can be attributed to a faster degradation of OSA performance versus NSS when certain conditions of the system dynamics that governs the time series are present [3]. In the case of the RNN its performance worsens compared to the OSA forecasting technique.

Table 4.13: MAPE Results for NSS Forecasting

TS	NNDE	NN	ARIMA	ANN RBF	RNN
henon	**87.285**	135.189	127.302	125.949	989.431
lorenz	51.588	255.224	86.862	**16.955**	57.057
mackey	0.977	49.353	1.545	**0.030**	86.205
rossler	**20.917**	113.637	32.336	34.402	113.673

4.3.4 Long-term Forecasting

This experiment will show the behavior of the forecasters in a very long term forecasting scenario. For this experiment, the training set consists of the first 40,000 samples of a synthetic chaotic time series, and the last 10,000 samples are used as validation set. Each forecasting method generates 50 simultaneous forecasts and will be repeated 200 times to complete 10,000 forecasts.

Table 4.14 shows the parameters used for the long term forecasting. Notice that, for every forecasting scenario, NNDE generates different parame-

ters. This happens because each forecasting scenario affects the landscape of the DE search space.

Table 4.14: Forecasters Parameters for Long Term Forecasting

TS	NNDE (m, τ, ϵ)	Deterministic NN (m, τ, ϵ)	ARIMA (p, d, q)	ANN RBF & RNN (m, τ)
henon	[1, 6, 3.581]	[2, 14, 0.061]	[4, 0, 5]	[2, 14]
lorenz	[16, 3, 1.657]	[3, 23, 3.138]	[0, 0, 0]	[3, 23]
mackey	[19, 16, 0.231]	[4, 60, 0.033]	[0, 0, 0]	[4, 60]
rossler	[12, 28, 10.233]	[4, 16, 3.506]	[0, 0, 0]	[4, 16]

Table 4.15 shows the MAPE scores of each forecaster. NNDE obtains the best MAPE scores in the lorenz and rossler time series, while RNN and ANN RBF obtain the best score in henon and mackey, respectively.

Table 4.15: MAPE Results for Long Term Forecasting

TS	NNDE	NN	ARIMA	ANN RBF	RNN
henon	182.885	339.408	183.050	162.082	**105.782**
lorenz	**9.577**	99.523	183.152	82.414	117.145
mackey	0.523	1.141	26.738	**0.111**	100.068
rossler	**7.725**	54.242	102.404	49.126	112.039

This case is quite interesting since, while NNDE is not as dominant as it was in the other experiments, obtained very good results in lorenz and rossler compared to the scores obtained by the other forecasters (almost an order of magnitude better with respect to the closest score). Both ANN's perform better for the henon time series, while for the mackey series NNDE comes second in performance. These results indicate that the proposed method excels at short term forecasting.

4.3.5 Noise

Considering that most time series contain deterministic white noise [71]. We tested the different synthetic time series with added noise whose magnitude is expressed as a Signal-to-Noise ratio (SNR) [74]. That is, for a given signal S, we add noise in the form:

$$X = S + G\sigma_S^2 r \tag{4.5}$$

where σ_S^2 is the variance of the time series, G is a pseudo-random number from a Gaussian distribution with 0 mean and standard deviation of 1 and r is the SNR of the noise we desire to add to the time series.

Table 4.16 indicates the parameters of the forecasters when noise is present. A new column named Noise-to-signal ratio (NSR) is added to indicate the level of noise added to each time series. Notice that the values for 0.00 ratio are duplicates of the former experiments (since they include no noise) and are included here for comparison and completeness. Again, we include two columns for the NNDE forecaster with the parameters obtained for the OSA and NSS methods.

Table 4.17 shows that when noise is present, the MAPE score of every forecaster worsens, although it does not necessarily gets worse directly from a lower noise ratio to a higher ratio. The NNDE forecaster outperforms the other forecasters in every noise amplitude except for $NSR = 0.20$ for mackey, where ANN RBF obtains a slight improvement over NNDE.

Similarly, Table 4.18 shows that when using the NSS method, the MAPE score of every forecaster increases, compared to a noise free environment. With this method NNDE still outperforms the other forecasters except in a few cases, where both of the ANN obtain better MAPE scores.

4.4 Chapter Conclusions

From the results, the forecasters we used to compare against NNDE had problems when dealing with both chaotic and noisy time series. ARIMA cannot produce a model when the size of the time series increases. Surprisingly, ARIMA is capable of producing a model when noise is introduced. NN with the deterministic method is unreliable in producing a good quality forecast. ANN RBF can produce in some cases better results than NNDE, but the implementation used of this forecaster is unable to train with more than 20,000 samples. This problem adds an extra complexity to this method, which is the selection of the more representative samples of a time series. RNN generates good forecast scores when used in an OSA environment, but behaves poorly when used to generate multiple forecasts simultaneously.

Most importantly, it was possible to test the theory presented in [41]. From our results, we found that the deterministic method falls short when testing all the possible embedded dimensions and time delays for a given time series, producing poor combinations of these parameters. We consider

Table 4.16: Forecasters Parameters with Noise-Added Time Series

TS	NSR	NNDE OSA (m, τ, ϵ)	NNDE NSS (m, τ, ϵ)	NN (m, τ, ϵ)	ARIMA (p, d, q)	ANN RBF & RNN (m, τ)
henon	0.00	[4, 1, 0.061]	[10, 47, 3.581]	[2, 14, 0.061]	[4, 0, 5]	[2, 14]
	0.05	[8, 1, 0.386]	[11, 47, 3.857]	[7, 4, 0.386]	[4, 0, 5]	[7, 4]
	0.10	[7, 1, 0.416]	[5, 48, 1.307]	[7, 6, 0.416]	[4, 0, 5]	[7, 6]
	0.15	[14, 1, 3.222]	[9, 49, 3.344]	[7, 11, 3.222]	[4, 0, 5]	[7, 11]
	0.20	[11, 1, 2.881]	[11, 4, 4.227]	[7, 13, 2.881]	[4, 0, 5]	[7, 13]
	0.25	[2, 2, 1.379]	[36, 13, 5.295]	[8, 11, 1.379]	[4, 0, 4]	[8, 11]
lorenz	0.00	[47, 1, 3.138]	[2, 44, 31.773]	[3, 23, 3.138]	[0, 0, 0]	[3, 23]
	0.05	[41, 3, 4.482]	[2, 44, 33.231]	[5, 18, 4.482]	[3, 0, 4]	[5, 18]
	0.10	[86, 1, 15.696]	[2, 41, 32.523]	[6, 17, 15.696]	[3, 0, 5]	[6, 17]
	0.15	[86, 5, 30.855]	[2, 22, 37.628]	[7, 18, 30.855]	[3, 0, 4]	[7, 18]
	0.20	[11, 50, 28.131]	[90, 6, 24.132]	[6, 18, 28.131]	[5, 0, 4]	[6, 18]
	0.25	[44, 12, 23.694]	[2, 50, 47.014]	[8, 20, 23.694]	[3, 0, 3]	[8, 20]
mackey	0.00	[51, 23, 0.033]	[3, 50, 0.878]	[4, 60, 0.033]	[0, 0, 0]	[4, 60]
	0.05	[19, 39, 0.320]	[4, 32, 1.063]	[6, 1, 0.320]	[0, 0, 5]	[6, 1]
	0.10	[30, 34, 0.452]	[3, 50, 1.005]	[7, 1, 0.452]	[1, 0, 0]	[7, 1]
	0.15	[62, 18, 0.430]	[4, 28, 0.724]	[6, 4, 0.430]	[1, 0, 0]	[6, 4]
	0.20	[19, 24, 0.794]	[4, 30, 1.252]	[6, 2, 0.794]	[2, 0, 0]	[6, 2]
	0.25	[14, 12, 0.796]	[4, 35, 1.332]	[7, 1, 0.796]	[3, 0, 0]	[7, 1]
rossler	0.00	[17, 1, 1.892]	[4, 50, 31.891]	[4, 16, 1.892]	[0, 0, 0]	[4, 16]
	0.05	[12, 29, 14.125]	[21, 7, 33.619]	[6, 100, 14.125]	[0, 0, 5]	[6, 100]
	0.10	[72, 9, 16.388]	[20, 8, 36.624]	[7, 99, 16.388]	[0, 0, 5]	[7, 99]
	0.15	[12, 15, 21.643]	[4, 50, 33.548]	[6, 100, 21.643]	[1, 0, 0]	[6, 100]
	0.20	[37, 9, 12.092]	[5, 37, 40.471]	[7, 97, 12.092]	[2, 0, 0]	[7, 97]
	0.25	[74, 20, 45.856]	[5, 42, 45.856]	[7, 98, 45.856]	[3, 0, 0]	[7, 98]

Table 4.17: MAPE results for OSA Forecasting with Added SNR

TS	NSR	NNDE	NN	ARIMA	RBF	RNN
henon	0.00	**0.319**	31.753	211.046	47.281	7.5061
	0.05	**15.237**	253.962	188.259	93.987	28.230
	0.10	**33.078**	326.300	218.753	64.651	49.874
	0.15	**48.180**	77.328	760.590	81.790	248.704
	0.20	**43.026**	75.814	151.966	103.551	93.038
	0.25	**96.337**	444.042	222.862	102.539	151.824
lorenz	0.00	**0.448**	3.396	166.105	32.312	1.353
	0.05	**9.151**	14.354	159.186	80.744	50.408
	0.10	**17.120**	25.161	198.100	102.354	24.003
	0.15	**39.163**	62.044	300.294	99.026	93.472
	0.20	**50.620**	222.476	1435.675	105.952	424.042
	0.25	**76.062**	196.988	705.226	93.497	181.775
mackey	0.00	**0.007**	0.104	2.081	0.010	6.798
	0.05	**0.849**	1.131	16.973	1.141	4.802
	0.10	**2.018**	2.499	20.015	2.904	4.953
	0.15	**2.724**	3.439	19.339	3.957	4.758
	0.20	4.967	5.194	19.765	**4.963**	6.478
	0.25	**6.010**	6.615	21.289	6.808	6.592
rossler	0.00	**0.159**	0.818	103.522	248.621	15.761
	0.05	**8.335**	16.704	102.967	31.497	12.192
	0.10	**17.230**	21.405	105.699	105.567	19.543
	0.15	**21.132**	29.328	104.243	94.619	34.688
	0.20	**23.385**	31.274	108.544	102.099	31.981
	0.25	**48.479**	52.965	120.252	103.048	66.998

Table 4.18: MAPE Results for NSS Forecasting with Added SNR

TS	NSR	NNDE	NN	ARIMA	ANN RBF	RNN
henon	0.00	**87.285**	132.357	118.83	158.800	989.431
	0.05	**90.644**	128.861	118.31	124.668	125.310
	0.10	**88.090**	349.271	133.309	120.545	389.232
	0.15	**103.239**	374.920	207.065	113.352	526.951
	0.20	**83.460**	113.314	119.580	237.858	285.278
	0.25	**111.629**	188.995	141.141	126.412	186.991
lorenz	0.00	51.588	97.786	100.04	**33.904**	57.057
	0.05	55.197	167.097	99.504	**27.006**	108.391
	0.10	60.775	202.857	100.582	104.340	**55.160**
	0.15	**72.316**	241.777	101.632	99.056	75.765
	0.20	107.659	966.083	121.136	**106.246**	462.774
	0.25	**93.495**	389.837	102.963	94.129	244.057
mackey	0.00	0.977	8.869	99.13	**0.042**	86.205
	0.05	**1.506**	2.892	101.236	2.440	95.356
	0.10	**2.441**	3.171	102.444	5.620	190.476
	0.15	**2.976**	3.570	99.921	3.161	90.373
	0.20	**5.310**	7.725	101.407	5.870	14.423
	0.25	**6.440**	9.103	99.782	12.699	94.421
rossler	0.00	**20.917**	36.058	99.862	126.380	113.673
	0.05	**27.308**	65.372	100.017	83.123	152.874
	0.10	**37.799**	99.792	100.027	103.247	109.746
	0.15	**36.871**	86.372	99.452	100.097	111.372
	0.20	**32.762**	64.831	100.270	101.519	42.464
	0.25	**58.726**	130.058	100.191	100.908	224.187

that, a revision of this method is necessary, and that more robust and search intensive algorithms should be considered.

The difficulty to forecast synthetic time series varies vastly from one series to another. Henon proved to be the most difficult to predict for NNDE, followed by the Lorenz system, the Rössler attractor, and the Mackey-Glass equation. We consider that this situation occurs because of the nature of the time series, since the Henon map is the time series with the highest variability (this is, the rate of negative and positive changes from one time instant to another). However, in the next chapters will be very clear that synthetic time series are in general the easiest time series to forecast.

In summary, over all the experiments performed for the set of forecasting techniques used in this comparison, we conclude that NNDE provides the best forecast accuracy in most cases.

Chapter 5

Case Study: Wind Speed Forecasting

Wind speed forecasting is a very interesting problem, both because of its nature and because of its importance for managing the energy that can be supplied by wind. Forecasting wind speed is necessary for many reasons. The first and most important is to identify the best locating to place wind farms, since wind speed varies from region to region. Also, since wind speed is not constant, and electricity consumption in a large scale view cannot be interrupted, so it makes it necessary to know when the wind speed for a certain wind farm will decrease in order to provide energy from another source.

In past works there has been strong evidence that wind speed is of chaotic nature [10; 29]. As this chapter exposes with the use of Rosenstein's Algorithm for calculating the Maximum Lyapunov Exponent, we can conclude that this affirmation is correct.

This chapter shows a comparison between time series forecasters, where NNDE proves to be competitive in forecasting wind speed.

5.1 Related Work

We surveyed the recent literature and found a vast collection of wind forecasting models that deal with its complicated behavior. Accurate wind speed forecasting is an active multidisciplinary research field. Eolic power generation is directly related to wind speed, which presents a high level of po-

tential harmful uncertainty to the efficiency of energy dispatch and management. Therefore, one of the biggest goals in wind speed forecasting is to reduce and manage the uncertainty with accurate models with the aim of increasing added value to the wind power generation. Nevertheless, wind speed prediction is difficult to perform accurately, and requires more than the traditional linear correlation-based models (i.e., Auto-Regressive Integrating Moving Average). Wind speed dynamics presents both strong chaotic and random components [49], which must be modeled and explained from the non-linear dynamics perspective.

We found in the literature a diverse collection of wind power and speed forecasting models to meet the requirements predicting at specific time horizons. Some of them are discussed in the remaining of this section.

The work of [90], is a comprehensive survey that organizes the forecasting models according to prediction horizons, e.g., immediate short time (8-hour ahead), short term (one-day ahead), and long term (greater than one-day ahead). [5] states that, for short term forecasting horizons (from 1 to 3 hours ahead), statistical models are more suitable than other kinds of models. In contrast, for longer horizons, the alternative methods perform better than the pure statistical models.

[57] present a comprehensive review where they cite an important number of wind power and speed forecasting models, compared by standard error measures and length of the prediction horizon. Regarding recent and concrete wind speed forecasting models, we find that the most promising results are hybrid approaches. [57] state that the Adaptive Neuro-Fuzzy Inference System (ANFIS) and Artificial Neural Networks (ANN) models. The paper compares ANFIS, ANN, and ANFIS+ANN model, presenting better performance improving by 5% in average. The hybrid model presents a MAPE improvement of 25% for 24-hour step ahead forecasts.

Another interesting multivariate model is proposed in [7] for one step ahead forecasting, using Nonlinear Autoregressive Exogenous Artificial Neural Networks (NARX). The NARX model considers wind speed, wind direction, atmospheric pressure, air temperature, solar radiation, and relative humidity. The model is compared against ARIMA model; NARX reports a precision improvement between 5.5% and 10%, and 12.8% for one-hour and ten-minute sample period time series.

[16] compare and analyze the effectiveness of WPPT (Wind Power Prediction Tool) with GWPPT (Generalized Wind Power Prediction Tool) using wind power data of 4 wind turbines located in Denmark. WPPT is also com-

pared with Non-parametric Kernel Regression, Mycielsky's Algorithm [34], Autoregressive (AR), and Vector Autoreggresive (VAR). The time series has measurements every 10 minutes. The results show that GWPPT is the most competitive method for 12-hour ahead forecasting (72 steps). Nevertheless, for one step ahead AR and VAR were the most accurate methods, improving approximately 8.6% w.r.t. WPPT and GWPPT models. GWPPT is useful for 12-hour ahead forecasting, improving an average of 7% w.r.t. the WPPT model, a 41% improvement w.r.t. the AR model, and 50.8% w.r.t. the VAR model.

[39], present a one-day ahead wind power forecasting using a hybrid method based on the combination of the enhanced boosting algorithm and ARMA (ARMA-MS). An ARMA model is selected with parameters $p = 1$ and $q = 1$ for the AR and MA components, respectively. The ARMA-MS algorithm is based on the weighed combination of several ARMA forecasters. ARMA-MS is tested with wind power data from the east coast of the Jiangsu Province.

Despite the plethora of algorithms and models to forecast wind speed, the authors found few articles using non-linear time series theory [72; 77]. Non-linear time series theory is useful to identify the structure and attractors, and predict time series with non-linear and potential chaotic behavior, e.g., the Simple Non-Linear Forecasting Model based on Nearest Neighbors, proposed in [41].

We compare one of the simplest non-linear forecasting models (NN), with its equivalent statistical counterpart such as ARIMA. Based on experimental results, we show that the simple Non-linear model outperforms the statistical model ARIMA when we compare forecast accuracy for non-linear time series for one day ahead forecasting. Taking NN as the base model, we report the accuracy of a set of NN based models composed by a deterministic version of NN. The phase space reconstruction is given by the sub-sampling interval or time delay τ and the system dimension m; those parameters are used to produce the characteristic vectors known as delay vectors. τ and m are determined using mutual information and false nearest neighbors, respectively. Fuzzy Forecast (FF – [23]), that can be considered as a kind of Fuzzy NN, produces forecasts using fuzzy rules applied to the delay vectors. ANN-cGA [68] evolves the architecture of ANN's to minimize the forecasting error. As part of the evolved architecture, it determines the relevant inputs to the neural network forecaster. This concept is closely related to the phase space reconstruction process. EvoDAG [28] evolves forecasting functions that take

as arguments part of the history of the time series. Finally, we use NNDE (which is the basis of this thesis), which, by using a Differential Evolution process, obtains a set of optimized parameters for the given time series.

In past works there has been strong evidence that wind speed is of chaotic nature [10; 29]. The results corroborated in this chapter, by means of the use of Rosenstein's Algorithm for calculating the Maximum Lyapunov Exponent also suggest that wind speed time series are chaotic. Also, in this chapter we will show that NNDE can be competitive in forecasting wind speed.

5.2 Data Analysis

This section provides a brief analysis of the data used for this case study. It will focus on the completeness of the data sets and some of their characteristics, such as its seasonality and chaotic nature. As in the previous chapter, the seasonality is measured with the Auto Correlation plot of the time series and the chaotic component is determined with the Maximum Lyapunov Exponent calculated numerically.

5.2.1 Data Sets

To compare the techniques, 20 data sets from two different origins were selected. One is a selection of 10 of stations from the compilation "Six- and Three-Hourly Meteorological Observations from 223 Former U.S.S.R. Stations (NDP-048)" [67]. The other 10 data sets are a subset of the network of weather stations located in Michoacán, Mexico [8]. Every data set contains wind speed measurements collected at 10 meters of height.

These data sets were selected since they present different problems encountered in non-synthetic experimentation, most importantly: noise, outliers, data precision, and missing values. In the particular case of the Russian stations, wind speed is represented by integers, with no decimals. This adds a layer of noise, and data lacks granularity. The Mexican stations have one decimal digit.

While there are practically no missing values in the Russian data sets, the Mexican data sets lack many measurements. Because of this, some adjustments to train the forecasters and measure their performance were necessary. Those adjustments are described in Section 5.3.2.

Also, mostly in the Mexican data sets, some outliers were identified, both in the training and validation sets of some stations. As with the missing values, a few adequations were implemented in the forecasters to correctly treat these data sets.

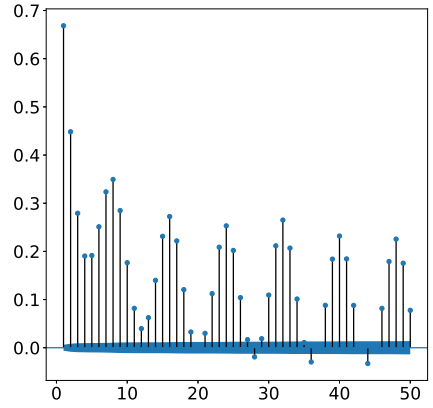
Table 5.1 show the names and origin of the data sets used.

Table 5.1: Data Sets Used

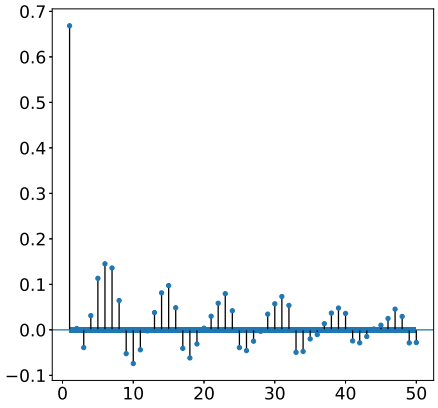
Russian Stations	Mexican Stations
20891	aristeomercado
22641	cointzio
22887	corrales
23711	elfresno
24908	lapalma
27947	lapiedad
28722	malpais
29231	markazuza
30230	melchorocampo
37099	patzcuaro

5.2.2 Auto-Correlation Analysis of the Data Sets

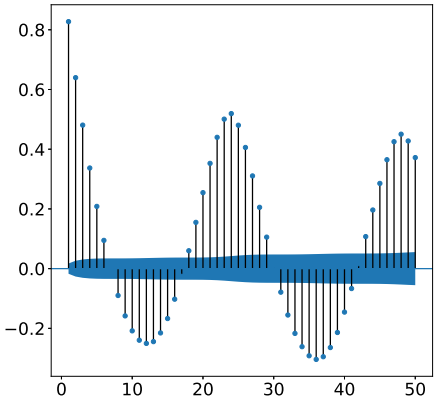
To identify seasonality in a time series it is necessary to analyze its auto-correlation plot. The auto-correlation and partial auto-correlation plots for the time series 24908 and lapiedad are shown in Figure 5.1.



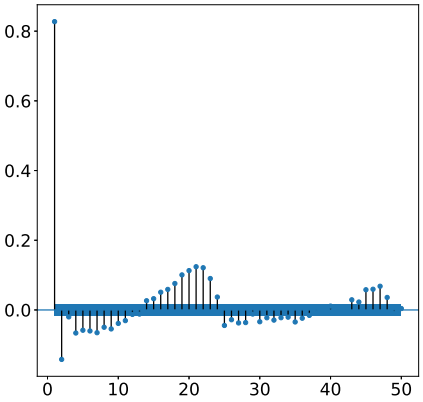
(a) ACF for 24908



(b) PACF for 24908



(c) ACF for lapiedad



(d) PACF for lapiedad

Figure 5.1: ACF and PACF of 24908 and lapiedad

In every plot there is an increased auto-correlation every 8 and 24 lags for the Russian and Mexican stations, respectively. Many time lags exhibit auto-correlation values that exceed the 5% significance threshold; this fact indicates that the null hypothesis that there is no correlation for those time lags can be rejected.

5.2.3 Dickey-Fuller Test of the Data Sets

In order to determine if the data sets are stationary the Dickey-Fuller test was used. Table 5.2 shows the results of the Dickey-Fuller test applied to the wind time series.

Table 5.2: Dickey-Fuller Test Results of Wind Speed Time Series

Time Series	**ADF Statistic**	**p-value**
20891	-32.201210	0.000000
22641	-27.683992	0.000000
22887	-27.512232	0.000000
23711	-25.905334	0.000000
24908	-25.946519	0.000000
27947	-26.740374	0.000000
28722	-27.764511	0.000000
29231	-26.060155	0.000000
30230	-28.532636	0.000000
37099	-27.419562	0.000000
aristeomercado	-9.435058	0.000000
cointzio	-5.698314	0.000001
corrales	-4.032036	0.001252
elfresno	-12.132229	0.000000
lapalma	-16.976743	0.000000
lapiedad	-15.414692	0.000000
malpais	-4.686351	0.000089
markazuza	-5.848714	0.000000
melchorocampo	-7.348543	0.000000

The critical values for the ADF statistic are: 1%: -3.430, 5%: -2.862, and 10%: -2.567.The Dickey-Fuller test shows that every time series is stationary.

5.2.4 Maximum Lyapunov Exponent Analysis of the Data Sets

Wind speed has been identified as chaotic or as a non linear system [29; 36; 10; 37]. To test these assertions we estimated the maximum Lyapunov exponents of the wind data sets.

The exponents are shown in Table 5.3. These exponents were calculated in the range of $m \in [2, 10]$. For the sake of comparison and as reference, Table 5.3 includes the Lyapunov exponents of one time series not used in this work, the Logistic Map [54].

Table 5.3: Lyapunov Exponents of the Data Sets

Time Series	**Minimum Lyapunov Exponent**	**Maximum Lyapunov Exponent**	**Average Lyapunov Exponent**
20891	0.055 75	0.197 37	0.103 54
22641	0.0773	0.1798	0.128 99
22887	0.044 41	0.164 26	0.088 51
23711	0.088 88	0.1388	0.111 07
24908	0.102 56	0.148 84	0.126 27
27947	0.019 75	0.121	0.066 21
28722	0.056 11	0.176 67	0.108 33
29231	0.081 37	0.180 23	0.114 05
30230	0.0789	0.120 83	0.106 41
37099	0.095 89	0.147 97	0.117 65
aristeomercado	0.103 84	0.160 13	0.129 79
cointzio	0.113 16	0.204 15	0.152 03
corrales	0.063 79	0.133 01	0.107 27
elfresno	0.0644	0.145 74	0.109 85
lapalma	0.103 75	0.119 46	0.111 18
lapiedad	0.060 09	0.181 09	0.117 37
malpais	0.111 16	0.157 28	0.141 08
markazuza	0.097 06	0.209 36	0.157 35
melchorocampo	0.061 59	0.186 38	0.134 15
patzcuaro	0.0426	0.159 05	0.096 48
logistic map	0.315 07	0.592 13	0.4832

The calculated exponents are consistent with the ones observed in the literature. [29] reported an MLE of 0.115 in their data set, while the average MLE obtained by all of the data sets used in this thesis is 0.116379. Those exponents indicate that the data contains predictable components (seen in part in their respective ACF), but they also explain the difficulty to predict

this kind of data in the long term.

5.3 Experiments and Results

An often required wind forecasting task is to forecast the wind speed for the following day. In order to gather more performance information than just one execution of the different forecasting algorithms, we perform a One Day Ahead forecast for the last 10 days of each time series. In performing ODA forecasting, the forecaster generates the number of samples contained in one day of observations at a time.

After we have forecasted one whole day, we consider time advances and the next day of real wind speed observation is available. Since we are forecasting for 10 days, the last 10 days of measures of each data set were saved in the validation set; the rest of the data was used as training set. The sampling period of the Russian stations is three hours, while the sampling period of the Mexican stations is one hour. The number of samples used for training varied depending of the time series from 875 to 25,000 samples, while the length of the validation sets were 80 and 240 for the Russian and Mexican time series, respectively.

5.3.1 Experiment Settings

For this case study a set of forecasting techniques was proposed. The forecasters used are: ARIMA, NN, NNDE, FF, ANNcGA, and EvoDAG [28]. Each forecasting technique has different modeling considerations, which are described next:

ARIMA - The ARIMA implementation we used was the one included in the R statistical package [65]. The order of the ARIMA model was estimated using the `auto.arima` function.

NN - To determine the NN parameters (m, τ, and ϵ) we used the deterministic approach described in [41]. The deterministic approach uses the Mutual Information algorithm to obtain τ and the False Nearest Neighbors algorithm to find an optimal m; ϵ is found by testing the number of neighbors found for an arbitrary ϵ value, which is updated by the rule $\epsilon \leftarrow \epsilon \times 1.2$ when not enough neighbors are found.

NNDE - In NNDE the NN parameters are found by a DE optimization, where individuals are vectors of the form $[m,\ \tau,\ \epsilon]$. Because of the stochastic nature of DE, this optimization process is executed 30 independent times. The set of parameters that yield the least error score is the one used to forecast.

FF - This technique compiles a set of fuzzy rules that describe the time series by using delay vectors of dimension m and time delay τ. These parameter values are set to the same as those obtained by the deterministic method used by NN [41; 1]. Since the time series contains outliers, FF uses a simple filter which replaces any value greater than 6σ (σ is the standard deviation of the time series) with the missing value indicator.

ANNCGA - This method determines the optimal topology of a MLP using Compact Genetic Algorithms. The optimization process consists in finding the optimal set of inputs (past observations), the number of hidden neurons in the hidden layer, and the training algorithm.

EvoDAG - EvoDAG uses its default parameters and m is set to three days behind.

The parameters used by the different methods for the OSA forecasting method are shown in Table 5.4.

Table 5.5 shows the parameters used by the techniques for ODA forecasting.

Once the different forecasting models were trained, we proceeded to test their forecasts for the proposed forecasting task.

5.3.2 Performance Analysis

As previously indicated, the Mexican Data Sets present missing values in both the training and validation sets. To preserve the integrity of the results, when forecasting, if the value to predict is a missing value that measure does not contribute to the error score of the forecaster. If a missing value is found in the forecasting delay vector the forecaster returns the last known value of the time series as the forecast.

To measure the performance of the forecasters, we used the Symmetric Mean Average Percentage Error (SMAPE).

Table 5.4: Parameters of the Forecasting Techniques for OSA

Time Series	NN [m, τ, ϵ]	NNDE [m, τ, ϵ]	ARIMA (p, d, q)	ANN [m, h, TM]	FF [m, τ]
20891	[8, 1, 6.3197]	[50, 68, 15.3355]	(1, 1, 5)	[57, 24, bfgs]	[8, 1, 20]
22641	[7, 6, 6.3197]	[40, 34, 7.0342]	(0, 1, 5)	[59, 39, bfgs]	[7, 6, 20]
22887	[8, 13, 4.3887]	[20, 78 3.9590]	(3, 1, 2)	[61, 64, bfgs]	[8, 13, 20]
23711	[7, 5, 1.0206]	[30, 69, 4.3251]	(1, 1, 1)	[54, 35, bfgs]	[7, 5, 20]
24908	[7, 1, 2.1164]	[37, 28, 4.9439]	(3, 0, 3)	[62, 59, bfgs]	[7, 1, 20]
27947	[6, 1, 2.1164]	[29, 59, 7.7369]	(2, 1, 4)	[39, 37, bfgs]	[6, 1, 20]
28722	[6, 1, 3.0477]	[7, 25, 0.0000]	(3, 1, 1)	[45, 47, bfgs]	[6, 1, 20]
29231	[6, 5, 5.2664]	[17, 27, 7.9497]	(5, 1, 2)	[41, 35, bfgs]	[6, 5, 20]
30230	[6, 1, 3.0477]	[12, 50, 3.2369]	(1, 1, 5)	[33, 25, bfgs]	[6, 1, 20]
37099	[6, 1, 1.0206]	[23, 53, 2.8868]	(2, 1, 3)	[43, 63, bfgs]	[6, 1, 20]
aristeomercado	[8, 8, 10.92052]	[43, 4, 25.4163]	(1, 0, 2)	[16, 4, bfgs]	[8, 8, 20]
cointzio	[6, 6, 4.3887]	[3, 24, 7.7715]	(2, 1, 3)	[10, 2, bfgs]	[6, 6, 20]
corrales	[7, 6, 4.3887]	[47, 93, 17.6837]	(0, 1, 4)	[5, 60, rprop]	[7, 6, 20]
elfresno	[6, 9, 0.4101]	[2, 40, 1.3014]	(3, 1, 4)	[25, 16, gdx]	[6, 9, 20]
lapalma	[5, 5, 6.3197]	[5, 23, 3.3709]	(0, 1, 5)	[5, 21, cg]	[5, 5, 20]
lapiedad	[5, 10, 4.3887]	[15, 3, 19.3116]	(2, 0, 5)	[32, 6, gdm]	[5, 10, 20]
malpais	[9, 1, 116.8422]	[13, 10, 30.9695]	(0, 1, 2)	[64, 5, gdm]	[9, 1, 20]
markazuza	[5, 1, 2.5397]	[1, 37, 0.0000]	(3, 1, 4)	[53, 19, bfgs]	[5, 1, 20]
melchorocampo	[5, 1, 4.3887]	[6, 3, 7.5547]	(1, 0, 2)	[39, 15, rprop]	[5, 1, 20]
patzcuaro	[11, 1, 10.9205]	[23, 1, 6.9086]	(5, 1, 0)	[29, 3, rprop]	[11, 1, 20]

Table 5.5: Parameters of the Forecasting Techniques for ODA

Time Series	NN [m, τ, ϵ]	NNDE [m, τ, ϵ]	ARIMA (p, d, q)	ANN [m, h, TM]	FF [m, τ]
20891	[8, 1, 6.3197]	[43, 51, 12.7127]	(1, 1, 5)	[57, 24, bfgs]	[8, 1, 20]
22641	[7, 6, 6.3197]	[26, 95, 6.3276]	(0, 1, 5)	[59, 39, bfgs]	[7, 6, 20]
22887	[8, 13, 4.3887]	[9, 1, 3.3457]	(3, 1, 2)	[61, 64, bfgs]	[8, 13, 20]
23711	[7, 5, 1.0206]	[29, 69, 4.4676]	(1, 1, 1)	[54, 35, bfgs]	[7, 5, 20]
24908	[7, 1, 2.1164]	[14, 100, 2.1546]	(3, 0, 3)	[62, 59, bfgs]	[7, 1, 20]
27947	[6, 1, 2.1164]	[14, 50, 13.8344]	(2, 1, 4)	[39, 37, bfgs]	[6, 1, 20]
28722	[6, 1, 3.0477]	[16, 20, 6.1942]	(3, 1, 1)	[45, 47, bfgs]	[6, 1, 20]
29231	[6, 5, 5.2664]	[43, 7, 8.1657]	(5, 1, 2)	[41, 35, bfgs]	[6, 5, 20]
30230	[6, 1, 3.0477]	[39, 29, 8.0562]	(1, 1, 5)	[33, 25, bfgs]	[6, 1, 20]
37099	[6, 1, 1.0206]	[1, 49, 28.0248]	(2, 1, 3)	[43, 63, bfgs]	[6, 1, 20]
aristeomercado	[8, 8, 10.9205]	[7, 16, 22.7237]	(1, 0, 2)	[16, 4, bfgs]	[8, 8, 20]
cointzio	[6, 6, 4.3887]	[1, 29, 7.7595]	(2, 1, 3)	[10, 2, bfgs]	[6, 6, 20]
corrales	[7, 6, 4.3887]	[24, 1, 19.3118]	(0, 1, 4)	[5, 60, rprop]	[7, 6, 20]
elfresno	[6, 9, 0.4101]	[5, 36, 18.9346]	(3, 1, 4)	[25, 16, gdx]	[6, 9, 20]
lapalma	[5, 5, 6.3197]	[1, 40, 0.0000]	(0, 1, 5)	[5, 21, cg]	[5, 5, 20]
lapiedad	[5, 10, 4.3887]	[11, 2, 21.1117]	(2, 0, 5)	[32, 6, gdm]	[5, 10, 20]
malpais	[9, 1, 116.8422]	[41, 2, 19.5309]	(0, 1, 2)	[64, 5, gdm]	[9, 1, 20]
markazuza	[5, 1, 2.53976]	[24, 1, 10.8781]	(3, 1, 4)	[53, 19, bfgs]	[5, 1, 20]
melchorocampo	[5, 1, 4.38871]	[1, 59, 3.1399]	(1, 0, 2)	[39, 15, rprop]	[5, 1, 20]
patzcuaro	[11, 1, 10.92052]	[24, 1, 14.1787]	(5, 1, 0)	[29, 3, rprop]	[11, 1, 20]

5.3.3 One Step Ahead Forecasting

In One Step Ahead Forecasting, the forecasting techniques generate one prediction by using the existing data as a training set. To make multiple OSA forecasts this process is repeated by incorporating the actual values from the validation set into the training set. The number of repetitions is defined by the forecasting task discussed in Section 5.3.1.

The following tables show the SMAPE scores of the forecasting techniques. The rows show the stations while the columns are the forecasting techniques. The method that produces the best score for each station is indicated in bold face. The lowest the error the better.

Table 5.6: SMAPE Results for OSA Forecasting

Station	NN	NNDE	ARIMA	ANNCGA	FF	EvoDAG	Naïve
20891	34.5254	**29.6564**	37.4543	177.1883	39.8560	34.7307	34.6054
22641	63.3452	**47.8307**	61.5878	156.2629	62.7660	61.9290	50.2778
22887	90.8306	**70.5060**	96.0223	168.8161	80.5303	85.3416	75.2674
23711	105.0322	**71.5152**	109.455	154.7403	111.6042	103.5979	78.7879
24908	145.6395	**91.8248**	146.1459	183.1195	145.5900	148.2764	102.7027
27947	51.7676	**47.7467**	52.5265	170.3111	55.7703	52.4430	51.6444
28722	77.4884	**56.7134**	78.1059	170.8697	83.4289	79.4778	60.6633
29231	52.3358	**48.1392**	53.8060	160.5618	65.0320	54.6628	53.5589
30230	136.3110	**121.5278**	150.1244	173.2819	149.5682	138.8897	134.6154
37099	40.5067	**36.3603**	40.0351	130.4847	48.4840	41.6525	41.7119
aristeomercado	**28.8245**	29.4727	49.3668	37.0639	30.3345	36.0221	30.3345
cointzio	19.8073	**19.1669**	25.7577	26.3309	20.0322	27.0801	19.6683
corrales	25.6525	23.8028	28.6350	29.6223	**22.1857**	27.6645	23.5479
elfresno	30.1826	29.1501	36.7167	63.1050	30.8922	62.3659	**29.1058**
lapalma	31.4742	27.9533	28.5202	34.2672	**27.8067**	36.4984	28.7231
lapiedad	29.5564	**29.5855**	29.9047	40.2763	31.0024	40.7741	29.9047
malpais	44.8658	**27.0334**	29.9755	36.5165	28.9620	59.1462	28.7169
markazuza	27.0431	**26.9222**	37.8807	36.1454	33.9183	41.9725	27.3552
melchorocampo	24.1748	**23.4114**	25.2651	35.8290	27.0689	26.6881	25.3230
patzcuaro	31.1288	**23.8355**	61.0764	37.2044	28.6106	52.9107	28.4476

NNDE obtains most of the best scores for this metric, followed by ANN and FF. NNDE obtains the best scores for the Mexican stations in six out of ten, while with the Russian time series in all of them. NNDE performs better than the rest of the forecasting techniques on the majority of the time series, followed closely by the ANN and FF techniques.

5.3.4 One Day Ahead Forecasting

For this scenario each forecaster generates a day of estimations at a time. Once these forecasts are made, one day of observations are taken from the validation set and are incorporated into the training set (replacing the values introduced by the forecaster), and the process is repeated until 10 days of forecasts are completed.

Table 5.7 shows the SMAPE scores of the 10 day forecasts produced by the different forecasting techniques.

Table 5.7: SMAPE Results for ODA Forecasting

Station	NN	NNDE	ARIMA	ANNCGA	FF	EvoDAG
20891	40.701	**33.552**	52.485	177.667	47.349	42.287
22641	71.605	**70.466**	74.538	138.389	76.620	73.837
22887	100.261	**91.541**	183.042	162.067	123.836	94.351
23711	111.488	**86.782**	91.577	144.784	120.273	106.386
24908	146.980	**90.196**	138.607	183.813	144.755	147.510
27947	57.268	**53.114**	56.960	166.419	64.736	57.073
28722	84.692	**74.403**	82.507	167.003	88.769	83.761
29231	55.995	**51.058**	60.928	160.847	64.375	56.976
30230	134.553	**125.952**	170.713	168.304	147.326	132.347
37099	42.350	**40.640**	42.964	116.413	49.845	41.770
aristeomercado	37.395	**38.259**	48.956	189.804	62.502	49.938
cointzio	45.312	**25.181**	76.323	189.027	61.984	39.590
corrales	37.860	**30.205**	145.898	179.650	54.242	44.388
elfresno	49.125	38.378	42.472	187.003	**36.649**	56.528
lapalma	36.503	**31.136**	141.309	181.602	39.919	34.783
lapiedad	56.940	50.624	200.0	180.052	**31.002**	64.312
malpais	46.046	**28.856**	40.755	198.142	62.484	46.742
markazuza	49.003	**36.916**	113.731	154.063	62.080	51.388
melchorocampo	29.965	**27.885**	117.441	185.632	40.899	35.134
patzcuaro	50.734	**39.728**	84.692	187.654	91.997	83.065

Figure 5.2 presents a plot of the forecast of the winning model, NNDE, for the malpais time series. As discussed earlier in the chapter, the quality of the data is not as one could expect. Time Series contain noise, outliers, and missing data, not to count the fact that they are chaotic. Those characteristics make them extremely difficult to forecast. Nonetheless, from the figure, we can observe that the model closely predicts the cyclic behavior of the data, not being able to account for the noise included in the validation set, nor the outliers.

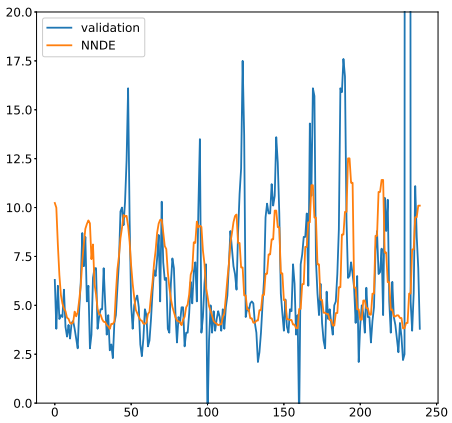


Figure 5.2: NNDE Forecast for the Malpais Validation Set

NNDE produces the best scores in most stations, which indicates that, for these time series, it is well suited to forecast many samples in advance, compared to the other forecasters (at least for this particular forecasting task).

5.4 Chapter Conclusions

The set of techniques included in this comparison were tested against twenty wind speed time series. Ten of them were obtained from Russian weather stations, sampled at 8-hour intervals and expressed as integers. This truncation can be seen as an introduction of noise to the original information. The other ten time series come from Mexican weather stations, sampled at hourly intervals, using one decimal digit. These time series had many missing data. The Maximum Lyapunov exponents were computed for each time series, showing that the time series are chaotic. In summary, the data we used for the comparison is chaotic, it contains noise, and has many missing data, therefore those time series are hard to predict in the long term.

The forecasting task was to predict 10 days, which represent 80 measurements for the Russian and 240 for the Mexican time series. We predicted those ten days in two different scenarios: One-Step Ahead and One-Day Ahead.

The results of the performance comparison show that NNDE outperforms the other methods in a vast majority of cases for OSA forecasting and in more than half in ODA forecasting. It is well known that the results of a performance comparison depends heavily on the error criterion used to measure performance of the forecasting techniques. In this case, we used MSE and SMAPE, and NNDE won in both cases, which places this technique as the most suitable for the forecasting tasks used in this comparison.

Chapter 6

Case Study: Solar Radiance and Temperature Forecasting

The global concern to reduce the emission of greenhouse gases and global warning, has encouraged the rapid growth in the integration of renewable energy sources in power grids. In addition to this, the incorporation of technological advances in different fields, including power electronics, measurement, control, and communications, among others, has taken the systems towards what is known as Smart Grids. The aim is to have more reliable and efficient electrical networks that contribute to energy sustainability, which implies important challenges and new paradigms in the planning and operation of electricity networks [91; 83; 88]. In this context, solar energy is essential, since there is a global increase in this type of clean generation, with a higher level of penetration in electricity grids, which entail different problems, due to the variability of solar energy [52; 73; 76; 62].

The variability in the energy coming from photo-voltaic systems can cause problems of different nature, such as stability, reliability, cost of energy, electric power balance, reactive power compensation, frequency response, energy management, resilience, among others. This variability also presents implications in energy markets [88; 73; 76; 62].

Solar photo-voltaic systems transform solar energy into electric power. The amount of energy a solar plant produces depends mainly on solar irradiance[1] and temperature. How a photo-voltaic system converts the energy

[1]Solar irradiance is the power per unit area received from the Sun in the form of electromagnetic radiation in the wavelength range of the measuring instrument. The solar irradiance integrated over time is called solar irradiation, insolation, or solar exposure.

conveyed from the sun into electrical energy depends on the solar panel characteristics. The first two variables depend on the weather and are constantly changing; the cell features can be considered fixed in the energy production process.

The main income to the photo-voltaic generation process depend on the weather conditions, which cannot be controlled. That is, we cannot predetermine how much energy a solar plant will produce in a given period of time. Under those conditions, solar plants are delivering (highly) variable amounts of energy to the electric network they are connected to. The network operators must have an estimate of the electric power delivered by a solar plant, in order to compensate, using other means of energy plants, when the solar plant will not be able to keep up with its expected production. Thus, it is necessary to forecast (i.e., to produce an estimate of the future) the amount of energy a photo-voltaic plant will produce in a given period of time.

The solar resource forecast, as previously mentioned, can be applied to address various problems faced by system operators, the supply companies, or developers of new projects. So several participants of the electric sector, can benefit from solar forecasting [88; 73].

In this chapter we compare alternative methods to forecast solar irradiance and temperature using data captured by a meteorological station, located at the Campus of the Universidad Michoacana de San Nicolás de Hidalgo, in the city of Morelia, Mexico. These alternative techniques are based on sliding window algorithms, such as nearest neighbors and artificial neural networks. Also, to improve this comparison, forecasts were made by using sARIMA (Seasonal ARIMA) and Long Short-Term Memory (LSTM) Artificial Neural Networks.

6.1 Related Work

Due to its relevance, solar prediction is a field that has been studied extensively in recent years, using different techniques, such as physical methods [12], statistical models (for example ARMA, ARIMA, ARMAX, FARIMA, etc. - [88; 75; 12; 86; 42; 50]), artificial intelligence techniques (ANN and deep learning, among others - [88; 75; 42; 50; 26; 79; 95]), as well as hybrid techniques (ARMA with ANN, auto-regressive ANN, ARMA and Time

Source: https://en.wikipedia.org/wiki/Solar_irradiance

Delay ANN (TDNN) - [91; 83; 88; 73]), seasonal auto-regressive integrated moving-average method (SARIMA - [50; 79; 18]), to mention a few.

The two main variables used to characterize the amount of energy produced by a photo-voltaic system are the solar irradiance over the panel and the atmospheric temperature. If both the irradiance and temperature can be accurately forecasted, then it is possible to calculate the output of the photo-voltaic panel precisely [59]. In this chapter, we will focus on the irradiance and temperature forecasting aspect.

In [56] a Multiplicative Auto-regressive Mean Average (ARMA) model was presented to generate an instant value of global irradiance. The data set used for the creation of this model corresponds to global irradiance measurements at 5-minute sampling periods, obtained with a radio-metric station located in the city of Cordoba, Spain. Their work showed that the agreement between the model output and the validation-data output is 65% of the complete series.

[47] proposed a method to predict the solar irradiance up to three days ahead, by using data from the European Centre for Medium-Range Weather Forecasts. In their work, they report that one day-ahead irradiance forecasts for single stations in Germany show an rRMSE[2] of 36%. For regional forecasts, accuracy increases with the size of the region. For the complete area of Germany, the rRMSE amounts to 13%.

In [19] a detailed analysis of different methods of solar irradiance is presented in a way that exposes the pros and cons of the different methods. This work first details the statistical methods and the ones based on satellite images of nebulous masses, then it focuses on a panorama of the methods based on artificial intelligence techniques.

In [89] a methodology of irradiance forecast using artificial neural networks (ANN) is presented. The ANN is trained with a vector of measurements, reconstructed with statistical techniques. The structure of this model is determined using the Levenberg-Marquardt algorithm for training the ANN, using cross validation. This work showed that the results obtained with their method, compared to a ANN's trained only with historical measurements were better given variable meteorological conditions.

---

[2]relative Root Mean Squared Error. Is expressed as $\text{rRMSE} = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}}{\sum_{i=1}^{n} y_i} \times 100$, where y_i is the i-th actual value, $\hat{y}_i$ is the i-th predicted value, and n is the length of the set.

[93] emphasize on the importance of the prediction of the incident solar radiation over the place where the photo-voltaic generation system is to be installed. They present a review of the main forecasting approaches using ANN techniques.

In [9] an ANN is proposed to determine the operation temperature of a photo-voltaic panel, based on measurements of air temperature and solar irradiance. That data was used to train an ANN. A study was made of the output power and electrical efficiency, based on the operational temperature of the panel.

In order to test the proposed forecasters, two time series and two sampling periods were used. The data of these time series was collected from the Universidad Michoacana de San Nicolás de Hidalgo (Mexico) central campus meteorological station, covering almost two years of measurements.

The first time series is the atmospheric temperature, measured in Celsius degrees, the second time series is the index of solar irradiance in watts per squared meter, both sampled in 10 minute and 60 minute resolutions. The names given to each time series are TEMP10 (temperature sampled at 10 minutes), TEMP60 (temperature sampled at 60 minutes), RAD10 (irradiance sampled at 10 minutes), and RAD60 (irradiance sampled at 60 minutes).

To train the forecasters, the last 10 days of the time series were used as validation set, while the rest of the series was considered the training set. Both 10-minute and one-hour sampling modes use the same days to train and validate the results.

6.2 Data Analysis

This section shows a succinct study of the time series used in this case of study. We will analyze and interpret the Auto-correlation plots of the time series in search of seasonality. We will also obtain and discuss the Maximum Lyapunov Exponents of the data.

6.2.1 Dickey-Fuller Tests

Table 6.1 shows the results of the Dickey-Fuller test of the time series.

Table 6.1: Dickey-Fuller Test Results of Temperature and Irradiance Time Series

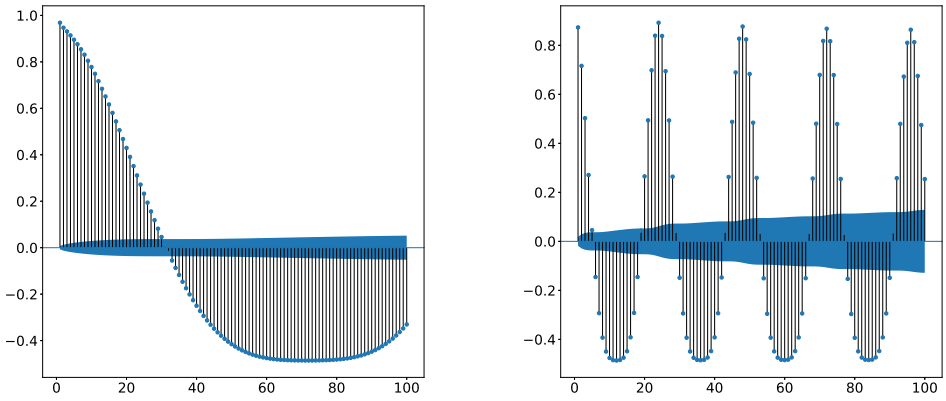
Time Series	ADF Statistic	p-value
RAD10	-47.420527	0.000000
RAD60	-8.797312	0.000000
TEMP10	-30.085949	0.000000
TEMP60	-5.469532	0.000002

The critical values for the ADF statistic are: 1%: -3.430, 5%: -2.862, and 10%: -2.567. The Dickey-Fuller test shows that every time series is stationary.

6.2.2 Auto-Correlation Analysis of the Data Sets

To identify possible periodicity present in the data, we obtained the Auto-Correlation plots of the time series.

Figure 6.1 shows the ACF for the RAD10 and RAD60 time series.



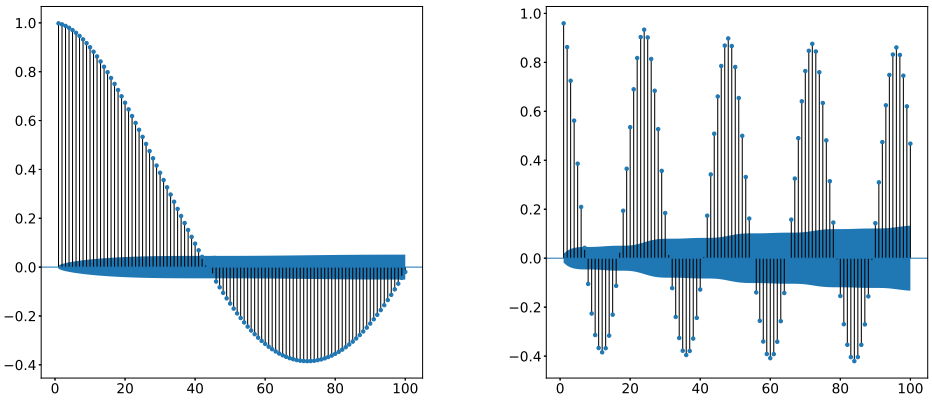
(a) ACF for RAD10 (b) ACF for RAD60

Figure 6.1: ACF's for the RAD10 and RAD60 Time Series

Figures 6.1a and 6.1b show that the auto-correlation values exceed the 5% threshold, which indicate that the null hypothesis that there is no correlation

for each time lag where it is exceeded can be rejected. For both time series there exist daily periodicity since the light coming from the sun slowly reaches a peak when it is at the zenith of the sensor and decreases towards the night.

Figure 6.2 shows the ACF's for the TEMP10 and TEMP60. A strong auto-correlation can be seen for the time lags corresponding to the daily measurements, indicating daily periodicity.



(a) ACF for TEMP10 (b) ACF for TEMP60

Figure 6.2: ACF's for the TEMP 10-min and TEMP 60-min Time Series

6.2.3 Maximum Lyapunov Exponent Calculation

The MLE's (in the range of $m \in [2, 10]$) of the time series used in this study are shown in Table 6.2.

Table 6.2: Lyapunov Exponents of the Data Sets

Time Series	**Minimum Lyapunov Exponent**	**Maximum Lyapunov Exponent**	**Average Lyapunov Exponent**
RAD10	-0.00292	0.02491	0.01197
TEMP10	$-\infty$	0.08731	N/A
RAD60	-0.00054	0.02606	0.01071
TEMP60	$-\infty$	0.00175	N/A

Table 6.2 shows that the temperature time series have negative MLE's, which is a sign of non chaotic behavior. Meanwhile, the irradiance time series show a negative MLE for one dimension but an overall positive average MLE for the rest of the tested dimensions. These results tells us that both kinds of time series contain cyclical behavior (which is possible to see visually inspecting the time series). However, having a positive exponent means that their predictability is low.

6.3 Experiments and Results

Two forecasting schemes were used. The One Step Ahead scheme, where the forecaster generates only one prediction at a time, repeated for the specified number days/samples. The other scheme used in this comparison is One Day Ahead, where the forecaster predicts one day of samples at a time in an iterative manner.

6.3.1 Experiment Settings

The forecasting task was defined to predict the last 10 days of data by using both OSA and ODA forecasting schemes. Each day consists of 144 or 24 samples (depending if it is 10-minute sampling or 60-minute sampling, respectively). In total 1440 and 240 forecasts were made for both samplings. The forecasters used in these experiments are the following:

sARIMA - The implementation of ARIMA in R [65] was used. The parameters $(P, D, Q)_s$ were obtained by using the function `auto.arima` and forcing the use of sARIMA by adding a seasonality of 144 and 24 for the 10 minute and 60 minute sampling time series, respectively.

LSTM - The neural network built consisted of a layer of 5 recurrent neurons with m inputs (this m is the same used by the NN method) and a dense layer with one output. The network used adam optimization and 4 recurrent iterations.

NN - For this method τ is determined by the mutual information method , m by doing a grid search $m \in [1, 100]$ and ϵ is obtained by measuring the number of neighbors present for the evaluation vector at the start

of the first day to forecast, if not enough neighbors are inside the radius of ϵ (for these experiments at least 50 neighbors are needed), ϵ is incremented in the form $\epsilon \leftarrow \epsilon \times 1.2$ with an starting value of 0.001.

NNDE - The parameters for DE were 30 iterations, population size of 30, the individuals could take values from $m \in [1, 100]$, $\tau \in [1, 100]$, $\epsilon \in [0, max(TS) - min(TS)]$ and recombination probability of 0.75. This was done in 30 independent executions.

Naïve - For the OSA forecasting scheme we included the Naïve method, which consists of making the forecast the last known value, this is $\hat{y}_{t+1} = y_t$.

The parameters obtained for the time series to be used by each forecaster in the OSA scheme are presented in Table 6.3.

Table 6.3: OSA Forecasters Parameters

Time Series	LSTM $[m, \tau]$	sARIMA $(p, d, q)_s$	NN $[m, \tau, \epsilon]$	NNDE $[m, \tau, \epsilon]$
RAD10	[2, 1]	$(4, 1, 4)_{144}$	[2, 1, 0.001]	[8, 20, 32.55681043]
TEMP10	[7, 1]	$(2, 1, 2)_{144}$	[7, 1, 0.590668229154]	[62, 30, 0.7012365]
RAD60	[4, 12]	$(5, 1, 2)_{24}$	[4, 12, 67.6170172238]	[64, 62, 953.59475495]
TEMP60	[5, 1]	$(3, 1, 5)_{24}$	[5, 1, 1.02067469998]	[41, 27, 1.77551006]

For the ODA scheme the parameters used are shown in Table 6.4.

Table 6.4: ODA Forecaster Parameters

Time Series	LSTM $[m, \tau]$	sARIMA $(p, d, q)_s$	NN $[m, \tau, \epsilon]$	NNDE $[m, \tau, \epsilon]$
RAD10	[14, 1]	$(4, 1, 4)_{144}$	[14, 1, 13.104630936]	[36, 18, 596.76738043]
TEMP10	[3, 1]	$(2, 1, 2)_{144}$	[3, 1, 0.114475459973]	[45, 42, 0.60916369]
RAD60	[6, 12]	$(5, 1, 2)_{24}$	[6, 12, 168.252776298]	[62, 62, 961.06028958]
TEMP60	[14, 1]	$(3, 1, 5)_{24}$	[14, 1, 10.92052578]	[41, 27, 1.77551006]

The forecasters models parameters were deployed to produce the forecasting results presented in the following subsection. In order to validate and compare the forecasts made by the forecasting methods, we used the Symmetric Mean Average Percentage Error (SMAPE).

6.3.2 Results

This subsection presents the results obtained by the different forecasting models when applied to the solar irradiance and atmospheric temperature time series. In those tables, the best forecasting results are highlighted in bold face. Every result is the SMAPE score of the forecasters for the indicated time series and sampling.

Table 6.5 shows the results for the One Step Ahead scheme with both 10 minute and 60 minute sampling for the Radiation and Temperature time series. From this table we can appreciate that, for this forecasting scheme, NNDE obtains the best results, with mixed competitors in each time series.

Table 6.5: SMAPE error for each time series in the OSA forecasting scheme

Series	LSTM	sARIMA	NN	NNDE	Naïve
RAD10	111.9842	110.4775	100.1628	**32.7693**	33.1607
TEMP10	1.3426	0.7418	0.428	**0.0**	0.9291
RAD60	142.7518	127.8605	102.372	**50.1201**	87.2675
TEMP60	3.4489	2.8702	2.7941	**0.0**	4.4906

It is possible to see that there is a significant gap in the errors when forecasting radiation and temperature. This is expected since the solar radiation time series are harder to predict (as demonstrated with their Lyapunov Exponents), in contrast with the temperature which is very regular. This fact is exploited by NNDE making perfect forecasts for these particular time series.

One thing that can be observed from these results is that the score for all forecasters is affected when using the 60 minute sampling. This behavior can be attributed to the reduced resolution of the data, which contributed to decrease the accuracy of the forecasters.

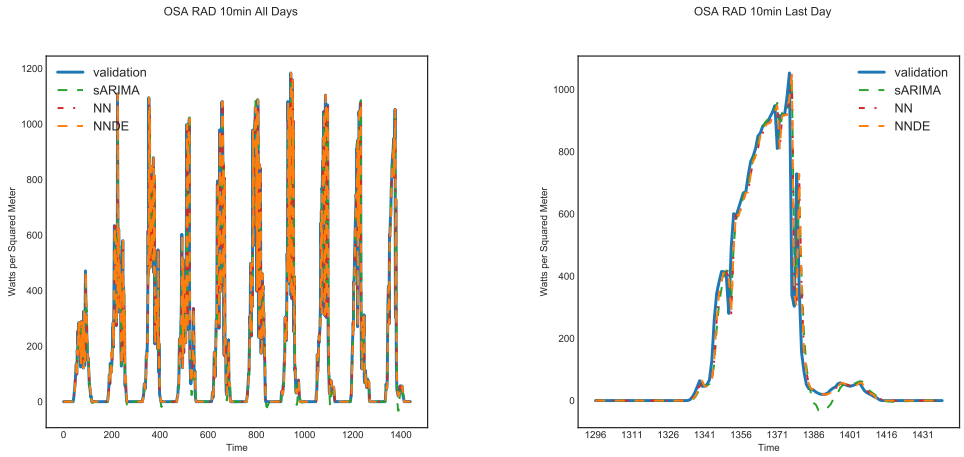
In the following figures only the best three forecasters are shown. This was done to avoid information overload on the figures and better exemplify the behavior of the forecasters.

Also, for the TEMP time series, NNDE is excluded from the figures, since it completely overlaps the validation set (it has zero error).

Figure 6.3 shows the OSA forecasts of the Radiation time series with 10-minute sampling, Sub-figure 6.3a shows the 10-day forecast and Sub-figure 6.3b shows the forecasts for the last day.

One behavior that can be seen from the figure is that the three best forecasters for this time Series (sARIMA, NN, and NNDE) follow very closely

the time series. However, on a closer inspection in Sub-figure 6.3b, this is the effect of naïve forecasts. Although is not quite clear, NNDE tends to reach the peaks more often than NN, which helps it to attain lower error.



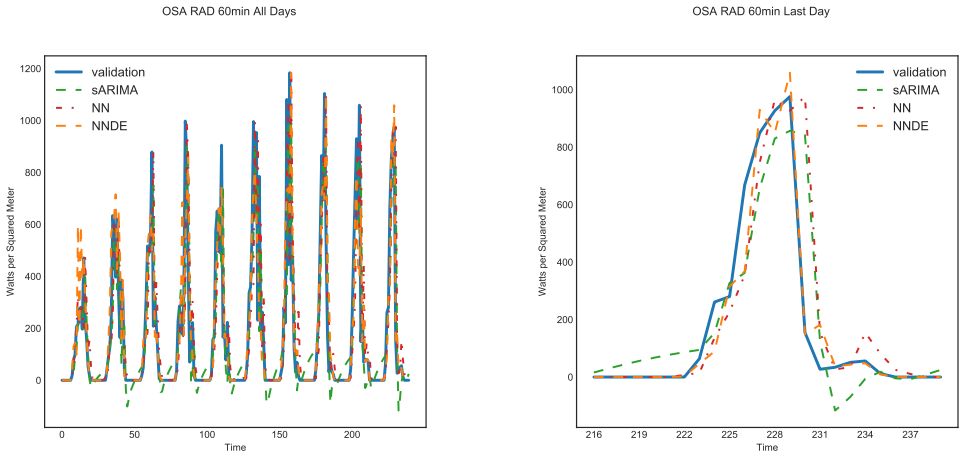
(a) All Days (b) Last Day

Figure 6.3: OSA Forecasts for the Radiation Time Series (10-minute sampling)

Figure 6.4 shows the OSA forecasts of the Radiation time series at 60-minute sampling. In this case the third best forecaster is sARIMA. An interesting behavior that can be seen is that sARIMA gives forecasts below 0, which physically is not possible. Also, both sARIMA and NN shows a lagged effect to the right around the 231 time tick (Sub-figure 6.4b). This tell us that both forecasters are reacting slowly to the change of value in the time series. With NNDE, is possible to see that it also has problems to follow the time series, but it both reacts faster than the other forecasters to the change of the curve of the series and, for this time series, it doesn't make naïve forecasts.

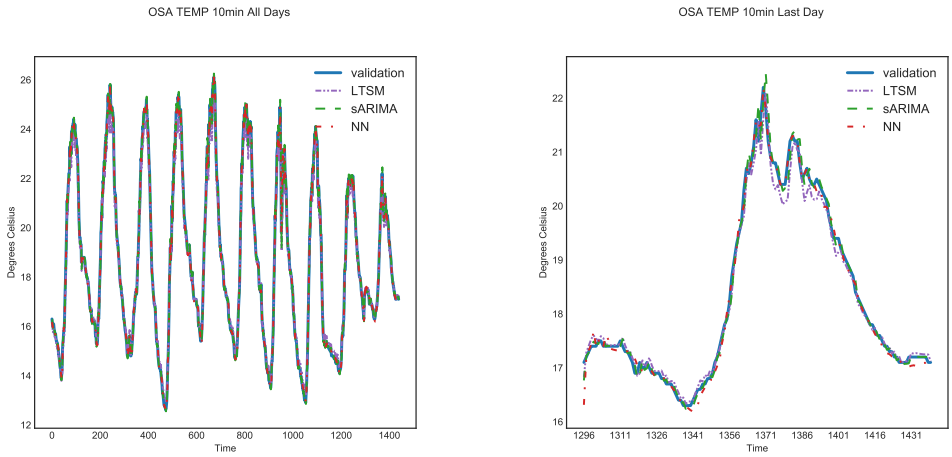
Figure 6.5 shows the OSA forecasts made for the Temperature time series with 10-minute sampling. It is possible to see that the forecasters follow more closely this time series. As the MLE table suggests, the TEMP time series should be easier to predict, which is corroborated by the forecasts.

Figure 6.6 shows the OSA forecasts of the temperature time series at 60-minute sampling. In this figure a couple of behaviors are noticeable. The



(a) All Days (b) Last Day

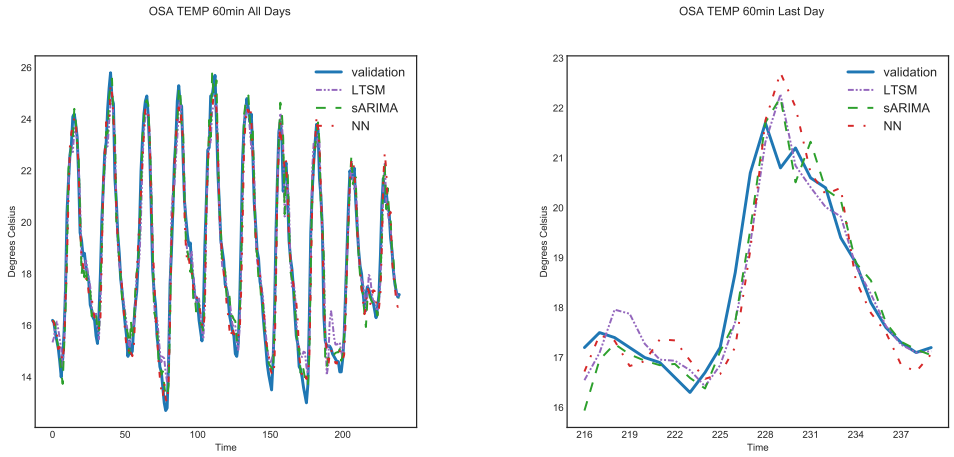
Figure 6.4: OSA Forecasts for the Radiation Time Series (60-minute sampling)



(a) All Days (b) Last Day

Figure 6.5: OSA Forecasts for the Temperature Time Series (10-minute sampling)

first one is that the forecasters tend to barely reach the valleys of the time series (Sub-figure 6.6a). The second one is that around the 228th and 231th



(a) All Days (b) Last Day

Figure 6.6: OSA Forecasts for the Temperature Time Series (60-minute sampling)

observations (Sub-figure 6.6b) their forecasts miss entirely the time series. Those behaviors are considered to be caused by the lack of intermediate observations, which could be missing valuable information. As mentioned before, this is reflected as a worsened score for every forecaster in the 60-minute sampling.

Table 6.6 shows the results for the One Day Ahead scheme, again with both 10 and 60 minute samplings for the Radiation and Temperature time series.

Table 6.6: SMAPE error for each time series in the ODA forecasting scheme

Series	LSTM	sARIMA	NN	NNDE
RAD10	108.9407	**63.3465**	126.3053	72.4807
TEMP10	1.0597	12.718	7.1361	**0.0**
RAD60	142.5605	59.6017	94.0312	**51.9376**
TEMP60	3.6307	12.7245	3.8223	**0.0**

In this scheme NNDE still obtains the best results except for the 10-minute Radiation time series where it obtains the second best result.

In ODA is possible to see a significant increase in the score of every

forecaster, with the exceptions of the LSTM forecaster and the temperature scores of NNDE. This behavior is expected, since in this scheme the forecasters use forecasted values to produce the following predictions, which introduces cumulative error [78].

However, for the temperature time series, NNDE is immune to this cumulative error, at least for this forecasting horizon. This is is considered to be caused by the nature of the time series, both because of its daily and annually seasonality, and because the data is represented by an integer with only one decimal digit. Both situations helps NNDE to make accurate forecasts.

Figure 6.7 shows the forecasts made by the three forecasters with the best scores for the radiation time series in the ODA scheme, with 10-minute and 60-minute sampling, respectively.

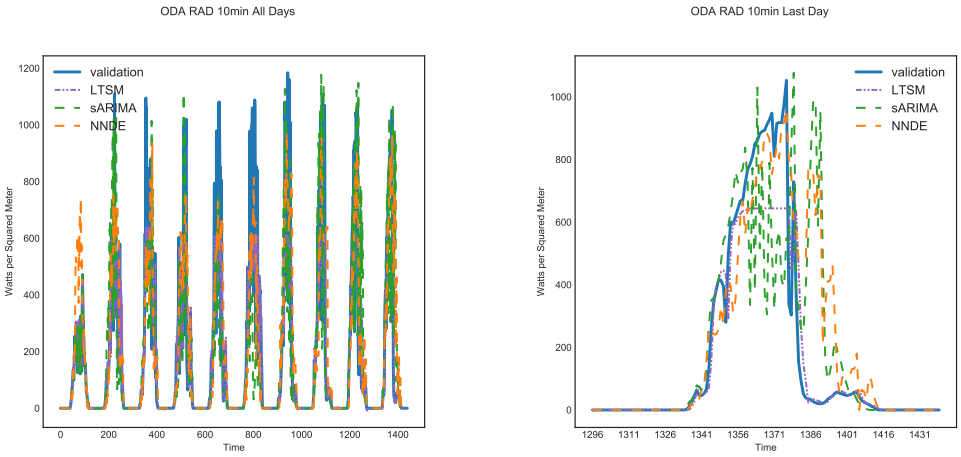
This case brings some interesting issues caused mostly by the same behavior. NN simply forecasts the mean value for that particular time tick, which makes its forecasts look like a more elongated and shorter curve (this behavior is clearly seen in Sub-figure 6.7b). LSTM is unable to forecasts above 600 W/m^2 in high irradiation days, which it can be attributed to a behavior like the one exposed by NN, where LTSM and NN forecasts the mean of the time series for that time tick. In the case of NNDE apparently follows correctly the start of the irradiance curve for a day, but as shown in Sub-figure 6.7b the descent of the curve is completely wrong. These are signs of a dominant past behavior that these forecasters try to replicate, but it is not present in this part of the time series.

Figure 6.8 shows the forecasts of the Radiation time series at 60-minute sampling. This figure is very similar to Figure 6.7 and most of the behaviors seen before replicate.

Figures 6.9 shows the forecasts for the TEMP time series at 10-minute sampling. For these forecasts, sARIMA tries to follow the series but the forecasts degenerate around the 600th and completely misses the time series by the 1000th sample (Sub-figure 6.9a). This behavior indicates that sARIMA cannot work with long forecasting horizons.

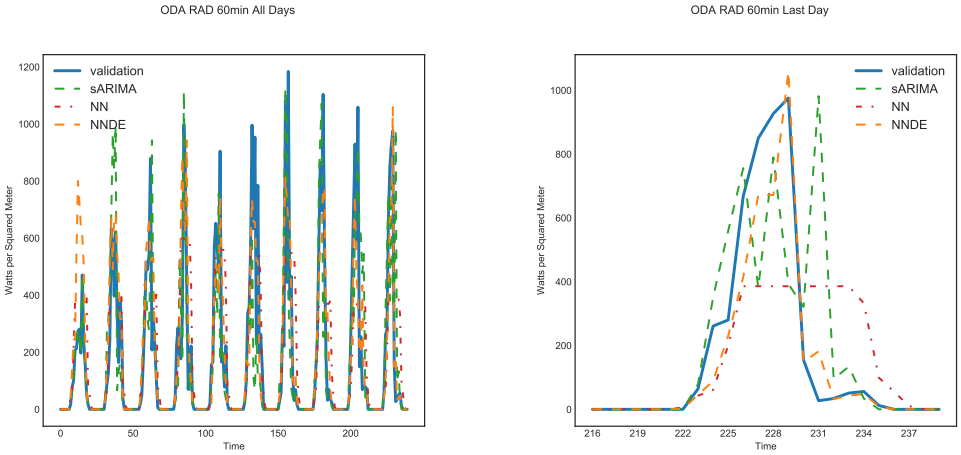
Figure 6.10 show similarities with figures 6.9, where NN simply reaches the mean high and low temperatures with some variations. Again, sARIMA stops following the time series around the 100th sample (Sub-figure 6.10a) and misses the time series by the 160th sample. LTSM shows great performance but some bad forecasts around the 180th sample (Sub-figure 6.10a), this behavior of LTSM can be attributed to the low number of samples.

From the results it is possible to say that NNDE obtains the best forecasts



(a) All Days (b) Last Day

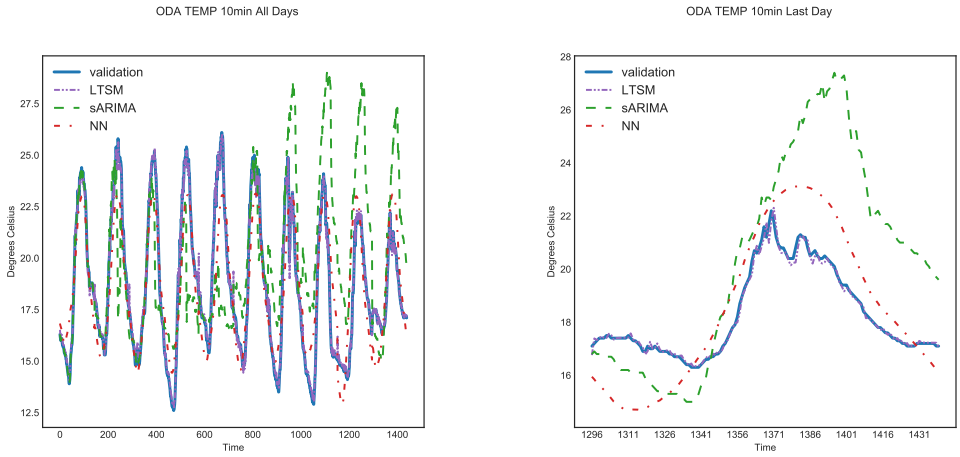
Figure 6.7: ODA Forecasts for the Radiation Time Series (10-minute sampling)



(a) All Days (b) Last Day

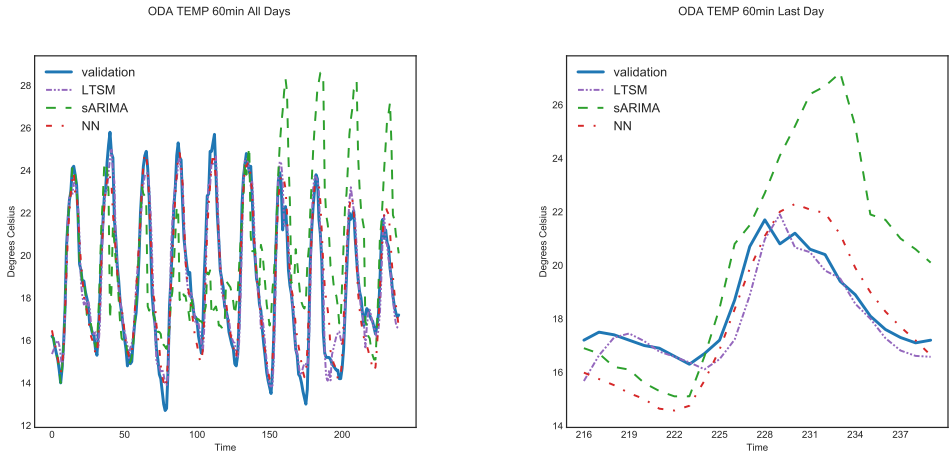
Figure 6.8: ODA Forecasts for the Radiation Time Series (60-minute sampling)

scores followed by sARIMA and the LSTM networks and that NNDE is particularly good at forecasting temperature.



(a) All Days (b) Last Day

Figure 6.9: ODA Forecasts for the Temperature Time Series (10-minute sampling)



(a) All Days (b) Last Day

Figure 6.10: ODA Forecasts for the Temperature Time Series (60-minute sampling)

6.4 Chapter Conclusions

The results obtained from this type of forecast are, to say the least, contrasting. The temperature time series, at least for the place where the readings were taken, is easy to forecast. The MLE of the temperature time series implied that this time series contain much less uncertainty and it was verified by the error obtained by most of the forecasters. However, the irradiance time series demonstrated to be very hard to forecast, with very high error on every forecaster and many of them unable to beat the naive forecast. Still, NNDE was able to obtain the least error for the irradiance time series and perfect scores on temperature.

Unfortunately, at the time of writing this thesis, we do not have access to more temperature measurements at other places, but we consider necessary to test NNDE against more adverse weather measurements.

Solar irradiance has been touted as difficult to forecast and it was clearly seen in this comparison. We consider that a forecasting scheme directed to forecast solar irradiance based purely on historical data, will not be entirely successful unless it is supported with additional data, such as atmospheric pressure and humidity of the location. Under this premise, a possible course of action would be the integration of these additional data into the delay vectors, expanding the notion of phase space reconstruction where not only the observed data is used to recreate the phase space, but also the control variables of the underlying dynamical system.

Chapter 7

Case Study: Electrical Demand Forecasting

Energy operators require accurate predictions of the behavior of the load in order to prevent any lack of energy. In open electrical markets, precise predictions of the load also are desirable since they allow to establish the prices of energy beforehand. Although electrical energy consumption tends to be regular, this is, increases its consumption in the morning, decreases slightly past noon, and increases abruptly at night[1]. Industrial activity, sports and social events, and most importantly lack of work activities, modify the behavior of the load.

Thanks to a partnership between the Centro Nacional de Control de Energía (National Center of Energy Control – CENACE) and the UMSNH in conjunction with the Secretaría de Energía (SENER), we were able to conduct tests and experiments over the CENACE data.

This chapter, as in previous case studies, shows a statistical analysis of the data, examines the chaotic properties of the time series, and shows a comparison of different forecasters compared to NNDE.

7.1 Related Work

Since electrical consumption is critical in today's society, being capable of deducting in a precise manner the required demand for each hour of the day is more than desirable for the operators of the electrical network.

[1]This behavior makes it to be known as "duck curve"

Electrical demand behaves regularly, but high demand activities alter the curve significantly. These alterations appear as high frequency peaks (this is, high demand that only appears for a time instant in the time series).

For forecasting purposes, this analysis is considered short term load forecasting, since the range of forecasts is in the order of 15 minutes sampling to forecast 24 hours of demand.

There exist many implementations of ANN's for the task of forecasting electrical demand at various levels (this is, from buildings to entire electrical grids).

[27] made an ANN to forecasts the demand in buildings that used as inputs the actual and forecasted temperature values, the electrical load, the hour and the day, achieving high precision in their forecasts.

In an attempt to characterize the demand by activity, [82] proposed a method that separates the load by type (residential, commercial, entertainment, public lightning, industrial, and non-industrial), then by using model with a Self Organizing Map (SOM), they produced monthly forecasts for each activity group, with a combined MAPE of 1.52%.

Continuing with the characterization of the demand, [38] proposed a solution to forecast 48 hours of electrical demand. By using clustering, they identified 13 different patterns, then they produced 13 different SVM models that used as inputs the last 48 samples of the load, the average temperature, the forecasted temperature, and the day of the week. With their method they achieved an average MAPE error of 3%.

[21] proposed a method that identified two types of days (regular and anomalous), then by using a SOM gating network identified the type of day to forecast. Then, depending on the type of day, 24 SVM's (one for each hour) would forecast the day. The SOM network takes as input the maximum load of the previous day, the average maximal daily load of the last week, the forecasted maximum temperature of the next day, the average maximum temperature of previous two days, the forecasted maximum humidity of the next day, the forecasted average wind speed of the next day, the temperature sensitivity coefficient, a weekend indicator and a holiday indicator. On the other hand, the SVM's of regular days uses as input the hourly load and the hourly temperature of the last 168 observations of each variable in intervals of 24 hours. For the anomalous days models the SVM's use the hourly load in lags of 24 hours of the last 5 days, the hourly load of the previous Saturday, the hourly load of the previous Sunday and the hourly temperature of the last 168 observations in intervals of 24 hours. The resulting model obtained

an average MAPE of 2.3.

7.2 Data Analysis

In this section we will provide a brief analysis of the time series provided. We will focus on the seasonality, stationarity and chaotic components of the time series that compose the SIN (from spanish *Sistema Eléctrico Nacional* – National Electrical System).

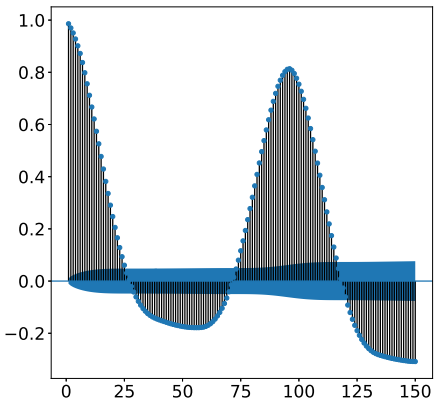
The SIN is divided in seven regions: Central (CEL), Noreste (NES), Noroeste (NOR), Norte (NTE), Occidental (OCC), Oriental (ORI), and Peninsular (PEN).

The data comprises 13 months of data measures in a one-minute resolution. The data was sub-sampled to 15 minutes resolution since the forecasting task requires this resolution.

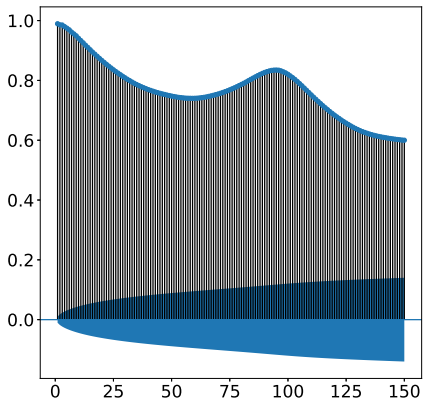
As in this thesis, the seasonality is measured with the Auto Correlation plot of the time series, the stationarity is calculated with the Dickey-Fuller test, and the chaotic component is represented with the Maximum Lyapunov Exponent calculated numerically.

7.2.1 Auto-Correlation Analysis of the Data Sets

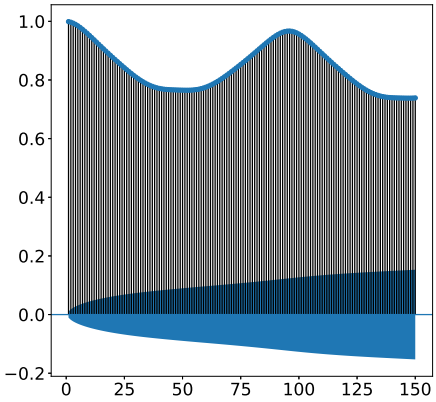
We obtained the ACF plots of all of the regions that compose the SIN. Figure 7.1 shows the ACF plot of four of the regions. In all of the plots a strong daily seasonal component is present. Is possible to see that the auto-correlation values exceeds the 5% threshold, which indicate that the null hypothesis that there is no correlation for each time lag where it is exceeded can be rejected.



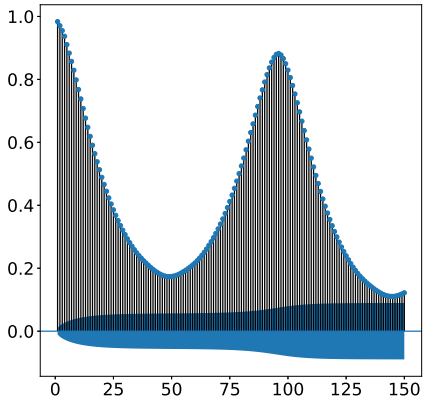
(a) ACF for CEL



(b) ACF for NES



(c) ACF for NOR



(d) ACF for ORI

Figure 7.1: ACF of Four Regions of the SIN

Examining the plot of the whole time series (Figure 7.2) is possible to appreciate a clear annual change of demand. However, because of the amount of data available, it was not possible to make the plots of the annual ACF.

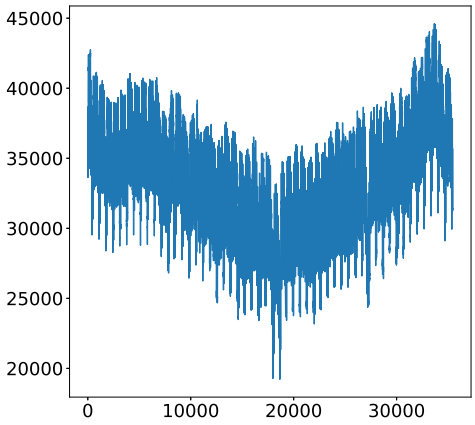


Figure 7.2: SIN Time Series

7.2.2 Dickey-Fuller Test of the Data Sets

To check if the time series are stationary or not, the Dickey-Fuller test was used. Table 7.1 shows the ADF statistic and p-value of each time series.

Table 7.1: Dickey-Fuller Test Results of Wind Speed Time Series

Time Series	**ADF Statistic**	**p-value**
CEL	-24.430219	0.000000
NES	-7.562009	0.000000
NOR	-5.516391	0.000002
NTE	-6.467922	0.000000
OCC	-14.851454	0.000000
ORI	-14.828665	0.000000
PEN	-11.073954	0.000000
SIN	-11.762601	0.000000

The critical values for the ADF statistic are: 1%: -3.430, 5%: -2.862, and 10%: -2.567. The Dickey-Fuller test shows that every time series is stationary. This results can be deceiving, since it is known that the demand of electrical energy is always increasing. The explanation for these results is that the data set is relatively short (it is composed of one year of observations) and

the global trend of the time series is not perceptible. Nonetheless, for the experiments the time series will be treated as stationary.

7.2.3 Maximum Lyapunov Exponent Analysis of the Data Sets

Although there are not any analysis that have tested the electrical load chaos levels, in most of the literature state that, while regular in its shape, the behavior of the series alternate depending of many factors. This alternation can be considered as chaos. It is also important to note that, industrial activity do not follow any pattern of use, which introduces to the time series abrupt changes that can be considered as noise. Table 7.2 shows the calculated Lyapunov Exponents in the range of $m \in [2, 10]$.

Table 7.2: Lyapunov Exponents of the Data Sets

Time Series	Minimum Lyapunov Exponent	Maximum Lyapunov Exponent	Average Lyapunov Exponent
CEL	0.0158	0.0460	0.0278
NES	0.0093	0.0492	0.0185
NOR	0.0048	0.0336	0.0131
NTE	0.0065	0.0552	0.0180
OCC	0.0160	0.0858	0.0373
ORI	0.0198	0.0610	0.0340
PEN	0.0102	0.0795	0.0303
SIN	0.0131	0.0831	0.0334

Table 7.2 shows that every time series contains positive MLE's. These MLE's confirm that the electrical load do contain alternating behavior.

7.3 Experiments and Results

This case study was treated differently than the other case studies done in this thesis. Since the variable of interest is the net demand of the SIN and this series is the sum of all of the regions that compose the system, the

problem was divided in forecasting each region independently, and the sum of the forecasts is the resulting prediction of the system.

Also, the intended use of the forecasting system is to produce forecasts 30 minutes ahead, instead of for the next time instant as in the previous cases. This changes the forecasting horizon of the forecasting task from one to two steps in the future. Furthermore, CENACE requires the instant forecasting error not exceed 500MW.

This section shows a comparison of the forecasting error obtained by three techniques: NNDE, MLP, and DA-RNN (Dual-Stage Attention-Based Recurrent Neural Network) [64].

7.3.1 Experiment Settings

One change that was introduced in this case of study is how the delay vectors are constructed. Instead of only using the samples provided by the time series, control variables are added to the vectors. Each vector S_t includes the day of the week and the hour (in minutes from the start of the day) as control dimensions. In addition to the time series $\mathbb{S}$, a sequence with the date and hour when the sample was taken was used. The resulting vector is shown in (7.1).

$$S_t = [\text{DoW}(d_{t-(m-1)\tau}), \text{MoD}(d_{t-(m-1)\tau}), s_{t-(m-1)\tau}, \ldots, s_{t-\tau}, s_t] \tag{7.1}$$

$d_{t-(m-1)\tau}$ is the date and hour of the first element of the delay vector, $\text{DoW}(\cdot)$ returns the day of the week (as a real number in the range $[0.0, 0.8571]$[2] starting on Monday) that corresponds to the date $d_{t-(m-1)\tau}$, and $\text{MoD}(\cdot)$ returns the minute of the day (represented as a real number in the range $[0.0, 0.9298]$[3]) that corresponds to the hour at which the data was sampled $d_{t-(m-1)\tau}$. This change was introduced since it allows the forecasters to take into account the influence of external factors [21].

The forecasting task is to forecast the SIN net demand from the 00:00 hours of June 24th 2018 every 15 minutes 30 minutes ahead, for a total of 92 forecasts. The forecasters and some of their settings are the following:

[2]The days are represented as a number from 0 to 6 and then divided by 7.

[3]Instead of representing the hour as HH:MM, the hour is converted to minutes from 0 to 1,339 and then divided by 1,440.

NNDE - The parameters for DE were 30 iterations, population size of 30, the individuals could take values from $m \in [1, 100]$, $\tau \in [1, 100]$, $\epsilon \in [0, max(TS) - min(TS)]$ and recombination probability of 0.75. This optimization process was done in 30 independent executions.

MLP - The topology of the designed MLP consists of 96 input neurons, two hidden layers of 100 neurons each and, a single output neuron.

DA-RNN - The learning rate and number of epochs were obtained by performing DA-RNN on a Bayesian optimizer over 30 independent executions, the batch size was set to 4.

7.3.2 Results

The parameters used by NNDE are shown in Table 7.3. MLP used the same parameters for every time series ($[96, 1]$). DA-RNN used a matrix of 10 observations of each time series to train and produce its forecasts.

Table 7.3: Forecasters Parameters

Time Series	NNDE $[m, \tau, \epsilon]$
CEL	[6, 4, 0.34100608]
NES	[8, 1, 0.14140621]
NOR	[2, 46, 0.05696183]
NTE	[2, 4, 0.05038196]
OCC	[13, 31, 0.34437311]
ORI	[2, 6, 0.06026154]
PEN	[1, 22, 0.02691927]

For this case study the metric to compare the forecaster is MAE. This metric was selected since the error is represented in the same units as the time series, which was also a petition made by CENACE.

Table 7.4 shows the forecasting errors for the SIN net demand of the forecasters. The table also included a column for the SICAPI, which is the forecaster used by CENACE. NNDE obtains the best forecasting error, outperforming all of the forecasters by a good margin.

Table 7.4: MAE error for SIN Time Series

Series	NNDE	MLP	DA-RNN	SICAPI
SIN	**213.3**	232.4	240.5	223.6

The following figures show the forecasts made by each forecaster compared to the validation set.

Figure 7.3 shows the forecasts made by NNDE. NNDE shows a good approximation of the time series from 00:15 to 03:15 but misses a big drop that lasted 30 minutes. Also, it can be seen that from 09:30 to 18:15 the forecast follows the increasing trend but mostly bellow the actual value. This behavior tells us that NNDE was able to capture the global trend in the time series, but smooths the high frequency variations which may not be desirable. From 19:45 21.15 the forecast lags behind the actual curve, specially at the 20:30 and 20:45. This tells us that most of the days observed by NNDE did not had this kind behavior, since the curve that NNDE produces is less abrupt than the actual one.

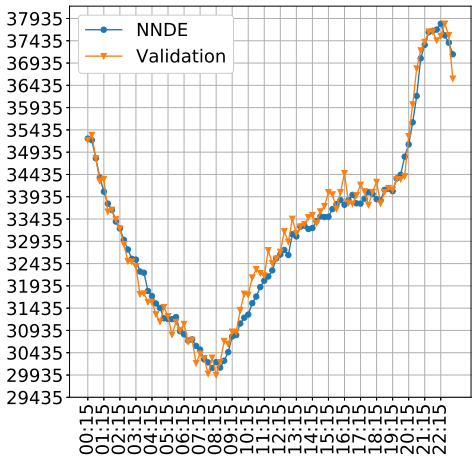


Figure 7.3: NNDE Forecasts vs Validation

Figure 7.4 shows the forecasts made by MLP. This forecaster obtained a very good approximation of the time series for the range between 00:15 up to 06:45, but misses the trend of the high frequency changes from 07:00 to 08:15. From 09:30 to 19:00 MLP as NNDE follows the increasing trend tightly in comparison with NNDE, but as with MLP, it misses the high frequency changes present in the validation set. However, and what penalizes the most the score obtained by MLP, are the forecasts made from 19:30 to 21:45. The trend captured by MLP does not correspond to the actual values of the time

series. This fact gives weight to the assumption that the curve of this day is atypical and was under-represented in the history of the time series.

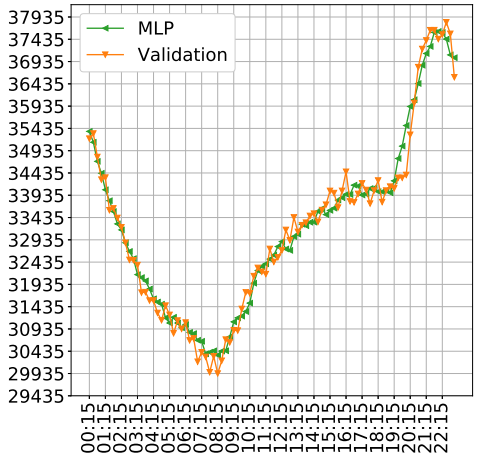


Figure 7.4: MLP Forecasts vs Validation

Figure 7.5 shows the forecasts made by DA-RNN. This method as the others follows the decreasing trend in the time series from 00:15 to 03:30, where as in the NNDE forecasts, DA-RNN does not follow the abrupt change in the time series. This behavior can be seen in all of the forecasters, with MLP being the less affected by it. Interestingly, from 08:00 to 15:00, DA-RNN tries to adjust to the high frequency changes, but misses the most from 15:15 to 17:15. This tells us that DA-RNN could capture better these high frequency changes but it can also be a sign of over-fitting (more data may help to determine better the causes of this behavior). From 20:00 to 21:15 DA-RNN, as the other forecasters, cannot follow entirely the increase of demand.

The forecast figures showed consistently that none of the forecasting methods can follow the increase of demand towards the end of the day. Considering the assumption that the ramp of demand was atypical, or the data does not contain enough samples of a day like this one, a simple analysis was proposed. The histogram of the errors should approximate a normal distribution if enough data samples were present in the data set.

Figure 7.6 shows the histogram of the forecasting errors made by NNDE and a normal fit from the data. The histogram does not resemble a normal

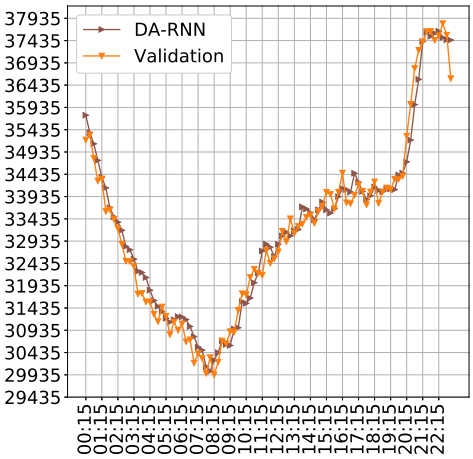


Figure 7.5: DA-RNN Forecasts vs Validation

distribution curve. Nonetheless, there is a decline of higher magnitude errors in the tails.

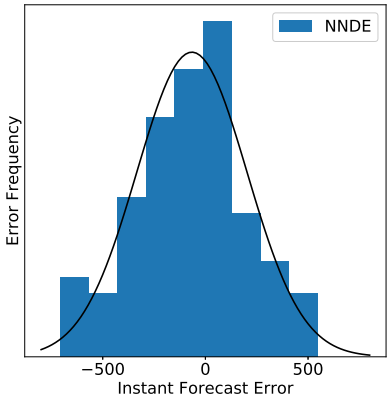


Figure 7.6: Histogram of NNDE Forecast Errors ($\mu = -65.64$, $\sigma = 268.33$)

Figure 7.7 shows the histogram of the errors obtained by MLP. In this case the histogram does not follow the normal curve either. As with NNDE, the errors are not centered at 0.

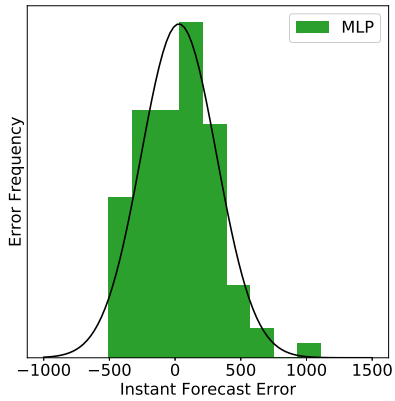


Figure 7.7: Histogram of MLP Forecast Errors ($\mu = 28.99$, $\sigma = 287.37$)

Figure 7.8 shows the histogram of the errors made by DA-RNN. Again, the errors do not follow the normal curve and are not centered at 0.

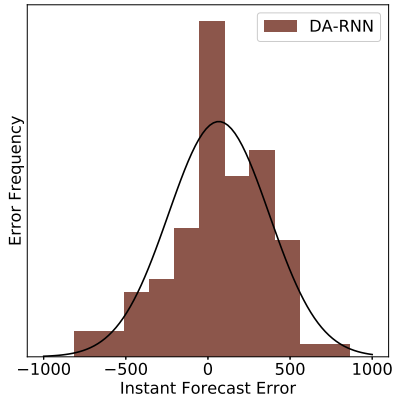


Figure 7.8: Histogram of DA-RNN Forecast Errors ($\mu = 66.19$, $\sigma = 306.58$)

From the three histogram plots we can conclude that there is bias in the mean of every forecaster, which can be a sign of that the forecasted day has atypical features and/or there still exist structure to be modeled in order to improve the forecasts.

7.4 Chapter Conclusions

With this case study it was possible to assess that pure phase space reconstruction can come short in forecasting time series. The inclusion of the day of the week and hour of the day dimensions helped to vastly improve the quality of NNDE and MLP forecasts. However, as the results showed, there is room for improvements. The literature exposed that many other factors can contribute to the electrical demand, specifically the atmospheric temperature, humidity, and wind speed; none of these variables were available for this study.

There is also the lack of more data samples. Although the results are acceptable, this is, NNDE was able to beat the forecasting method used by CENACE. The results show that the day analyzed can be considered atypical. This means that the usual pattern of demand is not present in the dataset. With the incorporation of more data samples, along with added control variables such as temperature, humidity and wind speed, we consider it is possible to improve the performance of NNDE.

Chapter 8

Discussion and Conclusions

This thesis proposed a new forecasting method for time series, which is a combination between a statistical method supported by the phase space reconstruction theory, and an stochastic optimization heuristic. The implementation of the method was intended to be simple so it could be deployed in a production environment.

This chapter discusses the characteristics of the data used to test NNDE, the results obtained in the different case studies presented as well as the quality of these results.

8.1 Chaos Identification

One definition that we have tried to establish clearly over this thesis, is that noise and chaos are completely different characteristics that a time series can possess. Because of this, a time series can be chaotic and noisy, and the correct identification of both is a problem on its own and it is not in the scope of this thesis. Nonetheless, the intention of this work was to develop a time series forecasting method that could deal with noisy and chaotic time series. As such, the first step was to correctly identify chaos.

The numerical calculation of the spectra of Lyapunov exponents has been explored from different angles and approaches. In this work we decided to use Rosenstein's Algorithm because it is easy to understand, this is, it provides a value that tells directly the amount of chaos present on the time series. It is also easily available, and it is robust in the presence of noise. Yet, it is far from perfect. In Chapter 4 we demonstrated that the approximated values

of the exponents tend to be a fraction of the actual exponent. Also, this method only determines the maximum Lyapunov Exponent, not the complete spectra, which allows only to imply that chaos is present. A more complete analysis, like the calculation of the fractal dimension of the time series, can not be done with this particular calculation of Lyapunov exponents.

However, the intention we had with the use of this numerical algorithm is to be able to assert that, if chaos is present for a certain embedding dimension, this quantity will represent how much the calculated MLE relates with the calculated value of another time series, instead of using a qualitative indicator. In this aspect, we consider that it served for this purpose well, since the more chaotic a time series is (in the form of the calculated MLE), the larger forecasting error.

8.2 Noise Identification

Throughout this work we did not explicitly introduced a metric or heuristic to determine the presence of noise. We only stated the existence of noise in the time series by observation. This was done on purpose, since at the time of writing this thesis we could not find a reliable method to verify the assumption of presence of noise. In a practical sense, noise must be completely identified in order to produce better forecasts. As we have proved in Chapter 4, forecaster accuracy is affected considerably by noise.

Moreover, we did not tried to manipulate the time series more than necessary in order to provide a common ground for all the forecasters, including NNDE. This resulted to be completely detrimental for the scores obtained by all of the forecasters (more of this in Section 8.3). In a production environment, many characteristics of the time series will be adapted to help the selected forecaster, such as provide stationarity, filter noise, scale the magnitudes of the series, remove outliers, etc. However, this was not the intention of this work, we preferred to assume its presence and ignore it.

8.3 Analysis of the Results

For all of the case studies displayed in this thesis we provided results tables for the forecasting error obtained by NNDE and the forecaster to which was compared to. These errors were calculated mostly as MAPE and SMAPE.

Since these metrics are independent of the units of the data, they can be used to compare dissimilar time series between them, and they give us a glimpse of how well the forecast was without knowing much about the magnitudes of the time series.

An observation that can be made from the results is that, the parameters obtained by NNDE are potentially valid only for the time instant that were produced. Since it is impossible to know the changes that a time series can suffer in the future, the parameters should be considered adequate only for an undetermined number of forecasts. A dynamic adjustment of the parameters is a possible future work that should be explored.

The values of the errors obtained by all of the forecasters when noise is present is, to say the least, worrying. The range of error is far from optimal for a critical operation such as electrical power generation by wind turbines or by photo-voltaic panels. In this sense, we consider that the results obtained in this thesis represent the actual state of the art of time series forecasting. This is, the data available contains many characteristics that contribute negatively to the forecasting tasks, making the forecasters perform badly. However, we consider that, while some results are far from optimal, NNDE is a step in the right direction since the forecasts obtained with it surpass (and in some cases vastly) the performance of other forecasters that are in use today.

In the end, any forecaster must be designed to be used as part of a decision making process. NNDE requires from 2-20 hours to calculate the best parameters (running 30 independent executions in parallel), while the process of making forecasts with the obtained parameters can be done in less than a few seconds. The computer server used to run the experiments is composed of 96 cores running at 2Ghz with 220Gb of RAM. NNDE used on average 30 cores and around 20Gb of RAM. While those values exceed the capabilities of a regular PC (at the moment of writing this), we consider that at most 20 hours of computer work are in the range of feasibility of use. With the use of newer technology based on GPU's (Graphic Processing Unit) we consider that the calculation of the distances between neighbors and the determination of the best set of parameters can be parallelized, resulting in a drastic reduction of computer work time.

8.4 Conclusions

NNDE accomplished the objectives described in Chapter 1. The forecasts obtained are competitive for chaotic time series in the presence of noise.

It is important to note that both the quality and quantity of the data varied considerably in each case study. However, we did not find any correlation between the quantity or the quality of the data and the forecasts obtained. Instead, what affects adversely all of the forecasters are the high frequency variations present in the data. The nature of these high frequency changes is a complete subject of discussion, since they can be considered noise or chaotic dynamics. In Chapter 4 we introduced noise to the time series and the scores that the forecasters obtained dropped considerably. In later chapters however, what can be considered noise could be part of the dynamic system.

We identified that the phase space reconstruction theory holds true for synthetic time series, where the number of variables that comprises the dynamical system is small. But, for dynamical systems with an unknown number of variables, forecasting processes that only observe one variable will produce poor forecasts. In this thesis we incorporated some control variables that improved the performance of NNDE (Chapter 7). We consider that the addition of more control variables that impact directly on the time series to forecast will help to reduce the forecasting error.

Overall, despite the difficulties present in each time series, NNDE was able to produce in most of the cases the best results.

8.5 Future Work

Besides of the possible improvements mentioned in Section 8.3 there are several routes of action that we consider will help to improve the performance of NNDE.

A simple feature is to weigh the neighbors. The idea behind is that the contribution of the farthest neighbors should be less than the contribution of the closer neighbors. One way to weight to neighbors is to sort them from the closest to the farthest and are multiplied by a decreasing factor as neighbors move away from the center of the neighborhood. Another way is to apply a Gaussian mask to the area of the neighborhood, where the neighbors are multiplied by a factor in function of their position in the area, which is the

elevation of the Gaussian mask.

One problem with Nearest Neighbors is that it is based on the assumption that vectors that are closer in phase space will have similar outcomes. However, the problem is that it is not possible to know if all of the vectors inside this neighborhood actually have the same target, since this space could be large enough to include vectors that have completely different outcomes. This problem is minimized by optimizing ϵ, but still can happen. One possible solution is to apply a clusterization process to each neighborhood to identify the "class" of the neighbors and use only the ones that share the same properties.

As we showed in Chapter 7, it is preferable to forecast components of a time series (if they exist) in order to produce forecasts of the time series of interest. With that idea in mind, a decomposition process (such as wavelet decomposition) could be used to decompose the time series in order to isolate problematic components (such as high frequency noise) and produce forecast over the components rather than the time series of interest.

Bibliography

[1] H. Abarbanel. *Analysis of observed chaotic data*. Springer Science & Business Media, 2012.

[2] G. Boeing. Visual analysis of nonlinear dynamical systems: chaos, fractals, self-similarity and the limits of prediction. *Systems*, 4(4):37, 2016.

[3] R. Boné and M. Crucianu. Multi-step-ahead prediction with neural networks: a review. *9emes rencontres internationales: Approches Connexionnistes en Sciences*, 2:97–106, 2002.

[4] G. E. Box, G. M. Jenkins, and G. C. Reinsel. *Time series analysis: forecasting and control*. John Wiley & Sons, 2011.

[5] L. Bramer. *Methods for modeling and forecasting wind characteristics*. PhD thesis, Iowa State University, The address of the publisher, 2013.

[6] D. S. Broomhead and D. Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[7] E. Cadenas and W. Rivera. Wind speed forecasting in three different regions of mexico, using a hybrid arima–ann model. *Renewable Energy*, 35(12):2732–2738, 2010.

[8] G. S. Camargo and N. G. Barriga. Preliminary identification study of the wind resource at the state of michoacán. In *2014 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–7, Nov 2014. doi: 10.1109/ROPEC.2014.7036351.

[9] İ. Ceylan, O. Erkaymaz, E. Gedik, and A. E. Gürel. The prediction of photovoltaic module temperature with artificial neural networks. *Case Studies in Thermal Engineering*, 3:11–20, 2014.

[10] P. Chen, H. Chen, and R. Ye. Chaotic wind speed series forecasting based on wavelet packet decomposition and support vector regression. In *IPEC, 2010 Conference Proceedings*, pages 256–261. IEEE, 2010.

[11] Z. Chen and Y. Yang. Assessing forecast accuracy measures. *Preprint Series*, 2010:2004–10, 2004.

[12] I. Colak, M. Yesilbudak, N. Genc, and R. Bayindir. Multi-period prediction of solar radiation using arma and arima models. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 1045–1049. IEEE, 2015.

[13] A. J. Conejo, M. A. Plazas, R. Espinola, and A. B. Molina. Day-ahead electricity price forecasting using the wavelet transform and arima models. *IEEE transactions on power systems*, 20(2):1035–1042, 2005.

[14] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo. Arima models to predict next-day electricity prices. *IEEE transactions on power systems*, 18(3):1014–1020, 2003.

[15] G. W. Crawford and M. C. Fratantoni. Assessing the forecasting performance of regime-switching, arima and garch models of house prices. *Real Estate Economics*, 31(2):223–243, 2003.

[16] C. Croonenbroeck and D. Ambach. A selection of time series models for short- to medium-term wind power forecasting. *Journal of Wind Engineering and Industrial Aerodynamics*, 136:201–210, 2015. ISSN 01676105. doi: 10.1016/j.jweia.2014.11.014.

[17] E. De La Vega, J. J. Flores, and M. Graff. k-nearest-neighbor by differential evolution for time series forecasting. In *Nature-Inspired Computation and Machine Learning*, pages 50–60. Springer, 2014.

[18] C. L. Dewangan, S. Singh, and S. Chakrabarti. Solar irradiance forecasting using wavelet neural network. In *Asia-Pacific Power and Energy Engineering Conference (APPEEC), 2017 IEEE PES*, pages 1–6. IEEE, 2017.

[19] M. Diagne, M. David, P. Lauret, J. Boland, and N. Schmutz. Review of solar irradiance forecasting methods and a proposition for small-scale

insular grids. *Renewable and Sustainable Energy Reviews*, 27:65–76, 2013.

[20] D. A. Dickey and W. A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366a):427–431, 1979. doi: 10.1080/01621459.1979.10482531. URL `https://doi.org/10.1080/01621459.1979.10482531`.

[21] S. Fan and L. Chen. Short-term load forecasting based on an adaptive hybrid method. *IEEE Transactions on Power Systems*, 21(1):392–401, 2006.

[22] J. D. Farmer and J. J. Sidorowich. Predicting chaotic time series. *Physical review letters*, 59(8):845, 1987.

[23] J. J. Flores, F. González-Santoyo, B. Flores, and R. Molina. Fuzzy NN time series forecasting. pages 167–179, 2015.

[24] J. J. Flores, J. R. C. González, R. L. Farias, and F. Calderon. Evolving nearest neighbor time series forecasters. *Soft Computing*, Sep 2017. ISSN 1433-7479. doi: 10.1007/s00500-017-2822-1. URL `https://doi.org/10.1007/s00500-017-2822-1`.

[25] R. Fried and A. C. George. *Exponential and Holt-Winters Smoothing*, pages 488–490. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2_244. URL `https://doi.org/10.1007/978-3-642-04898-2_244`.

[26] K. Gairaa, F. Chellali, S. Benkaciali, Y. Messlem, and K. Abdallah. Daily global solar radiation forecasting over a desert area using nar neural networks comparison with conventional methods. In *Renewable Energy Research and Applications (ICRERA), 2015 International Conference on*, pages 567–571. IEEE, 2015.

[27] P. A. Gonzalez and J. M. Zamarreno. Prediction of hourly energy consumption in buildings based on a feedback artificial neural network. *Energy and buildings*, 37(6):595–601, 2005.

[28] M. Graff, E. S. Tellez, S. Miranda-Jiménez, and H. J. Escalante. Evodag: A semantic genetic programming python library. In *Power, Electronics*

and Computing (ROPEC), 2016 IEEE International Autumn Meeting on, pages 1–6. IEEE, 2016.

[29] Z. Guo, D. Chi, J. Wu, and W. Zhang. A new wind speed forecasting strategy based on the chaotic time series modelling technique and the apriori algorithm. *Energy Conversion and Management*, 84:140–151, 2014.

[30] M. T. Hagan, H. B. Demuth, and M. H. Beale. *Neural network design*, volume 20. 1996.

[31] S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[32] M. Hénon. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50:69–77, feb 1976.

[33] D. L. Hitzl and F. Zele. An exploration of the hénon quadratic map. *Physica D: Nonlinear Phenomena*, 14(3):305–326, 1985.

[34] F. O. Hocaoğlu, M. Fidan, and Ömer N. Gerek. Mycielski approach for wind speed prediction. *Energy Conversion and Management*, 50 (6):1436 – 1443, 2009. ISSN 0196-8904. doi: https://doi.org/10.1016/j.enconman.2009.03.003. URL `http://www.sciencedirect.com/science/article/pii/S0196890409000788`.

[35] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[36] X. Hongfei and D. Tao. Chaotic prediction method of short-term wind speed. In *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE2011) November 19–20, 2011, Melbourne, Australia*, pages 479–487. Springer, 2012.

[37] R. Huffaker and M. Bittelli. A nonlinear dynamics approach for incorporating wind-speed patterns into wind-power project evaluation. *PloS one*, 10(1):e0115123, 2015.

[38] A. Jain and B. Satish. Clustering based short term load forecasting using support vector machines. In *2009 IEEE Bucharest PowerTech*, pages 1–8. IEEE, 2009.

[39] Y. Jiang, C. Xingying, Y. Kun, and L. Yingchen. Short-term wind power forecasting using hybrid method based on enhanced boosting algorithm. *Journal of Modern Power Systems and Clean Energy*, 5 (1):126–133, 2017. doi: 10.1007/s40565-015-0171-6. URL `https://doi.org/10.1007/s40565-015-0171-6`.

[40] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed 13/05/2019].

[41] H. Kantz and T. Schreiber. *Nonlinear time series analysis*. Cambridge university press, 2004.

[42] E. Kardakos, M. Alexiadis, S. Vagropoulos, C. Simoglou, P. Biskas, and A. Bakirtzis. Application of time series and artificial neural network models in short-term forecasting of pv power generation. In *Power Engineering Conference (UPEC), 2013 48th International Universities'*, pages 1–6. IEEE, 2013.

[43] R. G. Kavasseri and K. Seetharaman. Day-ahead wind speed forecasting using f-arima models. *Renewable Energy*, 34(5):1388–1393, 2009.

[44] H. Kumar and S. B. Patil. Estimation & forecasting of volatility using arima, arfima and neural network based techniques. In *Advance Computing Conference (IACC), 2015 IEEE International*, pages 992–997. IEEE, 2015.

[45] J. Lampinen and I. Zelinka. Mixed integer-discrete-continuous optimization by differential evolution. In *Proceedings of the 5th International Conference on Soft Computing*, pages 71–76, 1999.

[46] H. Leung, T. Lo, and S. Wang. Prediction of noisy chaotic time series using an optimal radial basis function neural network. *IEEE Transactions on Neural Networks*, 12(5):1163–1172, 2001.

[47] E. Lorenz, J. Hurka, D. Heinemann, and H. G. Beyer. Irradiance forecasting for the power prediction of grid-connected photovoltaic systems. *IEEE Journal of selected topics in applied earth observations and remote sensing*, 2(1):2–10, 2009.

[48] E. N. Lorenz. Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*, 20:130–148, mar 1963.

[49] H.-y. Luo, T.-q. Liu, and X.-y. Li. Chaotic forecasting method of short-term wind speed in wind farm [j]. *Power System Technology*, 9:019, 2009.

[50] J. Ma and X. Ma. State-of-the-art forecasting algorithms for microgrids. In *Automation and Computing (ICAC), 2017 23rd International Conference on*, pages 1–6. IEEE, 2017.

[51] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

[52] Y. S. Manjili, R. Vega, and M. M. Jamshidi. Data-analytic-based adaptive solar energy forecasting framework. *IEEE Systems Journal*, 12(1): 285–296, 2018.

[53] J. M. Matías and J. C. Reboredo. Forecasting performance of nonlinear models for intraday stock returns. *Journal of Forecasting*, 31(2):172–188, 2012. ISSN 1099-131X. doi: 10.1002/for.1218. URL `http://dx.doi.org/10.1002/for.1218`.

[54] R. M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560):459–467, 1976.

[55] D. C. Montgomery, C. L. Jennings, and M. Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[56] A. Moreno-Munoz, J. De La Rosa, R. Posadillo, and V. Pallares. Short term forecasting of solar radiation. In *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, pages 1537–1541. IEEE, 2008.

[57] I. Okumus and A. Dinler. Current status of wind energy forecasting and a hybrid method for hourly predictions. *Energy Conversion and Management*, 123:362 – 371, 2016. ISSN 0196-8904. doi: https://doi.org/10.1016/j.enconman.2016.06.053. URL `http://www.sciencedirect.com/science/article/pii/S0196890416305428`.

[58] E. Olmedo. Comparison of near neighbour and neural network in travel forecasting. *Journal of Forecasting*, 35(3):217–223, 2016. ISSN 1099-131X. doi: 10.1002/for.2370. URL `http://dx.doi.org/10.1002/for.2370`.

[59] C. R. Osterwald. Translation of device performance measurements to reference conditions. *Solar cells*, 18(3-4):269–279, 1986.

[60] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a time series. *Physical review letters*, 45(9):712, 1980.

[61] A. K. Palit and D. Popovic. *Computational Intelligence In Time Series Forecasting*. Springer, 2005.

[62] I. Pavičić, A. Župan, and S. Cazin. Challenges of the transmission system operator to managing distributed generation and consumption. In *Smart Grid Metrology (SmaGriMet), 2018 First International Colloquium on*, pages 1–5. IEEE, 2018.

[63] K. Price, R. M. Storn, and J. A. Lampinen. *Differential evolution: a practical approach to global optimization*. Springer, 2006.

[64] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*, 2017.

[65] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

[66] H. R. Rangel, V. Puig, R. L. Farias, and J. J. Flores. Short-term demand forecast using a bank of neural network models trained using genetic algorithms for the optimal management of drinking water networks. *Journal of Hydroinformatics*, 19(1):1–16, 2017.

[67] V. Razuvaev, E. Apasova, R. Martuganov, and D. Kaiser. Six-and three-hourly meteorological observations from 223 ussr stations. Technical report, Oak Ridge National Lab., TN (United States), 1995.

[68] H. Rodriguez, J. J. Flores, V. Puig, L. Morales, A. Guerra, and F. Calderon. Wind speed time series reconstruction using a hybrid

neural genetic approach. *IOP Conference Series: Earth and Environmental Science*, 93(1):012020, 2017. URL http://stacks.iop.org/1755-1315/93/i=1/a=012020.

[69] M. T. Rosenstein, J. J. Collins, and C. J. De Luca. A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 65(1-2):117–134, 1993.

[70] O. Rossler. An equation for hyperchaos. *Physics Letters A*, 71(2):155–157, 1979.

[71] P. Rothman. *Nonlinear time series analysis of economic and financial data*. Springer Science & Business Media, 2012.

[72] G. Santamaría-Bonfil, A. Reyes-Ballesteros, and C. Gershenson. Wind speed forecasting for wind farms: A method based on support vector regression. *Renewable Energy*, 85(C):790–809, 2016. URL https://EconPapers.repec.org/RePEc:eee:renene:v:85:y:2016:i:c:p:790-809.

[73] J. L. Sawin, F. Sverrisson, K. Seyboth, R. Adib, H. E. Murdock, C. Lins, I. Edwards, M. Hullin, L. H. Nguyen, S. S. Prillianto, et al. Renewables 2017 global status report. 2013.

[74] D. Schroeder. *Astronomical Optics*. Electronics & Electrical. Academic Press, 2000. ISBN 9780126298109. URL https://books.google.com.mx/books?id=v7E25646wz0C.

[75] A. Shakya, S. Michael, C. Saunders, D. Armstrong, P. Pandey, S. Chalise, and R. Tonkoski. Solar irradiance forecasting in remote microgrids using markov switching model. *IEEE Transactions on Sustainable Energy*, 8(3):895–905, 2017.

[76] H. Sheng, J. Xiao, Y. Cheng, Q. Ni, and S. Wang. Short-term solar power forecasting based on weighted gaussian process regression. *IEEE Transactions on Industrial Electronics*, 65(1):300–308, 2018.

[77] C. Skittides and W.-G. Früh. Wind forecasting using principal component analysis. *Renewable Energy*, 69:365 – 374, 2014. ISSN 0960-1481. doi: https://doi.org/10.1016/j.renene.2014.03.068. URL http://www.sciencedirect.com/science/article/pii/S0960148114002432.

[78] L. A. Sorjamaa Antti. Time series prediction using dirrec strategy. In *ESANN'2006 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium)*, pages 143–148. Springer, 2006.

[79] M. C. Sorkun, C. Paoli, and Ö. D. Incel. Time series forecasting on solar irradiation using deep learning. In *Electrical and Electronics Engineering (ELECO), 2017 10th International Conference on*, pages 151–155. IEEE, 2017.

[80] J. Sprott. Simplifications of the lorenz attractor. *Nonlinear dynamics, psychology, and life sciences*, 13(3):271, 2009.

[81] J. C. Sprott and J. C. Sprott. *Chaos and time-series analysis*, volume 69. Citeseer, 2003.

[82] D. Srinivasan. Energy demand prediction using gmdh networks. *Neurocomputing*, 72(1-3):625–629, 2008.

[83] E. B. Ssekulima, M. B. Anwar, A. Al Hinai, and M. S. El Moursi. Wind speed and solar irradiance forecasting techniques for enhanced renewable energy integration with the grid: a review. *IET Renewable Power Generation*, 10(7):885–989, 2016.

[84] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[85] D. Tao and X. Hongfei. Chaotic time series prediction based on radial basis function network. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, volume 1, pages 595–599. IEEE, 2007.

[86] S. I. Vagropoulos, G. Chouliaras, E. G. Kardakos, C. K. Simoglou, and A. G. Bakirtzis. Comparison of sarimax, sarima, modified sarima and ann-based models for short-term pv generation forecasting. In *Energy Conference (ENERGYCON), 2016 IEEE International*, pages 1–6. IEEE, 2016.

[87] M. van Gerven and S. Bohte. *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018.

[88] C. Wan, J. Zhao, Y. Song, Z. Xu, J. Lin, and Z. Hu. Photovoltaic and solar power forecasting for smart grid energy management. *CSEE Journal of Power and Energy Systems*, 1(4):38–46, 2015.

[89] F. Wang, Z. Mi, S. Su, and H. Zhao. Short-term solar irradiance forecasting model based on artificial neural network using statistical feature parameters. *Energies*, 5(5):1355–1370, 2012.

[90] X. Wang, P. Guo, and X. Huang. A review of wind power forecasting models. In *Energy Procedia*, volume 12, pages 770–778, 2011. ISBN 1876-6102. doi: 10.1016/j.egypro.2011.10.103.

[91] Y. Wang, Y. Shen, S. Mao, G. Cao, and R. M. Nelms. Adaptive learning hybrid model for solar intensity forecasting. *IEEE Transactions on Industrial Informatics*, 2018.

[92] L. M. X. Xu. Rbf network-based chaotic time series prediction and its application in foreign exchange market. In *Proceedings of the international conference on intelligent systems and knowledge engineering (ISKE 2007)*, 2007.

[93] A. K. Yadav and S. Chandel. Solar radiation prediction using artificial neural network techniques: A review. *Renewable and Sustainable Energy Reviews*, 33:772–781, 2014.

[94] J.-S. Zhang and X.-C. Xiao. Predicting chaotic time series using recurrent neural network. *Chinese Physics Letters*, 17(2):88, 2000.

[95] Y. Zhang, M. Beaudin, H. Zareipour, and D. Wood. Forecasting solar photovoltaic power production at the aggregated system level. In *North American Power Symposium (NAPS), 2014*, pages 1–6. IEEE, 2014.