Adaptive Prediction of Targets in Image Sequences using Sparse Models

João Miguel Limpo de Lacerda Pestana Girão

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): João Manuel Lage de Miranda Lemos
Jorge Dos Santos Salvador Marques

Examination Committee

Chairperson:
Supervisor:
Members of the committee:

June 2019

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgements

First and foremost, I would like to thank both my supervisors, Prof. Dr. João Miranda Lemos and Prof. Dr. Jorge Salvador Marques for their assistance and guidance throughout the entire time spent on this thesis. The growth in experience and knowledge achieved during these last months I owe to them.

I would also like to thank INESC-ID for their financial support and for granting me access to their facilities and multiple resources, as well as my laboratory coworkers for their help and advice.

Thank you to all the colleagues I've met during these past six years for turning this journey into a delightful experience. From those, a most wholehearted thanks to those that have, in one way or another, molded me into a better version of myself. Notably to Luís Rei, Catarina Campos, Artur Alves, António Almeida, Gonçalo Duarte, Pedro Bhagubai and many others that will undoubtedly be forever in my memories.

A special thanks goes to my family, namely my father, my mother and my sister, for believing in me and shaping me to be the man I am today.

To my closest friends, whose friendships I cherish like no other, I address my ineffable joy of having you with me on this adventure we call life.

FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

This work counted with the support of a SPARSIS project research grant, funded by FCT under contract PTDC/EEIPRO/0426/2014.

Resumo

A predição de eventos futuros foi sempre vista como uma ferramenta útil na resolução de problemas de controlo. A posse de informação sobre os próximos estados do sistema a estudar fornece uma profunda compreensão do seu funcionamento. Este relatório explora as consequências e as razões por trás da adição de esparsidade na identificação dos modelos que regem o sistema e na predição dos seus resultados. Para testar a validez desta solução, uma comparação entre o desempenho de métodos que ativamente procuram respostas esparsas e outros que não o fazem é realizada. Com o propósito de tentar descrever o sistema com um modelo oculto de Markov, um método alternativo de identificação é proposto.

A estimativa dos valores dos resultados é feita através de uma abordagem linear. Para generalizar a predição de coordenadas de um alvo para grupos de alvos, um método que aplica a resolução da equação de Fokker-Planck no movimento dos pontos adjacentes aos objetos a seguir é construído. A escolha da representação baseada numa grelha de pontos provém de uma variedade de fatores, tais como a natureza discreta dos dados usados e a redução da carga de trabalho devido à simplificação do espaço de estados.

Este trabalho serve de base para futuros desenvolvimentos de métodos de predição mais rápidos e de confiança. Particularmente, este relatório elabora as vantagens e desvantagens da introdução de esparsidade como uma parte crítica do preditor. O resultado final é uma solução adequada para predições a curto prazo.

Palavras-chave: Controlo adapatativo, Esparsidade, Identificação de sistemas, Predição de séries temporais.

Abstract

The prediction of future events has always been regarded as a useful tool for solving control problems. Possessing knowledge over the forthcoming state of the studied system provides a deeper understanding of its behaviour. This report explores the addition of sparseness in the identification of the models generating the output and subsequent effects on the forecasting of said outcome. A comparison between sparsity aware and unaware methods is performed to test the validity of each solution. An alternative system identification method is proposed after pondering on the assumption of a Hidden Markov Model representing the system.

A linear approach is considered for the estimation of the output data values. In order to accommodate the prediction of several target coordinates as a group, an application of the Fokker-Planck equation to the motion of the objects is devised on a set of points surrounding the tracked objects. The choice of a grid-based representation is backed up by various factors such as the discrete nature of the data values in the preferred database and the reduction of the workload due to the simplification of state space.

This work provides a basis for the continued search of faster and reliable forecast methods. Particularly, the report elaborates on the advantages and disadvantages of introducing sparseness as a crucial part of the predictor. The end result is an adequate solution for short-term predictions.

Keywords: Adaptive control, Sparsity, System identification, Time-series forecasting.

Contents

List of Figures

List of Tables

Glossary

AR AutoRegressive.

ARMA AutoRegressive Moving Average.

HMM Hidden Markov Model.

LASSO Least Absolute Shrinkage and Selection Operator.

MA Moving Average.

RLS Recursive Least Squares.

RW-LASSO RLS-Weighted LASSO.

Chapter 1

Introduction

The aim of this work is the development of an algorithm for recursive target tracking using adaptive predictors with continuous estimation. The use of sparse techniques to improve the estimation of the motion of said targets by using parsimonious motion models will also be explored. This work has a close relation with real-life applications as it is inserted within the scope of project SPARSIS (`https://www.it.pt/Projects/Index/4440`).

The *Sparse Modeling and Estimation of Motion Fields* project, acronymed SPARSIS, is an FCT (Fundação para a Ciência e a Tecnologia) funded project whose purpose is, as the name suggests, the exploration of the usage of sparse techniques to improve the estimation of multiple motion fields as well as space-varying matrices (stochastic matrix) describing the switching process associated to the movement of a target. By the end of this dissertation, it is expected that at least one of the desired scientific outcomes of this project has been achieved, these being: the obtention of reliable estimates for the model parameters; to remove the overly smoothed character of the field estimates; the acceleration of the estimation procedure towards real time applications and extension of these techniques to multi-camera settings and the development of an adaptive algorithm for the online estimation of multiple motion fields able to update in a recursive way the number of fields and the field estimates whenever new information arrives.

1.1 Motivation and prior work

Nowadays, with the amount of sensors deployed all around the globe there is an excess of data available and a shortage of means to analyze and process the information in it contained. Take, for example, the area of video surveillance, where most of the data generated by the hundreds of cameras in public places is left unprocessed, and only a small portion is observed by humans. Therefore, there is the need for the implementation of a solution that estimates the normal behaviour and detects abnormal events, preferably, in real-time. The use of motion fields in the description of such actions or trajectories is considered, due to the usefulness that this method has proven to have in this subject. Furthermore, the methods associated to time series forecasting, that are usually applied in a different framework will also

be explored as a source of inspiration.

The ability to predict or estimate future values of a series is a feature highly sought by many, especially those working in areas like business, management and other economic related fields. Understanding the way a series varies and how it will behave in the long or short-term allows for a more accurate and less risky planification of sales strategies, contingency plans and other applications [1, 2]. From the usage of simpler methods such as the popular autoregressive integrated moving average model (ARIMA) or the Kalman-filter based solutions, to non-linear models reliant on neural-networks and even combinations of both linear and non-linear methods, there exists a notable number of approaches for very distinct forecasting problems [3, 4]. Over the years, various approaches have been proposed for the forecasting of data, be it either using just non-linear solutions [5, 6] or coupled with other linear methods [7].

The stochastic approach to the forecasting of time-series led to the formulation of autoregressive (AR) and moving average (MA) models which generated a considerable body of literature in the area of time series, dealing with parameter estimation, identification, model checking, and forecasting. Later, this knowledge was integrated by Box and Jenkins in their publication *Time Series Analysis: Forecasting and Control* [8]. The ARIMA models and their extensions were popularized due to the advent of the computer [9] and are still largely used these days as a solution for linear forecasting problems..

On the other hand, the relatively recent nonlinear methods that employ neural networks and other machine-learning methods are becoming more and more popular with the evolution of the computer capabilities. These metods are particularly useful for nonlinear processes that have an unknown functional relationship and, consequently, are difficult to fit.

Each of these methods possesses its advantages and disadvantages. As with many other situations, there is no optimal solution for the time-series prediction problem. The best option is usually the one that better performs according to the specific requirements of the issue at hand. Although management and economic applications are the most obvious motivations for the usage of forecasting, they are by far not the only attractive options, even more so because with the development of new technologies and methods, new fields of study surge and this menu of options is bound to grow. The main motivation behind this work is the application of this prediction in areas related to the motion of targets. More specifically, in this report the creation process of an algorithm capable of estimating and predicting the trajectories of singular and/or groups of targets is planned and described.

The field of motion estimation has been the subject of major attention, with various contributions and new applications surging over the past few years [10]. Some of these applications include weather phenomena tracking and prediction [11–13], the characterization of human movement using the dynamic model of the movement as a feature for gesture recognition [14] and person re-identification [15], public surveillance and anomaly detection [16, 17]. Motion processes associated with object trajectories are prone to changes due to their dependence on the objects surrounding environment. Although most of these changes originate from unpredictable or unknown sources, their influence in the variation of the motions is a major factor to be considered when studying or working with systems that describe these processes. In a dynamic system such as this one, the ability to adapt to the situation at hand is a key component that affects the performance of predictors. There have been some approaches that sought to

estimate the different motion fields active in target trajectories, either by clustering various trajectories according to certain patterns present in the data [18], taking into account the position of the targets, resulting in a space-dependent motion model [19] or by modelling the observed trajectories using random vector fields [20]. While these and other projects[1] have dealt with the offline estimation of motion fields, in this work an online algorithm that updates its results as new information becomes available is sought.

Lastly, in an attempt to enhance the estimation and identification of these motion models, sparsity is pursued, as it is preferable to simplify the solution instead of over-fitting it to the problem at hand. There is already a broad set of sparse algorithms to choose from, with many modifications of other estimation algorithms being studied [21–23], though in this report only one sparsity-aware algorithm will be tested. The choice was made after reviewing the results from [24].

1.2 Objectives

This report expands on time series forecasting directed mainly, but not only, towards the prediction of one or more targets future positions based on previous information obtained from video image sequences. More specifically, considering methods directly related to the usually called Box-Jenkins approach [8], an algorithm is proposed and implemented. It is expected that this algorithm be capable of identifying and selecting the appropriate models that make up the input data signals, estimate the parameters associated to said models and with that provide a reasonably accurate estimate of a fair amount of future data values. A crucial part of this algorithm takes its roots in the work done by Rui Brás on Adaptive Control of Sparse Models [24], with the intent of applying the knowledge gained from that report in more realistic scenarios. Granted that albeit an important aim of this report is the study of the impact of sparsity aware models in the forecast of subsequent values of a desired series of data, the contribution of both sparsity aware and unaware algorithms will be compared.

1.3 Report Outline

This report is composed of various chapters, starting with Chapter 1, that contains a brief introduction of the work done, as well as the motivation and objectives behind it. In Chapter 2 an explanation of the motion generation model is given and some examples are shown. Chapter 3 goes over the estimation of signals and estimation algorithms used. The prediction of signals is reviewed in Chapter 4. Some results are shown in Chapter 5 while conclusions are taken in Chapter 6.

[1] ARGUS project: `https://www.it.pt/Projects/Index/1593`

Chapter 2

Problem definition and solution architecture

This chapter delves deeper into the description of the issue tackled in this report, providing the reader with a detailed analysis of the various components that compose the problem, and their subsequent solutions. Due to the intricate nature of the subject, it is broken down into several simpler tasks that will then be approached and studied individually. As solutions for those issues are devised and proposed, an algorithm containing such methods is constructed. The layouts of both the problem and the solution models are shown below.

2.1 Objective

The main goal of the work presented in this report is the development of an algorithm capable of forecasting the values of a time dependent series of points in an interval of time instances that succeed the data known at the present epoch. To pursue such an aim, it is essential to have an understanding of the origin of the signal, be it either by knowing the underlying model that produces it or by employing a model whose output closely approximates the desired one. Since the plant of the actual system is inaccessible, a replacement is proposed to emulate the behavior of the model in the hopes that the forecasting of future values of the time series obtained via this method resembles the actual thing.

It is assumed that the signal originates from white noise, presumably Gaussian, that then passes through two blocks, each adding a different component to the outcome. The first block creates an output that corresponds to the local deviation from the expected, more general motion that is generated by the second block. The signal x, henceforth designated the motion of the target (time series), is the product of including a stationary signal y, based on an AR/ARMA model in the creation of a non-stationary signal, as depicted by the diagram presented in figure 2.1 below.

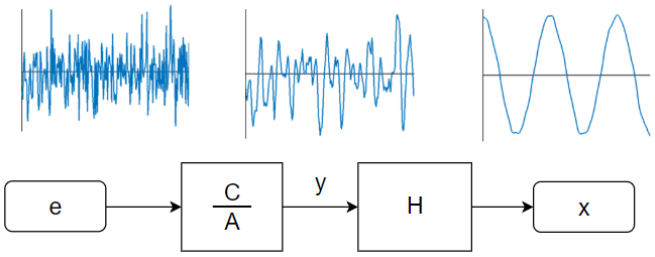


Figure 2.1: Block diagram of the motion generating system.

2.2 Motion Generation

The consecutive alterations in value of a sequence of points along a temporal axis produce an effect that, for an outside viewer, can be perceived as movement. In an attempt to describe this process of changing positions, the motion of an object is characterized based on the general evolution of these changes and the more detailed adjustments present in the series.

2.2.1 Stationary signal

The first step in generating motions consists in obtaining a stationary signal y with an AR/ARMA model

$$y(k) = \frac{C}{A}e(k), \tag{2.1}$$

where $e(k)$ is the error at the instance of time k with

$$E[e^2(k)] = \sigma_e^2. \tag{2.2}$$

This stationary signal, when receiving as input Gaussian white noise, results in a signal whose properties such as mean and variance are constant, hence the name. It is assumed that the polynomials of both parts of the fraction are monic and the system is causal and invertible, that is, the degree of the numerator is equal to that of the denominator. Under these assumptions, the general case comes as

$$\begin{cases} A(z) = z^n + a_1 z^{n-1} + ... + a_n, \\ C(z) = z^n + c_1 z^{n-1} + ... + c_n. \end{cases} \tag{2.3}$$

These expressions are, however, not useful in the context of prediction, and so, their reciprocal forms will be used instead. By multiplying each of them by q^{-n} one obtains the polynomials

$$\begin{cases} A^*(z) = 1 + a_1 z^{-1} + ... + a_n z^{-n}, \\ C^*(z) = 1 + c_1 z^{-1} + ... + c_n z^{-n}, \end{cases} \tag{2.4}$$

that relate the present to the past instead of the future to the present. From (2.1) it is possible to reach the expression of the next value of y by the multiplying both sides by A^* and isolating the current

instance of the output, as follows,

$$y(k)(1 + a_1 z^{-1} + ... + a_n z^{-n}) = e(k)(1 + c_1 z^{-1} + ... + c_n z^{-n}) \tag{2.5}$$

or

$$y(k) = -a_1 y(k-1) - ... - a_n y(k-n) + e(k) + c_1 e(k-1) + ... + c_n e(k-n) \tag{2.6}$$

Similarly, the formula for the output value m-steps ahead can be derived. The result is

$$y(k+m) = -a_1 y(k+m-1) - ... - a_n y(k+m-n) + e(k+m) + c_1 e(k+m-1) + ... + c_n e(k+m-n). \tag{2.7}$$

Further simplifications can be made by dropping the MA part of the model and assuming that only the error at the latest instance of time is relevant. Since all the terms of C except the first become zero, (2.7) cuts down to

$$y(k+m) = -a_1 y(k+m-1) - ... - a_n y(k+m-n) + e(k+m). \tag{2.8}$$

2.2.2 Non-stationary signal

Every motion can be described by a series of points that vary with the progression of a certain variable, usually time. This series can be represented by adding noise at each point of a predefined motion. The latter can be generated with the help of a single block that takes one of two forms. The first yields either straight lines or parts of parabolas and is called an integrator block. It simulates the behaviour of the function

$$H(z^{-1}) = \frac{1}{(1 - z^{-1})^N}, \tag{2.9}$$

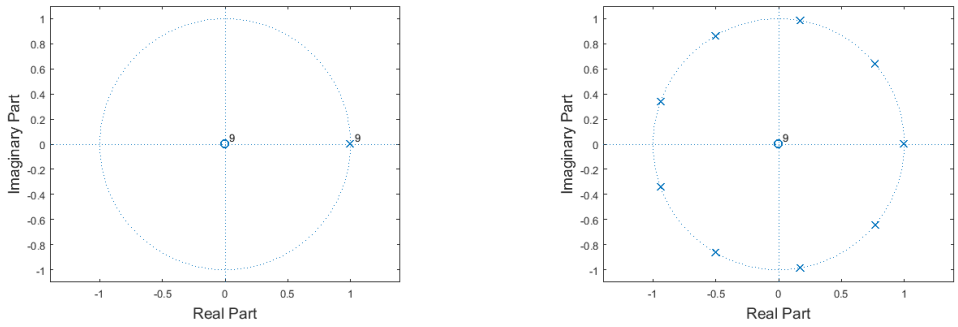
that represents an n-order integration of the input signal. The results of the second form vary in aspect, since the changes in period and initial conditions define the shape of the output signal. A block of this type is referred to as a periodic block and has the following expression

$$H(z^{-1}) = \frac{1}{1 - z^{-N}}, \tag{2.10}$$

where N is the period with which the movement repeats. The difference between an integrator block and a periodic block for a given value of N can be seen by examining the poles they exhibit in the z-plane, as shown in figure 2.2. While both these transfer functions have N zeros at the origin, the integration function concentrates all of its poles on the point $z = 1$. On the other hand the poles of the periodic function, whose positions are given by

$$\begin{aligned} z &= e^{-j\frac{2\pi k}{n}}, \\ k &= 0, 1, ..., n-1, \end{aligned} \tag{2.11}$$

are equally spaced around the unitary circumference.



(a) Poles and zeros of an integrator.

(b) Poles and zeros of a periodic block.

Figure 2.2: Z-plane representation of the transfer functions of an 9-order integrator block (a) and of a periodic block with period $n = 9$ (b).

To facilitate the distinction between these two blocks, forthwith an N-order integrator block will be addressed to as H_i^N, while a periodic block with the same order will come as H_p^N.

Integrator block

The output of an integrator block depends on the degree of the denominator and on the initial conditions, with the former defining the shape of the signal and the latter its position and growth. The equation for the output of one of these blocks H_i^N is

$$\begin{aligned} &y(k)(h_0 + h_1 + ... + h_N) = e(k), \\ &h_i = (-z)^{-i} p_i, \end{aligned} \tag{2.12}$$

with p_i being the ith entry of the row N of Pascal's triangle. For null inputs (noiseless outputs) and order 1 or 2 the resulting signal takes the form of a straight line passing on each initial condition. The oldest initial condition can be thought of as a starting point, since it refers to a constant term used to distinguish between different lines with the same order. Take, for example, the case of an order 2 integrator with initial conditions $y(1)$ and $y(2)$. The next term of the output of this system is given by

$$y(3) = 2y(2) - y(1),$$

or

$$y(3) = y(2) + \delta(y),$$

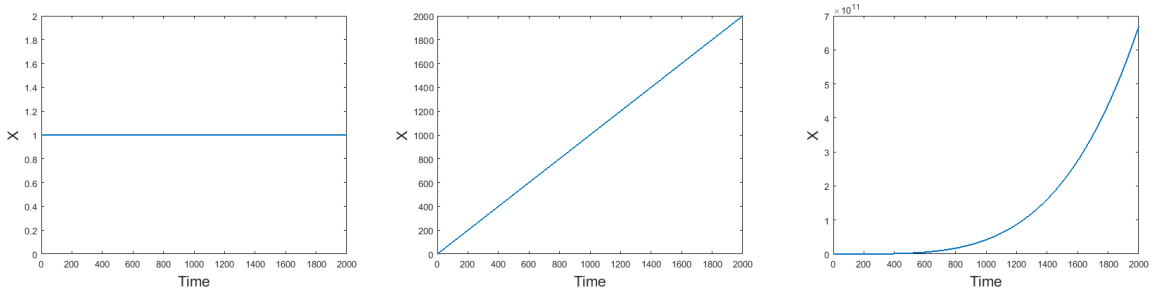
with $\delta(y)$ the difference between the initial conditions values. Analogously, the second term can be derived form the oldest initial condition by adding $\delta(y)$ to its value. Under these conditions the slope value can be computed through

$$\frac{dy(t)}{dt} = y(t) - y(t-1), \tag{2.13}$$

for H_i^2.

The value for the slope of H_i^1 is 0 since it originates a constant output. With higher order integrators ($N > 3$) the result is always the same, at least visually, as the only difference between two blocks with

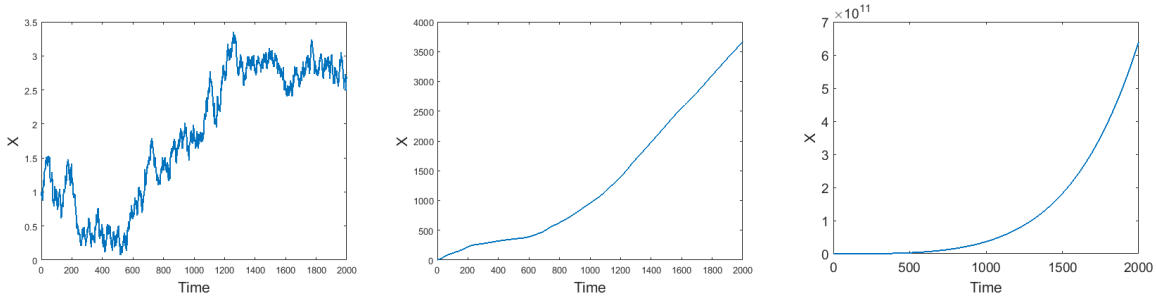
different integration order is the speed at which their value grows. To better illustrate this behavior some signals were generated, using different orders for the same model. The signals can be seen in figures 2.3 and 2.4.



(a) 1st order integration (b) 2nd order integration (c) 5th order integration

Figure 2.3: Integration of a noiseless vector of zeros with initial conditions [1] (a), [1, 1] (b) and [1,1,1,1,1] (c).

The results shown above are based on the assumption that the input signal is zero, however if some noise is added to the input (assumed Gaussian and white), the output will still be similar to its noiseless counterpart, figure 2.4. It is noticeable that the jitter from the noise is attenuated with every integration, becoming smooth after two integrations, with the quickly changing signal transforming into a slightly curved line. Furthermore, when the the signal stabilizes around a given value the output is similar to the noiseless input signal, as we can see in figure 2.4 (a) and (b) from the instant 1200 onwards.

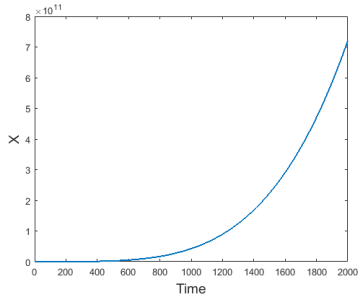


(a) 1st order integration with noise (b) 2nd order integration with noise (c) 5th order integration with noise

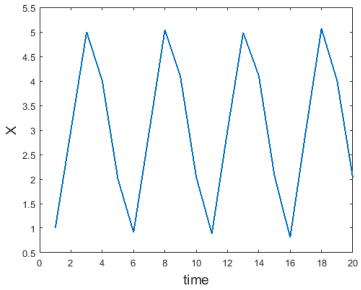
Figure 2.4: Integration of normally distributed noise with $\mu = 0$, $\sigma^2 = 0.05$ and initial conditions [1] (a), [1, 1] (b) and [1,1,1,1,1] (c).

Periodic block

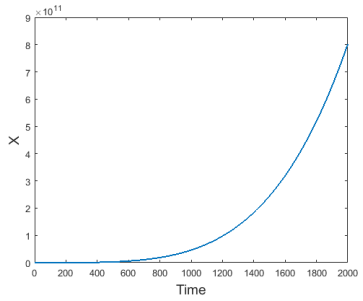
On the other hand, H_p^T generates a signal that repeats after some time T and whose output depends on all the initial conditions until that point. This behaviour gives us a plethora of different signals that can be generated by varying the periodicity and/or the values of the initial conditions, contrary to H_i^N which generates only two types of signals. While the change in initial conditions dictates the growth of the output in an integrator block, in a periodic block such changes have a whole other level of impact since they can alter the signal completely. In figure 2.5 one can see just how distinctly these blocks react to the same modification in initial conditions. The only noticeable difference in the output of the integration blocks is the value it reaches after 2000 periods of time, whereas with the same change in initial conditions the periodic block produced two different signals.



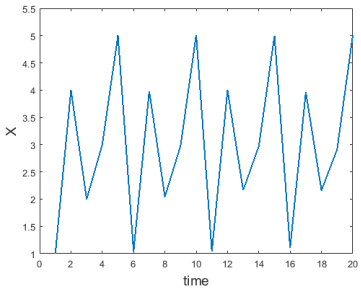
(a) Initial conditions [1 3 5 4 2]



(b) Initial conditions [1 3 5 4 2]



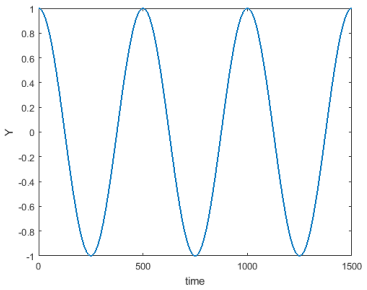
(c) Initial conditions [1 4 2 3 5]



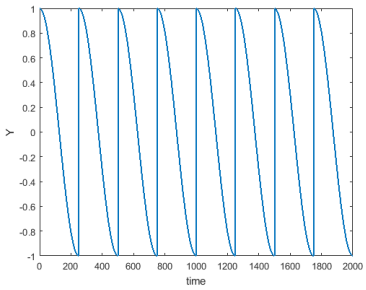
(d) Initial conditions [1 4 2 3 5]

Figure 2.5: Output of an integrator block (left) and of a periodic block (right) with different initial conditions.

The periodicity of the signal is also a factor to be considered when working with this type of block. Although most signals can, under certain conditions, be represented with multiple periods one has to take into consideration the number of initial conditions needed to generate the same signal. The effect of an incorrect change in period can be seen below in figure 2.6, where the period of the H block was cut in half but the initial conditions remained unchanged, thus generating a completely different signal than the previous output.



(a) Signal with period 500



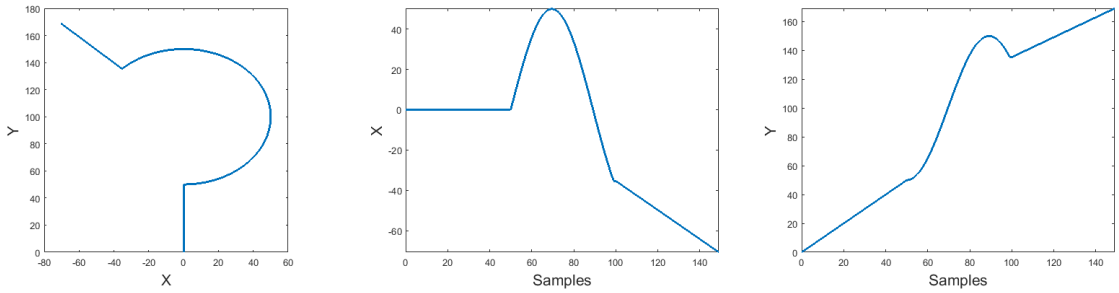
(b) Signal with period 250

Figure 2.6: Output of a periodic H block with input $x = \cos(2\pi \frac{t}{500})$ and period $T = 500$ (a) and $T = 250$ (b)

Variable H

While some motions can be described by a single non-stationary model, the possibility of existing a varying H block is high, especially when dealing with real case scenarios. Such an example is the path of vehicles and pedestrians in a city, where the different roads and streets provide a set of choices for the continuation of the motions. A common path with variable H is, for example, the path that a vehicle creates when travelling through a roundabout, where the entry and exit motions can usually be generated

by an integrator block H_i^n of order 2 or higher and the middle motion by an excerpt of a sinusoidal H_p^T. Since the motion is bi-dimensional, there is a need for two different motion vectors, although they are both described in the same way as before. Figure 2.7 illustrates the motion in the plane and the vectors varying with time.



(a) Motion of a vehicle in a roundabout. (b) Motion of the vehicle along the first coordinate. (c) Motion of the vehicle along the second coordinate.

Figure 2.7: Bi-dimensional motion (a) and its one-dimensional counterparts (b) and (c).

The prediction of the future motion is done through the use of an algorithm that takes as input the motion of a target or a group of targets, $\tilde{\mathrm{x}}$. The process of forecasting the coordinates starts with the inversion of the effect of the H block on the motion signal, thus generating an estimate of y, followed by the computation of the predicted values of the stationary signal m steps ahead by recreating the effect of the AR model. In summary, the two main steps of the algorithm are expressed as

$$\bar{y}(t) = \frac{1}{H}\tilde{x}(t) \tag{2.14}$$

and

$$y(t+m) = \frac{1}{A}\bar{y}(t). \tag{2.15}$$

The structure of the prediction algorithm is shown below in figure 2.8. Note that $\tilde{\mathrm{x}}$ is related to the output of the motion generation model by

$$\begin{aligned} \tilde{x}(t) &= x(t) + \mu(t), \\ \mu(t) &\sim N(0, \sigma_\mu^2), \end{aligned} \tag{2.16}$$

since it is assumed that there is some error μ present on the gathered motion data, possibly with machine and/or human origin (*e.g.* low image resolution, inaccurate point selection, etc.).

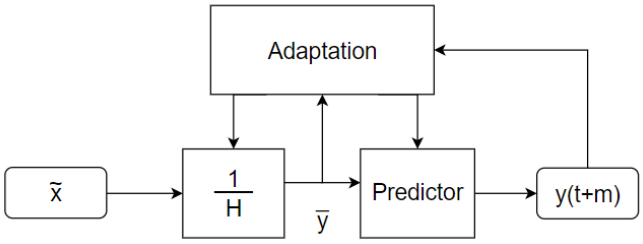


Figure 2.8: Block diagram of the prediction algorithm.

Chapter 3

Signal model identification

Here the first phase of the prediction algorithm is explained. This section corresponds to the initial stage of the Box-Jenkins approach, concerning the valid identification and selection of the models that make up the system being studied.

3.1 Adaptation

The actual scheme of the system that provides the data available for the development and testing of the algorithm is inaccessible. An alternative layout was proposed in the previous chapter to overcome this lack of knowledge in the hopes that with it similar results to the true values of the series may be achieved. Through a comprehensive inspection of the data, a configuration of parameters that result in an acceptable substitute of the desired model can be obtained. Such an estimate can be computed separately for each model, or in a loop until convergence is reached. No restrictions exist that prevent the model from being time variant, meaning it is allowed to change one or all of its models at any given instant. It is only assumed that these changes, if they occur, do so at a slower pace than the observation rate. In other words, the system stays constant long enough to produce a considerable amount of points that originate from the same model configuration. There is then the need for a section, in the algorithm to be developed, that addresses the estimation of both stationary and non-stationary parts of the motion model adaptively.

3.2 Non-stationary model

The motion model, to which the letter H is given for the purpose of identification, is responsible for the overall shape of the signal. With this model, a signal that bears some sort of resemblance to an actual trajectory is created, in the sense that it more or less appears to be similar to the path a target might take. In general, most often than not, a sequence of points pertaining to the motion of a target in a given interval contains more than one type of movement. This behavior can be simulated through the use of various models with different initial conditions, although it demands the presence of an adaptive factor in the model estimator. Furthermore, since the scope of this work includes real-time prediction as one

of its aims, an online method is chosen over other batch options. To ensure the correct representation of the model a performance evaluator conducts a test on a set of predefined models, running in parallel, and then chooses the one that outperforms the others, according to a certain criteria.

Understanding the way the model works is of utmost importance when specifying the evaluation criteria. An explanation of the theory behind the performance review is then necessary. Let x symbolize the set of data available and x_H the estimates obtained with a predefined motion model H. A qualitative analysis of the forecast accuracy, to assess the suitability of different models, is performed through the examination of the evolution of the prediction error, with the goal of minimizing the Euclidean distance between the calculated values and the actual reference. A measure of the eligibility of a model is obtained through the examination of a cost function that considers the sum of the squared errors

$$J(x_H) = \sum_{i=1}^{t} \big(x(i) - x_H(i)\big)^2, \tag{3.1}$$

or

$$J(e) = \sum_{i=1}^{t} e^2(i), \tag{3.2}$$

more commonly known as the Least Squares method.

While this equation provides the best fit for the whole motion, given the set of observations, it does not take into consideration the possible change in parameters that might have occurred during the gathering of the data, which plays a major role in the determination of the most appropriate model at a certain time. Instead, the system will give every error sample, regardless of how old it is, the same weight and consequently try to fit a single motion model to what may be the work of several motion models. To attenuate this effect and improve the performance evaluator, the error values will pass through a temporal low-pass filter, as depicted in figure 3.1, so that the system weighs each value based on how far ago it was obtained.

Consider a first order low-pass filter with transfer function

$$H_{LPF}(z) = \frac{1-\alpha}{z-\alpha}, \tag{3.3}$$

with impulse response

$$h_{LPF}(n) = (1-\alpha)\alpha^{n-1}u(n-1), \tag{3.4}$$

where $u(n)$ denotes the step function. The set of squared errors of a model H, denoted $e^2(n)$, can be described as a sum of dirac pulses with varying amplitude

$$e^2(n) = \sum_{k=0}^{\infty} e^2(k)\delta(n-k). \tag{3.5}$$

The output of error filtering can then be computed as the convolution of the input with the impulse response

$$y(n) = e^2(n) * h_{LPF}(n), \tag{3.6}$$

or

$$y(n) = \sum_{k=0}^{\infty} e^2(k) h_{LPF}(n-k). \tag{3.7}$$

Replacing (3.4) in the equation above one obtains an expression of the output for $n > 0$

$$y(n) = (1-\alpha) \sum_{k=0}^{n-1} e^2(k) \alpha^{n-k-1}, \tag{3.8}$$

with $y(n)$ the evaluation function for a predefined model H given n observations.

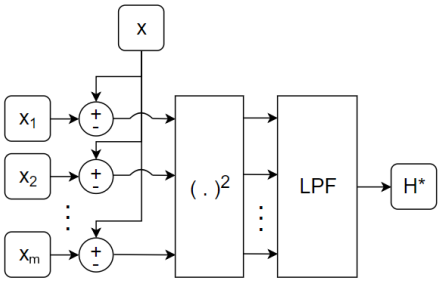


Figure 3.1: Model of the performance evaluator.

There is also the problem of the change in initial conditions in every iteration. Under this dynamical environment, the errors of a one-step predictor may be too low for the correct distinction of the models to occur. In order to counter this problem, a prolongation of the behaviour of each model is done and the signal passing through the filter will be the sum of a set of errors instead of a single error sample.

Another approach to solve this problem could consist in having a set of models with static (or perhaps with some sort of dependence between) initial conditions.

3.3 Stationary model

Typically, the trajectories found in the real world are imperfect and possess some perceivable amount of noise, compared to the more elegant theoretical paths. According to the proposed diagram of the data signal (figure 2.1), these discrepancies between the desired and the actual values are caused by an auto-regressive model, possibly with a moving average. In many cases, the addition of an input to compensate the effect of these inconsistencies results in a more correct guess of the output points. Adhering to that line of thought, it follows that an estimation of the parameters of such a model is the next logical step to take. This section provides a comparison between one of the most known algorithms for parameter estimation, the Least Squares method, and a modified version of the sparsity aware LASSO algorithm.

Formally, the problem states that given a set of p observations $\mathbf{Y} = [\ y(1) \quad y(2) \quad ... \quad y(p)\ ]^T$ and the corresponding input data that originated it $\mathbf{\Phi} = [\ \varphi(1) \quad \varphi(2) \quad ... \quad \varphi(p)\ ]^T$, the goal is to estimate the vector of the parameters θ such that

$$\mathbf{Y} = \mathbf{\Phi}\theta. \tag{3.9}$$

The solution of the equation above for $p = \dim \theta$, if it exists, comes in closed form, and the proof for this

statement is demonstrated quickly, with only two steps required

$$\mathbf{\Phi}'\mathbf{Y} = \mathbf{\Phi}'\mathbf{\Phi}\theta, \tag{3.10}$$

and

$$(\mathbf{\Phi}'\mathbf{\Phi})^{-1}\mathbf{\Phi}'\mathbf{Y} = \theta, \tag{3.11}$$

with $(\cdot)'$ and $(\cdot)^{-1}$ denoting the transpose and inverse transformations, respectively. As the observations continue piling up, however, the solution may cease to be. The objective in that case is the obtainment of a solution that minimizes the error

$$e = \mathbf{Y} - \mathbf{\Phi}\theta. \tag{3.12}$$

3.3.1 Recursive Least Squares

The first method described here takes a more straightforward approach to the problem at hand. The Least Squares method attempts to find the values of θ that, for each set of data, result in an output that is closest to the actual answer than any other option. This distance is measured in an Euclidean space. In other words, the algorithm consists in finding the minimizer of the sum of the squared errors

$$\hat{\theta} = arg\,\min_{\theta} \sum_{k=1}^{p} ||y(k) - \varphi(k)^T\theta||^2. \tag{3.13}$$

This approach has, however, a few drawbacks, since it requires all the data to be known before the estimation process. As time progresses, this procedure becomes impractical, due to both the computation needed to execute it and the memory required to store the data. An online recursive method is then sought to allow for a solution that achieves the same results while requiring a more reasonable amount of resources to function. The observation sets can be rewritten as

$$\begin{aligned} \mathbf{Y}(k) &= [\ \mathbf{Y}(k-1) \quad y(k)\ ]^T, \\ \mathbf{\Phi}(k) &= [\ \mathbf{\Phi}(k-1) \quad \varphi^T(k)\ ]^T, \end{aligned} \tag{3.14}$$

whereas (3.13) can be expressed as

$$\hat{\theta}(k) = \Lambda^{-1}(k) \sum_{t=1}^{k} y(t)\varphi(t), \tag{3.15}$$

with

$$\Lambda(k) = \sum_{t=1}^{k} \varphi(t)\varphi^T(t), \tag{3.16}$$

the information matrix that verifies

$$\Lambda(k) = \Lambda(k-1) + \varphi(k)\varphi^T(k). \tag{3.17}$$

Using (3.15),(3.16) and (3.17) it is possible to reach a recursive expression for the estimated parameters

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \Lambda^{-1}(k)\varphi(k)\big[y(k) - \varphi^T(k)\hat{\theta}(k-1)\big]. \tag{3.18}$$

In order to avoid inverting a matrix every iteration, instead of the information matrix, the covariance matrix is used

$$P(k) = \big[\Lambda(k-1) + \varphi(k)\varphi^T(k)\big]^{-1}, \tag{3.19}$$

since its value at time k is, applying the Matrix Inversion Lemma, given by

$$P(k) = P(k-1) - \frac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{1 + \varphi^T(k)P(k-1)\varphi(k)}. \tag{3.20}$$

Additionally, if a forgetting factor [25] is added to the cost function (3.13)

$$\hat{\theta} = arg\,\underset{\theta}{min} \sum_{k=1}^{p} \beta^{p-k}\big(y(k) - \varphi(k)^T\theta\big)^2, \tag{3.21}$$

to enable an ever-lasting sensitivity to the change in parameters, then the equations for the RLS come as

$$\begin{aligned}
&\varepsilon(k) = y(k) - \varphi^T(k)\hat{\theta}(k-1),\\
&\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\varepsilon(k),\\
&K(k) = P(k)\varphi(k),\\
&P(k) = \beta^{-1}\left[P(k-1) - \frac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{\beta + \varphi^T(k)P(k-1)\varphi(k)}\right].
\end{aligned} \tag{3.22}$$

The acceptable range for the forgetting factor values is between $0 < \beta < 1$, with bigger values providing the system with good stability and low variation, but worse tracking abilities, while lower values increase the tracking efficiency of the system and the variation of the system, which may result in a loss of stability. An overall idea of the number of samples that affect the outcome can be obtained by looking at the asymptotic memory $N = \frac{1}{1-\beta}$.

Further enhancements can be made by implementing a dynamic forgetting factor, since a fixed one is prone to estimator windup or covariance blow-up. This happens when the system input is not persistent or when it decreases significantly in amplitude and is not able to excite the system anymore. Under these circumstances, $\varphi(k)$ holds no new information and $P(k) = \beta^{-1}P(k-1)$, which causes the information matrix to grow in size and provide unreliable and biased estimates. Another problem is the slow convergence of the estimates after a long period of constant parameters. Consider the weighted sum of the squares of the *a posteriori* errors

$$\Sigma(k) = \beta(k)\Sigma(k-1) + \big(1 + \varphi^T(k)K(k)\big)\varepsilon^2(k), \tag{3.23}$$

with $\Sigma(k)$ the information content of the algorithm at time k. According to [26], a well-known method

for choosing the forgetting factor is by keeping the sum above constant $\Sigma(k) = \Sigma(k-1) = ... = \Sigma_0$, where Σ_0 is set to be equal to the expected noise variance, assuming it is known, times a nominal asymptotic memory length, $\Sigma_0 = \sigma_e^2 N_0$. From (3.23) and the asymptotic memory equation comes

$$N(k) = \frac{\Sigma_0}{[1 - \varphi^T(k)K(k)]\varepsilon^2(k)}.$$

With this choice it is possible to define the method for choosing the forgetting factor as follows

$$\beta(k) = max\left\{\beta_{min}, 1 - [1 - \varphi^T(k)K(k)]\frac{\varepsilon^2(k)}{\sum_0}\right\}. \tag{3.24}$$

3.3.2 RLS-Weigthed LASSO

Although the Least Squares method boasts some desirable qualities it is unable to provide a result with a lower dimension than that of the expected solution to the problem, since it considers that all the given variables affect the outcome of the known data equally. Consequently, every one of these variables is assigned a value different than zero, with no regard to its own or the overall magnitude of the values. In general, considering the wrong order when estimating model parameters can result in an overfit or underfit of the data, that in turn can then render future outcome predictions useless, due to the inaccuracy of the estimates. The LASSO method handles this problem by adding a new component to (3.13) with the purpose of nullifying parameters considered irrelevant for the model, thus yielding sparse solutions when the data calls for them. This method takes into consideration the ℓ_1 norm of the parameter vector θ as an additional restriction to try to minimize the number of non-null entries, at the cost of becoming non-differentiable and losing all the perks that come with the trait. The reformulated equation remains similar to the RLS estimate, with the exception of the new term that accounts for the sparser considerations in the solutions,

$$\hat{\theta} = arg\min_{\theta} \left|\left|\mathbf{Y} - \mathbf{\Phi}^T\theta\right|\right|_2^2 + \lambda\left|\left|\theta\right|\right|_1. \tag{3.25}$$

The change in estimate translates to a modification of the objective function, that now comes as

$$J(\theta) = \sum_{k=1}^{p} \beta^{p-k}\left|\left|y(k) - \varphi(k)^T\theta\right|\right|_2^2 + \lambda(k)\left|\left|\theta\right|\right|_1. \tag{3.26}$$

Regrettably, as with in the RLS, this computation becomes more complex as time progresses, increasing both the memory requirements and computation time needed. Hence, like before, a version of (3.25) that can be recursively updated is deduced. Rewriting the estimate equation

$$\hat{\theta} = arg\min_{\theta} a(n) + \theta^T R(n)\theta - 2\theta^T r(n) + \lambda(n)||\theta||_1, \tag{3.27}$$

where

$$a(k) = \sum_{i=1}^{k} \beta^{k-i} y(i)^T y(i),$$

$$r(k) = \sum_{i=1}^{p} \beta^{k-i} \varphi(i)^T y(i),$$

$$R(k) = \sum_{i=1}^{p} \beta^{k-i} \varphi(i)^T \varphi(i),$$

does the trick.

Under these added constraints the estimate retains its convexity, making it possible to find the minimum through linear programming techniques. This option is, by itself, a rather beneficial quality when considering the computation relief between such methods and more complex ones.

To further stimulate the sparseness of the solution, a signal dependent penalty function [27] that weighs each ℓ_1 norm term $|\theta_n|$ individually is applied. The weight of each term is computed through

$$w_{\mu(k)}(|\theta|) = \frac{(a\mu(k) - |\theta|)_+}{\mu(k)(a-1)} u(|\theta| - \mu(k)) + u(\mu(k) - |\theta|), \tag{3.28}$$

where $u(\cdot)$ is the step function and $(\cdot)_+$ the non-negative part of the function in parentheses, also described as

$$(x)_+ = \begin{cases} x, & x > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.29}$$

This function acts as a filter on each entry, fully penalizing (w = 1) those it deems unnecessary or undesirable, characterized by their low magnitude, and not imposing any penalization term on higher magnitude ones. Terms whose importance is neither high or low are penalized linearly according to their modular value. This last category consists of any parameter bigger than μ_k and smaller than $a\mu_k$, therefore, as the position of this region greatly influences the appearance of the output, a wise selection of both variables is necessary. It should be noted that (3.28) is updated using RLS estimates that are computed alongside, hence the name of the method. The cost function for the, now complete, RLS-Weighted LASSO (RW-LASSO) comes as

$$J(k)^{\text{RW-LASSO}} = \sum_{i=1}^{k} \beta^{k-i} ||y(i) - \varphi(i)\theta||_2^2 + \lambda(k) \sum_{n=1}^{N} w_{\mu(k)}(|\hat{\theta}_n^{\text{RLS}}(k)|)|\theta_n|, \tag{3.30}$$

that, although non-differentiable can be solved with the help of techniques such as sub-gradient based iterative minimizers or proximal gradient methods. Seen as this work seeks to apply the knowledge gained from [24] to the scope of this report, the latter method is considered, since it is shown that the first option suffers from slow convergence and does not guarantee the convergence of the parameter estimates $\hat{\theta}$(k) to their true value as k tends to infinity. Consequently, the main step of the algorithm built for solving (3.27) consists in computing

$$\text{Prox}_{\alpha_i ||\cdot||_1}\big(\theta^i - \alpha_i \nabla g(\theta^i)\big) = S_n\big(\theta^i - \alpha_i \nabla g(\theta^i)\big), \tag{3.31}$$

where

$$[S_n(\theta)]_n = sign(\theta_n)(|\theta_n| - \eta)_+$$

with $\eta = \alpha_i \lambda$,known as the Soft-Thresholding function, and $\alpha_i = \frac{\alpha}{i}$ or $\alpha_i = \frac{\alpha}{\sqrt{i}}$ the step size ($\alpha > 0$). It is shown in [27] that with a penalty parameter $\sqrt{k} < \lambda(k) < k$, a threshold parameter $\mu_k = \frac{\lambda(k)}{k}$ and the knowledge of all the past data, i.e. a forgetting factor of $\beta = 1$, the RW-LASSO method fulfills the oracle properties which guarantee that the performance of the estimator matches that of the true underlying model. Unfortunately, this fails when considering an estimator capable of handling model transitions in time ($\beta < 1$). The trade-off between performance and adaptability is something that should be meticulously studied to best fit in within the context of the problem.

A pseudocode representation of the algorithm described above is presented next.

Algorithm - RLS-Weighted LASSO

Inputs: $\beta,\ \epsilon,\ \varphi(k),\ y(k),\ a,\ \alpha,\ i_{\max}$

Initialization: $r(0) = 0,\ R(0) = 0,\ \lambda(0),\ \hat{\theta}(0) = 0,\ P(0) = \delta I,\ \delta = constant$

1. **for** $k = 1, 2, ...$ **do**
2. $r(k+1) = \beta r(k) + \varphi^T(i) y(i)$
3. $R(k+1) = \beta R(k) + \varphi^T(i)\varphi(i)$
4. $\varepsilon(k) = y(k) - \varphi^T(k)\hat{\theta}(k-1)$
5. $K(k) = \frac{P(k-1)\varphi(k)}{\beta + \varphi^T(k)P(k-1)\varphi(k)}$
6. $P(k) = \beta^{-1}\big[P(k-1) - K(k)\varphi^T(k)P(k-1)\big]$
7. $\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)\varphi(k)\varepsilon(k)$
8. $w_{\mu(k)}(|\theta|) = \frac{(a\mu(k) - |\theta|)_+}{\mu(k)(a-1)} u\big(|\theta| - \mu(k)\big) + u\big(\mu(k) - |\theta|\big)$
9. **for** $i = 1, ..., i_{\max}$ **do**
10. $\text{Prox}_{\alpha_i ||\cdot||_1}\big(\theta^i - \alpha_i \nabla g(\theta^i)\big) = S_\eta\big(\theta^i - \alpha_i \nabla g(\theta^i)\big)$
11. **end for**
12. **end for**

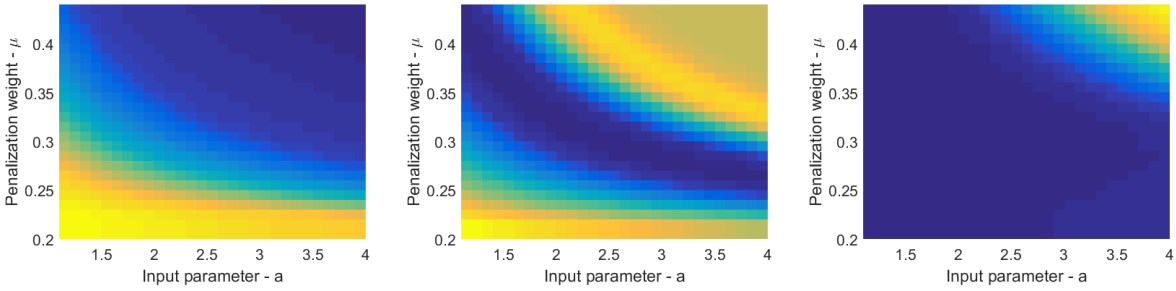
3.3.3 Variable calibration

The RW-LASSO is a rather flexible method when it comes to its output appearance, due to the amount of adjustable variables that in turn influence the outcome of the estimator. Below, a study on the impact of each parameter in the output of the system is described. To better modify these parameters, the study will focus on understanding how the interactions between certain inputs affect the estimates instead of considering every individual effect.

In total, there are five distinct variables that dictate the behaviour of the algorithm. From this set, three groups can be formed, based on the way they alter the performance of the method. One consists of variables that change the sparsity of the output, λ and a, another contains those that influence the system response time, α and i_{max}, and the last comprises the ones that deal with the adaptability of the system to time varying models, in this case, the forgetting factor β.

To adjust how sparse the solution is, first the range of acceptable values for the estimates must be specified. The input parameter vector will consist of 8 entries, of which a maximum of 4 are expected to

be relevant at any given time, on account of the sparseness the estimator strives for. The AR model is also assumed to be stable. These two conditions make it so that the range of interest for any parameter value is somewhere around 1. The algorithm is built to limit automatically the magnitude of the estimates so as to stabilize the model, but it still requires some additional information to distinguish between desirable and undesirable variable values. A proper selection of a and λ ensures that the method works toward finding an answer that meets the user's desires or expectations. The penalization area of (3.28) can be made wider or narrower with a steeper or more gradual slope at its end, by tweaking the input values. The distinction is made by treating every entry below a predefined threshold (μ_k) as an error and attributing them with the maximum weight of 1. The range of values in which there is uncertainty about the origin of the estimate, i.e. whether it comes from a valid estimation or if it is more likely to worsen the predictions made with the model, extends until a times the limit of the previous area. Figure 3.2 shows the evolution of the mean squared error on three models with different parameter magnitudes. Brighter colors are used to express higher values of the error, while the darker blue hues represent the more accurate combinations of the parameters. From the plots one can understand the collective effect that the variables exert on the system. There is a trend in the behaviour of the estimator. With the combined increase in both parameter values there is a decrease in error, followed by a subsequent increase. This happens because the method transitions from overfitting the model to underfitting it, however, in extreme cases, only one of these effects occurs. In figure 3.2 (a), the underfit of the model results in a response with all the parameters set to zero, whereas the actual magnitude of the coefficients in 3.2 (c) is big enough to reduce the effect of overfitting to an acceptable level.



(a) MSE on AR with $A = 1 - 0.2z^{-1} + 0.1z^{-}3$.

(b) MSE on AR with $A = 1 - 0.4z^{-2} + 0.5z^{-}3$.

(c) MSE on AR with $A = 1 - 1.5z^{-1} + 0.7z^{-}2$.

Figure 3.2: Mean squared error evolution of the estimation of sparse AR models with different levels of magnitude. The scale of the parameters from left to right is, in the context of this work, considered small, medium and big.

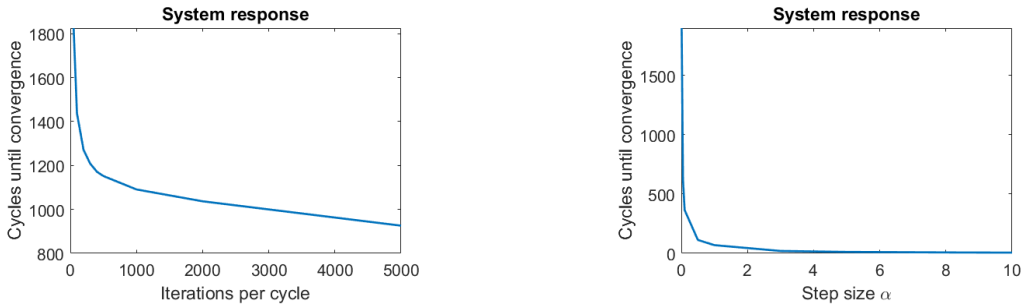
The best option lies somewhere between choosing a highly restrictive penalizing function for small values with a thin uncertainty area and one with a subtler, earlier drop in penalization that allows. Opting for a more balanced function that is somewhat restrictive with a narrower area of uncertainty, the author set the parameters $\lambda(k)$ such that μ_k was kept constant at 3.1 and a to 2.

Regarding the system response time, despite producing similar results, α and i_{max} approach the problem very differently. While they both attempt to maximize the extent of optimization possible at any time instance, one does so by reducing the number of iterations necessary to reach a certain threshold, whereas the other manages the limit of computation cycles the algorithm is allowed to perform.

The step size regulates how much a point moves when optimizing a function. It does so by controlling the effect that the point-wise gradient inflicts on the current parameter values, thus altering the maximum quantity of optimization possible per iteration. The degree of variation in response time is proportional to the magnitude of α. This means that with the same number of iterations, the distance the optimization function travels increases with the step size, making it logical to consider choosing the biggest available value for this variable. However, some values of α end up moving the estimate to a point with a higher cost than the previous one. Even if the value for the step size is reduced every iteration, given a big enough initial α and an insufficient amount of iterations, the gap in cost will never be closed. In such a situation, the estimation ends up diverging.

On the other hand, because the estimate is optimized via a numerical method, there exists a maximum feasible number of iterations which imposes a limit on how much the value can change with every cycle. These limitations are hardware dependent so, to regulate the feasibility of the algorithm, the variable i_{max} is introduced. This parameter determines the number of iterations per cycle and raising its value decreases the response time at the cost of additional computation time. It is similar to the use of brute force in problem solving, since it is a simple, yet inefficient tactic that focuses solely on producing results. Studying the interaction between these two behaviours should prove useful in finding a pair of values that diminishes the system response in a fast, stable and affordable way.

Fixing the proximal gradient step size and sweeping through various values for the number of iterations revealed the nature of the effect the change introduces to the system. As expected, the decrease in stabilization time is somewhat inversely proportional to the iterations done per cycle. This decline is not at all constant, becoming less steep for higher values of i_{max}. Switching the fixed variables yields a similar result, albeit a more drastic one. The contrast in the drop values presented in these graphics (figure 3.3) meets the expectations, seeing that the efficiency of both solutions corresponds to the nature of effect these have in the system behavior.



(a) System response time with varying number of iterations and $\alpha = 0.01$.

(b) System response time with varying step size and $i_{max} = 10$.

Figure 3.3: Effect of the variation of the number of iterations (left) and the step size (right) on the response time of the system.

It cannot be forgotten that while the competence of these methods greatly differs, the importance of each of the parameters does not, for each one positively influences the others performance. That being said, the response times for the different tests performed are shown in table 3.1.

Table 3.1: System response time, in epochs, with varying step sizes and number of maximum iterations.

		Step size α											
		0.01	0.05	0.1	0.5	1	3	5	7	8.5	10	50	100
i_{max}	10	1900+	622	365	113	69	20	12	9	7	6	2*	1*
	20	1900+	538	312	98	54	16	10	7	6	5	1*	1*
	50	1826	448	262	86	39	13	8	6	5	4	1*	1*
	100	1438	407	232	76	33	12	7	5	4	4	1*	1*
	200	1271	364	209	69	29	10	6	5	4	4	1*	1*
	300	1209	344	195	64	27	10	6	5	4	3	1*	1*
	400	1172	333	189	61	26	9	6	4	4	3	1*	1*
	500	1152	324	185	56	25	9	6	4	4	3	1*	1*
	1000	1091	304	173	52	23	8	5	4	3	3	1*	1*
	2000	1037	287	163	46	21	8	5	4	3	3	1*	1*
	5000	926	260	152	39	19	7	4	3	3	3	1*	1*

*Parameter estimation eventually diverged.

Lastly, the effect of the variation in the forgetting factor values is studied. Although this parameter also affects the system response time, it concerns mainly the adaptability of the algorithm to time-varying models. For this reason and for brevity too, the author thought it best to avoid grouping it with the previous variables. Much like the in the Least Squares method, the RW-LASSO β controls the susceptibility of the algorithm to changes in the system parameters. In fact, since they both work in the same manner, the results conveyed here apply to either of the two algorithms.

The conducted tests aim at understanding the consequences of applying different β values with distinct restrictions on the outcome of the estimation methods. More specifically, a comparison between the use of fixed and variable forgetting factors is done. First, a value close to, but strictly smaller than, 1 was chosen to allow the estimator to react to changes in parameters while maintaining an accurate estimate with a small misadjustment (figure 3.4). This resulted in a more stable prediction at the cost of a slower reaction time.

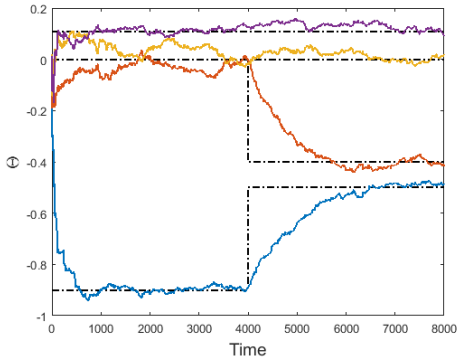
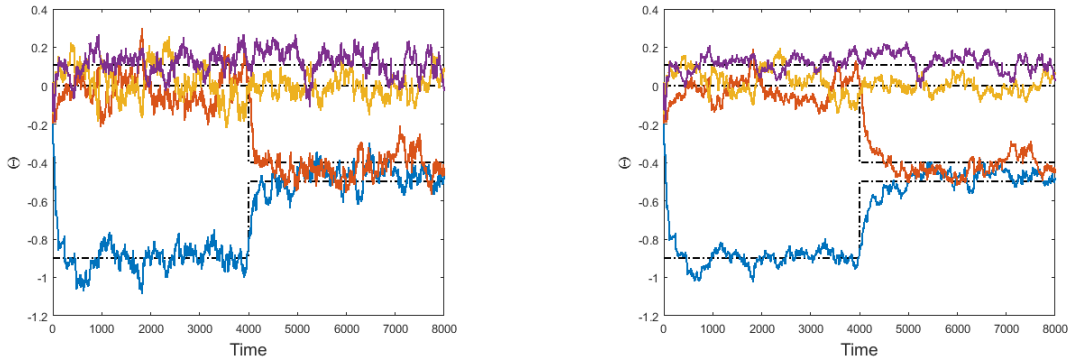


Figure 3.4: RLS estimation with $\beta = 0.999$.

Then, two different values for the lower bound of the forgetting factor were defined, namely $\beta_{\min} = 0.99$

and $\beta_{\min} = 0.995$, to study the effects of distinct intervals of possible values in the system response.

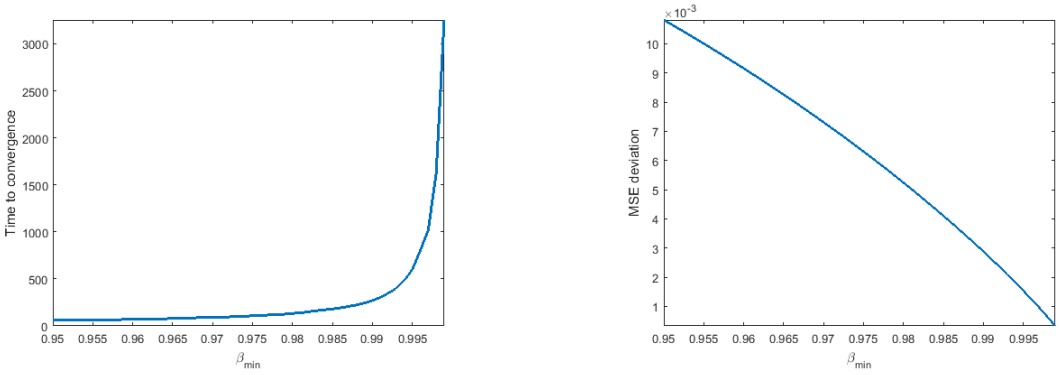


(a) RLS estimation with $\beta \in [0.99, 1[$.

(b) RLS estimation with $\beta \in [0.995, 1[$.

Figure 3.5: RLS estimation with a variable forgetting factor.

The results showed that a stricter limit provides a more stable estimation with lower misadjustment while a more relaxed limit offers better tracking capabilities, as was expected. To further illustrate this trade-off, some tests were conducted with various values for the lower limit of the forgetting factor value, and the results obtained were graphed for a clearer interpretation. In total, 50 values for $\beta_{\min}$, ranging from 0.95 to 0.999, were tested, and for each of those values 100 tests were performed, so as to attenuate the effects of the noise in any conclusions taken. Figure 3.6 depicts the growth of both the response time (3.6-a) and the misadjustment (3.6-b) of the system as a function of $\beta_{\min}$. For the latter, the mean squared error deviation of the estimates was computed as a way to quantify their variation around the actual values of θ.



(a) System response time.

(b) System variance.

Figure 3.6: RLS estimator characteristics as a function of the minimum forgetting factor allowed $\beta_{\min}$.

As concluded before, a higher value for the lower limit of the forgetting factor leads to a slower convergence of the estimates, with the number of samples needed for the stabilization of these estimates increasing exponentially as the limit grows. On the other hand, the system tends to become more stable the closer the limit is to 1. The decrease in variation is, however, not as drastic as the increase in response time. While the results were expected, i.e., bigger variations mean less iterations needed to converge as it is similar to bigger steps being taken at each iteration, this experiment was useful since it established a relation between the growth of both characteristics.

3.4 Markov Model

When working on problems with multiple approach options, there are some questions that must be asked. These questions revolve around the quality of the solution and serve as a guide to the attainment of an acceptably effective and efficient method whose results are in conformity to the needs and wants of the user. After understanding the details of a problem and reaching a consensus on how to tackle it, one should always ponder if that is indeed the best way to solve it, while considering other alternatives that could also work given the features of the phenomena being studied. In that regard, this author provides the reader with another perspective on the time-series forecasting issue, in hopes that it contributes to the discovery of a better solution or to an improvement of the current one. In particular, the most basic form of a Markov process, denominated a Markov chain, is considered.

A system is described as being a Markov chain if it can be modelled as a sequence of states whose possible outcomes, i.e. future states, at any specific instant in time, are determined solely by the state in which the system is at that time. Moreover, it is treated as a Hidden Markov Model (HMM) when part of the state is inaccessible. This seems to be the case of this problem, since from the observations available the amount of information is not enough to fully understand the reasons behind the system's behaviour.

For such an environment, a nonlinear filtering supervisor [28], closely resembling the Baum algorithm is chosen. This method uses Bayes law and some standard nonlinear filtering techniques to recursively propagate in time the probabilities of the states in two steps. The first operation models the state values based on the known interactions between the state space in two consecutive time instances, while the second one reshapes the estimates by filtering them after a new observation is made.

3.4.1 Prediction step

Let S be the full set of possible states, with $S_i, i = 1, 2, ..., n$ one of the n distinct options available. From the definition of the Markov chain, the probability of jumping from the current state i to another state j is determined by the transition coefficient connecting the pair, t_{ij}. A matrix that describes the relation between all of the pairs of states is named a transition matrix and shall henceforth be given the letter T for future reference. Every row of T must sum to 1, since it is a matrix describing the probability of every possible event within a system. Analogously, the likelihood of being in state i at a certain time instance k, denoted as $P_i(k|k-1)$, is computed from the weighted sum of the prior probability of all previous states,

$$P_i(k|k-1) = \sum_{j=1}^{n} t_{ji} P_j(k-1). \tag{3.32}$$

The expression above relates the entire state space at time k to a particular state residing in the following time instance. A generalization of (3.32) that depicts the variation of the whole state space throughout time is achieved simply by stacking every distinct outcome in a vector. With $P(k)$ the set of probabilities

of all states at time k, the predicted values for each of the states in the next instance of time are given by

$$P(k|k-1) = T'P(k-1), \tag{3.33}$$

with

$$P(k|k-1) = [P_1(k|k-1) \quad P_2(k|k-1) \quad \cdots \quad P_n(k|k-1)]'.$$

This operation, referred to as the prediction step, is used to infer outcomes based solely on the expectation of the prior knowledge evolution.

Taking the result of (3.33) and admitting it as the future prior enables the continuous use of the expression and doing so will result in the convergence of P. This "last" state is denominated the stationary probability vector π and is invariant to the application of the prediction step since, by definition, the outcome of the equation for a converged state is itself the converged state. In other words,

$$\pi = T'\pi. \tag{3.34}$$

Consider, for example, a situation where there are three possible states, whose transition matrix is given by

$$T = \begin{bmatrix} 0.9 & 0.07 & 0.03 \\ 0.15 & 0.8 & 0.05 \\ 0.01 & 0 & 0.99 \end{bmatrix}, \tag{3.35}$$

and with every state being equiprobable in the beginning, $P_0 = \left[\frac{1}{3} \quad \frac{1}{3} \quad \frac{1}{3}\right]'$. Each of the states possesses a common characteristic, the preference for the continuity of the state over the transition to another. One of them takes it further by completely eliminating the possibility of changing to the second state. This trait creates smoother transitions opposed to the inclination towards inter-state transitioning, that results in an oscillatory response. This effect can be perceived in the figure below.

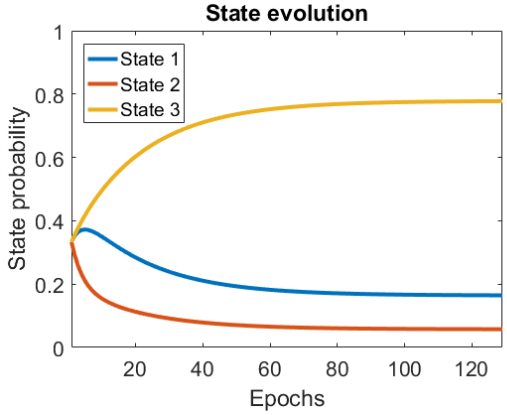


Figure 3.7: Evolution of the predicted state through time.

3.4.2 Filtering step

Whenever a new observation becomes available, a correction of the probabilities of the state space is necessary to ensure the plausibility of the predictions. This adjustment is the result of passing the guess on the state likelihood through a filter that takes as input the last observation obtained and the estimates of each state as to the value of this variable. These estimates are made using the known input configurations $x(k)$ that lead to the observation value and the model parameters of each distinct state θ_i,

$$\hat{y_i}(k) = \theta_i' x(k). \tag{3.36}$$

From there, the probability of an observation y given a certain state S_i is computed based on the disparity between both values

$$P(y(k)|S_i) = exp\Big(- \frac{\big(y(k) - \theta_i' x(k)\big)^2}{2\sigma^2}\Big), \tag{3.37}$$

where $exp(\cdot)$ is the natural exponential function. This function guarantees that some desirable conditions are met, such as the unit value for completely accurate predictions and an adjustable interval, based on the noise variance, for accurate guesses in which its value is still close to 1. It also boasts the differentiability trait, much appreciated in various situations.

Furthermore, if the assumption of variable independence is made, (3.37) can be generalized as the product of the partial probabilities of each of the variables,

$$P(y(k)|S_i) = \prod_{j=1}^{N} exp\Big(- \frac{\big(y_j(k) - \theta_i' x_j(k)\big)^2}{2\sigma_j^2}\Big). \tag{3.38}$$

Finally, by pairing both the prediction and filtering steps, a measurement for the likelihood of any state at the current epoch is obtained as

$$P_i(k) = CP(y(k)|S_i)P_i(k|k-1), \tag{3.39}$$

with C a normalizing constant so that the sum over all possible state probabilities equals 1.

3.4.3 Applications

After analyzing the theory behind an HMM based predictor, a comparison between this and the other possible methods is necessary. A deliberation on the potential uses of the nonlinear filtering supervisor follows.

Since this method assumes a Markov chain as the underlying model, it is unwise to use it alone to predict the outcome of the whole model, due to the sheer amount of available states. The same can be said for the prediction of the stationary model of the system, for the same exact reason. Regarding the non-stationary section, a forecasting of the future states of the H model might surely be a powerful addition to the motion predictor.

A major drawback of this solution is the need for the transition matrix to properly work. This demands

some prior knowledge that may not be available, which greatly hinders the usage of the supervisor in question for reliable estimations. However, circling back to the idea of aiding in the guess of posterior non-stationary model parameters, if one were to obtain the values of the transition matrix via recursive computation or previous data then this additional information could provide a faster switching in models for situations where the targets motion changes models.

As the author sees it, this transition aid could be directly applied in the continuous forecasting of the positions, by studying the variations in state of a vast amount of targets. Another possible application is the supplying of initial conditions in areas of high instability.

Chapter 4

Signal prediction

In this chapter the forecast of the system output is explored. This stage succeeds the model identification phase and makes use of the estimated parameters in the prediction of the targets positions. Here, both the prediction of a single target as well as groups of targets are analyzed.

4.1 Individual target forecasting

The behaviour of the time-series considered in this report comes from a relatively straightforward process, as is shown in the presumed underlying model that produces the data signals studied (figure 2.1). To predict the continuation of the series one needs to recursively apply the models with the estimated parameters and the necessary inputs. The non-stationary model receives two inputs: the past values of the series and the current deviation from the expected, and so does the stationary part: the set of past deviations and the current error value. Here are described three distinct inputs with very disparate acquisition levels. The error signal is assumed to be white, yet forecasting it is impossible. On the other hand, the target position values are easily obtained by extracting the desired information from the available motion data. In the middle lies the auto regressive time-series related to the disparity between the expected and the real outcome. Since this series relies on unknown or estimated inputs, its predictions have a lower accuracy than that of the general motion of the targets. Moreover, the estimates it depends on are assumed to have been computed with the correct H model, however, this is not the case when the motion of the targets changes due to the switching of these models. Nevertheless, the addition of such a signal might produce a more accurate forecasting of the time-series, and so, such an inclusion will be considered.

4.1.1 Linear prediction

While the continuation of the motion signal x is achieved through recursively updating the inputs and applying the same model to them, for the future values of y a different technique is considered. This method produces an estimate and its correspondent expected error value based on the long division of the AR polynomials and the error variance. Formally described as linear prediction, this approach

determines the expression of the output signal that minimizes the deviation of the prediction error in stationary regime, given the O^k observations made until the instant k (with more rigour, O^k is the σ-algebra induced by the observations up to k). In other words, the goal is the discovery of an expression for the estimates such that, under the defined conditions, the difference between them and the actual variable values is minimum

$$E\big[(y(k+m|k) - \hat{y}(k+m|k))^2 | O^k\big], \tag{4.1}$$

where the expression for the output m steps ahead can be derived from (2.1) and (2.4)

$$y(k+m) = \frac{C^*(z^{-1})}{A^*(z^{-1})} e(k+m). \tag{4.2}$$

Executing the long division of the polynomial fraction an equivalent expression for the model is obtained. Dividing the numerator C^* by the denominator A^* results in

$$\frac{C^*(z^{-1})}{A^*(z^{-1})} = F_m^*(z^{-1}) + z^{-m}\frac{G_m^*(z^{-1})}{A^*(z^{-1})}, \tag{4.3}$$

with

$$\begin{cases} F_m^*(z^{-1}) = 1 + f_1 z^{-1} + ... + f_{m-1} z^{-m+1}, \\ \frac{G_m^*(z^{-1})}{A^*(z^{-1})} = f_m + f_{m+1} z^{-1} + ..., \end{cases} \tag{4.4}$$

allowing for a separation of the terms that affect the output in two. The first term deals with the noise samples occurring in future epochs $k+1$ to $k+m$, while the second term manages all of the samples until the current time instance k. Taking (4.1) as the cost function and splitting it using (4.3), one attains an expression consisting of two independent terms, namely the past and present error values and the future ones,

$$J(k) = E\left[\Big(\frac{G_m^*(z^{-1})}{A^*(z^{-1})} e(k) - \hat{y}(k+m|k) + F_m^*(z^{-1}) e(k+m)\Big)^2 | O^k\right]. \tag{4.5}$$

Denoting $F_m^*(z^{-1})e(k+m)$ by $\varepsilon_m(k)$ and separating (4.5) a new expression is obtained

$$J(k) = E\left[\Big(\frac{G_m^*(z^{-1})}{A^*(z^{-1})} e(k) - \hat{y}(k+m|k)\Big)^2 | O^k\right] + E\big[\varepsilon_m^2(k) | O^k\big], \tag{4.6}$$

that reduces to

$$J(k) = \Big(\frac{G_m^*(z^{-1})}{A^*(z^{-1})} e(k) - \hat{y}(k+m|k)\Big)^2 + E\big[\varepsilon_m^2(k)\big] \tag{4.7}$$

when taking into account that the future values of e do not depend on the observations made until the present time and all the past values of the other variables are known. The first term of (4.7) is minimum when it is null and the second term does not depend on the output signal, and so, the optimal predictor is then given by

$$\hat{y}(k+m|k) = \frac{G_m^*(z^{-1})}{A^*(z^{-1})} e(k). \tag{4.8}$$

From (2.1), an expression for the error is deduced by isolating the $e(k)$ term, assuming it is possible to invert the polynomial fraction. With this reformulation, the equation above is rewritten as

$$\hat{y}(k+m|k) = \frac{G_m^*(z^{-1})}{C^*(z^{-1})}y(k). \tag{4.9}$$

Moreover, by assuming that the stationary signal is created using an AR model, (4.9) is simplified to

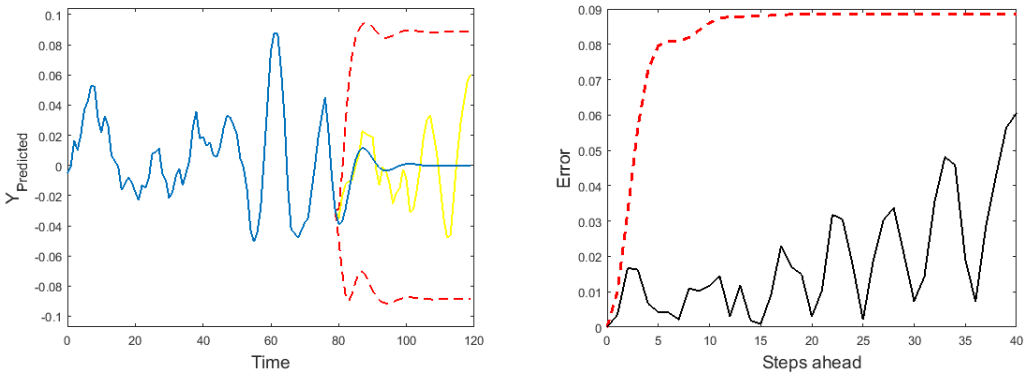
$$\hat{y}(k+m|k) = G_m^*(z^{-1})y(k). \tag{4.10}$$

The second term of the cost function represents a measurement of the forecasting error variance. This value grows with the number of predicted steps since it is computed using the F_m^* polynomial, whose number of terms is the same as m. Due to the properties of the noise (independent values and null mean), the crossed terms of ε_m^2 become zero and the expected value of a prediction m epochs ahead is reduced to

$$E\left[\varepsilon_m^2(k)\right] = (1 + f_1^2 + f_2^2 + ... + f_{m-1}^2)\sigma_e^2. \tag{4.11}$$

Although this variance is the result of an ever growing sum, the increase in value is expected to reduce significantly as the prediction steps grow. This is due to the assumed stability of the AR model.

Consider, for instance, an AR model of order 2 with a Hurwitz polynomial, that is, containing all of its poles inside the unit circle and therefore, being stable, with coefficients 1, -1.5 and 0.7. The last 80 observations are known and so is the variance of the input ($\sigma_e^2 = 0.01$). Any predictions made will not take into consideration any future values of the noise samples and the system will stabilize, behaving similarly to the impulse response of the model. On the other hand, the margin of error will grow as m increases. The behaviour of the predictor is illustrated in figure 4.1-(a), where the predicted values for the samples 81 to 120 are plotted along with the actual values of the output, in blue and yellow, respectively. A margin of error is used to plot the expected maximum prediction error at a given time, in red, and a comparison between it and the actual values of the error is shown in figure 4.1-(b). This margin was computed using (4.11), and although the number of correctly bounded errors varies with different AR models, it seems like a valid way of evaluating the confidence in our prediction.



(a) Output prediction. (b) Prediction error.

Figure 4.1: (a) Comparison between predicted (blue) and real (yellow) output values. (b) Plotting of the prediction error (black).

Even though the expected error growth seems to stagnate after about 10 instances of time, it only takes half that amount to increase above 90% of the convergence value, around ten times the input variance. With this fact in mind, it can be concluded that only predictions of one or two steps ahead are interesting, since they have error values small enough to be reliable.

4.2 Group prediction

More often than not, the image sets from which the target data is extracted comprise of various distinct time-series. From this extensive collection of data, a few targets may be grouped in smaller clusters consisting of targets with similar paths. This new scenario provides an acceptable environment for the testing of some interesting approaches, as one strives for a generalization of the prediction algorithm, able to handle such situations. For that reason, let us now consider the case where there exist multiple points of interest and study the methods used to predict their motion over time.

Whereas previously a single point was chosen to represent the target's position at a certain instance of time, this might not be the most useful way to do it when dealing with groups. Given that no restrictions are made as to the physical relation between targets in the same group, it may be unwise to describe them using only the coordinates of their centre of mass. Instead, a representation of the group using a random variable is considered. This variable possesses a probability that is dependent on the location of every target, exhibiting a bigger value when in close proximity of an element's coordinates and lower value on other points. The prediction now is a stochastic process that works by propagating said random variable X through time, first by applying the motion model and then adding some error to better represent the uncertainty of the estimates.

The propagated term can be computed from the previous time instance value of the random variable as such,

$$X(k+1) = t\big(X(k)\big) + E(k), \tag{4.12}$$

where $t(\cdot)$ represents the effect of motion model on the targets and $E(k)$ is an error term. Since the transformation of a random variable can be considered a new variable itself, an equivalent representation of (4.12) is

$$X(k+1) = Y(k) + E(k). \tag{4.13}$$

4.2.1 Target probability representation

In an ideal scenario, the position of every target would be known without any uncertainty, and thus, the p.d.f of a group with n elements would be given by a sequence of dirac delta functions, scaled as to not exceed the limit value of one when computing the cumulative density function, located at the coordinates of each target,

$$f(x) = \frac{1}{n}\sum_{i=1}^{n}\delta(x - x_i). \tag{4.14}$$

However, in practice, due to uncertainties inherent to these positions and their extraction, a different function must be considered if an accurate representation of the group density is to be had. The author

chose to use a sum of normally distributed curves with mean x_i and variance σ_e^2, since it seemed like an intuitive way to show the expected target coordinates as well as possessed an elegant level of simplicity. It should be noted that the parameter σ_e is problem dependent and acceptable values for it must be found.

This sum, known as mixture distribution, results in a non-negative function that integrates to 1, and can be expressed as

$$f(x) = \sum_{i=1}^{n} w_i p_i(x), \tag{4.15}$$

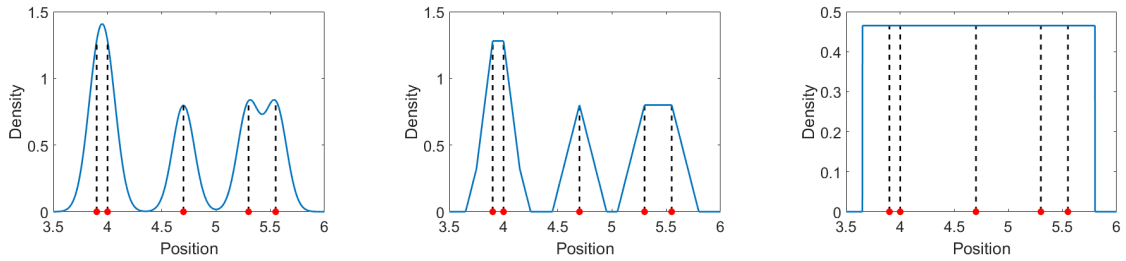
where w_i denotes the weight associated to the i-th density.

Since there is no difference in the importance of a target, the weights all have the same value and the p.d.f. of the targets at the epoch k turns into

$$f_X(x, k) = \frac{1}{n} \sum_{i=1}^{n} p(x_i). \tag{4.16}$$

Other possible functions include an uniform density distribution over an interval that encompasses all of the targets or the convolution of the original function described in (4.14) with a function representing the uncertainty error of each target.

A few examples of possible group densities are shown in the figure below.



(a) Normal mixture distribution. (b) Triangular mixture distribution. (c) Uniform density distribution.

Figure 4.2: Possible group density functions according to the normal (a), triangular (b) and uniform (c) distributions.

In general, the equation of the p.d.f. of X is not available. Instead, a discretized version of the function is used. This discretization contains the value of the function in every point of a grid of equispaced points in an area of interest and is assumed to be irrelevant outside of this area. This assumption exists solely for the purpose of reducing the computational burden of the algorithm, although the region of interest may be expanded if necessary.

4.2.2 Time propagation

The forecasting of the target group density is done through a process that produces a result equivalent to that of the underlying system model (figure 2.1). This procedure can be split in two main operations, a random variable transformation and a random variable sum. These two phases correspond to the application of the motion model and the addition of the prediction uncertainty, respectively.

First, to simulate the continuation of the target motions, a transformation is applied on the input

density X to create a new variable Y. This transformation is assumed to be reversible and differentiable. Furthermore, since both the X and Y are random variables, each of them has a cumulative distribution function (c.d.f.) and density, respectively $F_X(x)$, $f_X(x)$ and $F_Y(y)$, $f_Y(y)$. These c.d.f. are related to one another by

$$F_Y(y) = P(Y \leq y) = P(t(X) \leq y) = P(X \leq t^{-1}(y)) = F_X(t^{-1}(y)). \tag{4.17}$$

Moreover, the density of the transformed random variable Y can be computed with the density of X. From the definition of c.d.f.comes that

$$F_Y(y) = \int_{-\infty}^{y} f_Y(u)du, \tag{4.18}$$

and consequently

$$f_Y(y) = \frac{d}{dy}F_Y(y). \tag{4.19}$$

Applying (4.17) on (4.19) and differentiating with respect to y, the equation for the density takes the form

$$f_Y(y) = f_X(t^{-1}(y))\left|\frac{d}{dy}t^{-1}(y)\right|. \tag{4.20}$$

Although elegant on paper, this process is not as straightforward as it seems. This is due to the fact that only knowing the discrete version of the function requires the computation of (4.20) to be done numerically, as opposed to the direct analytic solution one obtains with the original function. Furthermore, the non-malleable grid makes it so that a transformed point that belonged to the grid on a past epoch is very unlikely to land on one of the grid points in the future epochs. As an example one iteration of the transformation $t(X) = 1.25X$ is shown in the graphic below, where the points belonging to the grid are marked with a black circle.

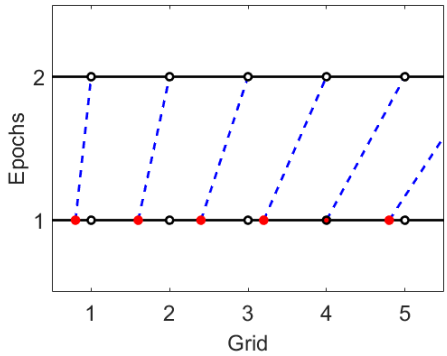


Figure 4.3: One iteration of the transformation $t(X) = 1.25X$.

The one-dimensional grid, with unitary spacing, remains the same as the signal is transformed. Consequently, only the value of the density of Y at $y = 5$ is known since it originated from the coordinate $x = 4$. For all the other points in the image, their reverse transformation (red circles) falls between known grid points. To obtain the values of the density in these situations, an interpolated function is used in place of the original one.

As for the second operation, the p.d.f. of a random variable X that is the result of the sum of two independent random variables Y and E, is given by the convolution between the p.d.f. of the latter ones

$$f_X(x) = \int_{-\infty}^{\infty} f_Y(e) f_E(x-e) de = \int_{-\infty}^{\infty} f_Y(x-y) f_E(y) dy, \tag{4.21}$$

or, analogously for discrete random variables, the distribution function is

$$P(X = x) = \sum_{n=-\infty}^{\infty} P(Y = n) P(E = x - n). \tag{4.22}$$

The time propagation of the density through the repeated execution of (4.20) and (4.22) is the application of the Fokker-Planck equation to the motion of targets. Whereas the original equation describes the time evolution of a particle velocity p.d.f when under the influence of drag forces and random forces, this adaptation uses the shift and diffusion terms in an attempt to predict the future coordinates of a group of objects. This shift results in an extension or compression of the distribution, resultant of a translation of the entire space by a factor relative to the original position value. The diffusion term is simulated through the convolution between this shifted distribution and an error density. Altogether, these two operations mold the signal according to the expected behavior of the model. A single iteration of this method is portrayed in figure 4.4.

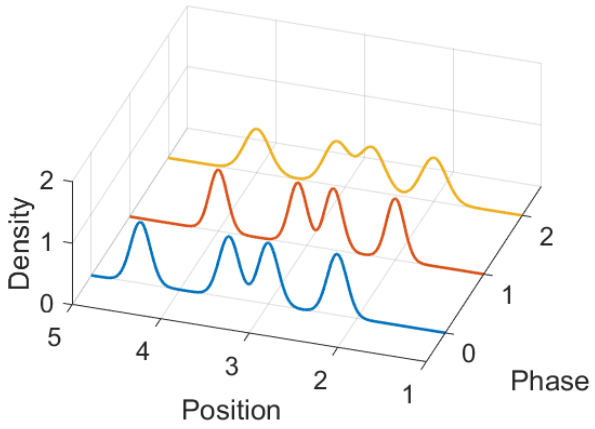


Figure 4.4: One-dimensional group density propagation process. The original signal (blue) is transformed (orange) and then some uncertainty is added (yellow).

This approach proved inefficient when tested with the actual motion models for two reasons. First, due to the occasional multiple initial conditions the resulting transformation presents itself bearing more targets than there are. This effect arises in situations where the targets move according to a model that requires more than one variable transformation. Consequently, the convolution between these shifted density functions creates areas correspondent to the motion of an imaginary target whose positions are a mixture of data from different sources.

In the graphic below, an example of this behaviour is shown. Applying a motion model that provides a linear movement on the consecutive densities of two targets creates two additional nodes based on the combination of the target positions.

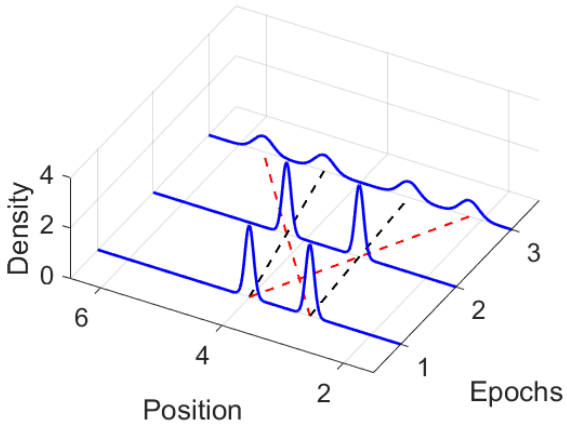


Figure 4.5: Erroneous shifting of the desired variables. In this picture are portrayed the correct translations in black dashed lines and the incorrect translations in red dashed lines.

These undesirable areas can be avoided by splitting the density map into several sections containing each a single area of interest. These areas can hold one or more targets depending on their proximity and associated uncertainty.

The second reason for the inefficiency of this method is the exponential diffusion of the densities. During the shifting phase of the process, there is a change in the width of the function, derived from the magnitude of the motion model coefficients. Since the models considered in this report always possess parameters with magnitude of at least one unit, a stretching of the areas of interest is bound to happen even without the addition of supplementary uncertainty. This effect is worse for the integral version of the H model, with the nodes expanding rapidly independently of the accuracy of the data. This tremendously limits the number of reliable predictions the algorithm can produce.

Note that both of these problems emerge during the first operation of the method. To counter these effects and improve the competence of predictor, the motion of the targets will be done separately for each target and will take as input a single point for every area of interest. The motion model will remain the same throughout the prediction step.

4.3 Data interpolation

Here, a study is conducted on few of the many existing interpolation methods. A brief description of each method is done before evaluating their performance. This evaluation will be done separately for one-dimensional interpolation and two-dimensional interpolation. The estimated points will have a forced minimum value of zero since the goal function is a density function. Also, the piece-wise constant interpolation method, also known as the nearest-neighbor interpolant, that attributes the data value of the nearest known point to the estimate, does not provide a smooth transition between two consecutive points and therefore will not be considered.

4.3.1 One-dimensional interpolation

This section deals with lower dimensional interpolation methods, for when a point in the coordinate system is given by $\big(x, f(x)\big)$.

Linear interpolation

Perhaps the simplest form of interpolation, second only to the piece-wise constant interpolation, this method takes two known points, with coordinates $\big(a, f(a)\big)$ and $\big(b, f(b)\big)$, and returns a value for the desired input x according to the equation of the straight line that intersects both these points,

$$f(x) = f(a) + (x - a)\frac{f(b) - f(a)}{b - a}.$$

While this method boasts an elevated relief in computation resources, it delivers an overall non-smooth and rough representation of the distribution function. This representation is especially rough around areas with a higher curvature, such as the extremes of a function.

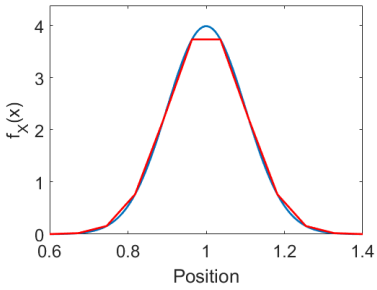


Figure 4.6: Linear interpolation of a normal distribution.

Cubic spline interpolation

Originally used to designate any elastic ruler that was bent to pass through a number of predefined points $\big(x_i, f(x_i)\big)$, also referred to as knots, a spline interpolant refers to a special type of function that is defined piece-wise by polynomials. This technique was employed in the creation of technical drawings for shipbuilding and construction by hand. From the n knots that make up the restrictions of the curve, it is possible to determine a series of polynomials that fit the data in a smooth manner. To do this, the equation of the function at each of the $n - 1$ intervals is computed so that not only the function, but also its first and second derivatives are continuous all throughout the range of input values x. These constraints are only met if at every non-extreme knot x_i both surrounding interpolated functions $g_{i-1}(x)$ and $g_i(x_i)$ are such that

$$\begin{cases} g_{i-1}^{'}(x_i) = g_i^{'}(x_i) \\ g_{i-1}^{''}(x_i) = g_i^{''}(x_i). \end{cases} \tag{4.23}$$

The classical approach is to use cubic splines, made of third degree polynomials, since any polynomial of degree 3 or higher satisfies these conditions.

At each interval it is known that

$$\begin{cases} g_i(x_i) = f(x_i) \\ g_i(x_{i+1}) = f(x_{i+1}) \\ g_i^{'}(x_i) = k_i \\ g_i^{'}(x_{i+1}) = k_{i+1}, \end{cases} \tag{4.24}$$

where k_i stands for the curvature of the function at the point x_i, that by definition is given by

$$k = \frac{f^{''}(x)}{\big(1 + f'^2(x)\big)^{\frac{3}{2}}}. \tag{4.25}$$

A more compact way to describe the function at each part is through the equation

$$g_i(x) = \big(1 - t_i(x)\big)f(x_i) + t_i(x)f(x_{i+1}) + t_i(x)\big(1 - t_i(x)\big)\Big(a_i\big(1 - t_i(x)\big) + b_i t_i(x)\Big), \tag{4.26}$$

with

$$t_i(x) = \frac{x - x_i}{x_{i+1} - x_i}, \tag{4.27}$$

$$a_i = k_i(x_{i+1} - x_i) - \big(f(x_{i+1}) - f(x_i)\big) \tag{4.28}$$

and

$$b_i = -k_{i+1}(x_{i+1} - x_i) + \big(f(x_{i+1}) - f(x_i)\big). \tag{4.29}$$

From (4.26) it is possible to obtain the equations for the first and second derivatives of the interpolant

$$g_i^{'}(x) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} + \big(1 - 2t_i(x)\big)\frac{a_i\big(1 - t_i(x)\big) + b_i t_i(x)}{x_{i+1} - x_i} + t_i(x)\big(t_i(x)\big)\frac{b_i - a_i}{x_{i+1} - x_i} \tag{4.30}$$

and

$$g_i^{''}(x) = 2\frac{b_i - 2a_i + (a_i - b_i)3t_i(x)}{(x_{i+1} - x_i)^2}. \tag{4.31}$$

Furthermore, if the function is to have a continuous second derivative then, according to (4.28),(4.29) and (4.31), the equality

$$\frac{k_{i-1}}{x_i - x_{i-1}} + \left(\frac{1}{x_i - x_{i-1}} + \frac{1}{x_{i+1} - x_i}\right)2k_i + \frac{k_{i+1}}{x_{i+1} - x_i} = 3\left(\frac{f(x_i) -}{(x_i - x_{i-1})^2} + \frac{f(x_{i+1}) - f(x_i)}{(x_{i+1} - x_i)^2}\right) \tag{4.32}$$

must be assured.

As it stands, (4.32) gives us $n-2$ equations that define the curvature at each of the middle knots. The two remaining equations dictate the behaviour of the function at the end points, such as the "Natural splines" equations

$$\begin{cases} \frac{2}{x_2 - x_1}k_1 + \frac{1}{x_2 - x_1}k_2 = 3\frac{f(x_2) - f(x_1)}{(x_2 - x_1)^2} \\ \frac{1}{x_n - x_{n-1}}k_{n-1} + \frac{2}{x_n - x_{n-1}}k_n = 3\frac{f(x_n) - f(x_{n-1})}{(x_n - x_{n-1})^2}, \end{cases} \tag{4.33}$$

that specify that the second derivative be zero in both extremities. Other end conditions include the "Clamped spline", that forces the slope at the ends to be the desired value, or the "Not-a-knot spline" where the third derivative of the interpolant is continuous at the x_2 and x_{n-1} points.

With this method, the resulting interpolation function incurs a smaller error and is smoother than linear interpolation. However, due to the global nature of the basis functions that make the spline function, the problem becomes ill-conditioned, i.e. a small change in input can lead to some big change in output, especially on the edges of the function. This effect is referred to as the Runge's phenomenon and, fortunately, can be completely mitigated by using splines of compact support that force the function to become zero outside some interval or other similar strategies.

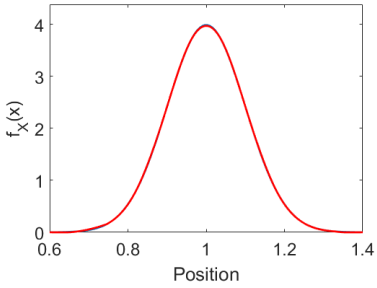


Figure 4.7: Spline interpolation of a normal distribution.

Monotone cubic Hermite interpolation

This interpolant results from modifying the cubic Hermite interpolation method, ensuring the monotonicity of the interpolation function. As with the previous approach, it uses a piece-wise third-degree polynomial spline to estimate the desired function. The difference is that the polynomial is specified in Hermite form, i.e., by its values and first derivatives at the end points of the corresponding domain interval. The interpolant exhibits an especially close approximation on functions whose value remains mostly constant, albeit being almost as rough as the linear interpolant around the extremes of the original function.

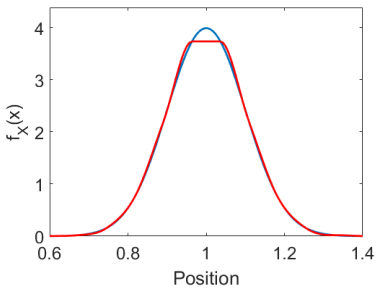


Figure 4.8: Monotone cubic Hermite interpolation of a normal distribution.

Lagrangian Interpolation

Another polynomial based interpolation function is the Lagrange interpolant. In this approach, given a set of points $\big(x_j, f(x_j)\big)$ with no two equal x_j the lowest degree polynomial function that fits the data points is obtained. This method produces an interpolation error proportional to the distance between the data

points to the power of n, with n the number of data points used to compute the polynomial. However, it is not wise to choose this solution when dealing with functions comprised of a large amount of points, since it becomes computationally more expensive to find the interpolation much faster than the other options. To reduce the error as much as possible while still maintaining a relatively low computation time, a Lagrangian piece-wise interpolant is sought. This modified version computes the interpolated function with a moving set of data points smaller than the original set. For relatively low degree polynomials, this method produces an error with acceptable magnitude while maintaining an adequate computation time. Its main disadvantage is that, similarly to the spline interpolant, this function also suffers from Runge's phenomenon. If left untouched, this interpolation possesses the closest approximation, out of all the considered methods, in the middle points and the farthest in the end points. This effect is mitigated with the usage of lower order Lagrangian interpolants near the extremes of the data points.

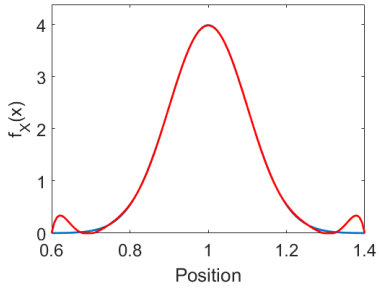


Figure 4.9: Moving Lagrangian interpolation with order 9 of a normal distribution.

Results

From the four methods studied, two classes are defined. Both the linear and monotone cubic interpolants outperform the other two in specific situations. These excel whenever the original function is mostly flat, as happens with the uniformly distributed density. For more general curves, however, the Spline and the Lagrangian interpolants are preferable. Since the chosen density takes the form of a mixture of Gaussian curves, it is only fitting that the latter class is favoured over the former.

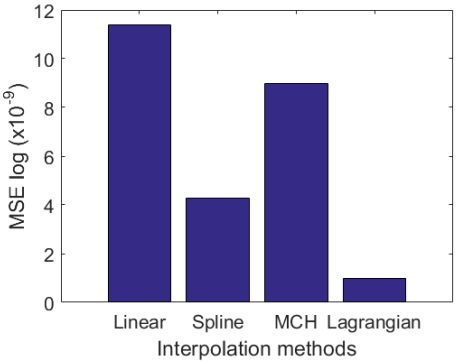


Figure 4.10: Interpolation mean squared error with varying methods.

From the bar graph above one can perceive the disparity in performance between the studied methods that confirms the previous hypothesis. There is a considerable gap in the error measurements separating both classes. Furthermore, even within the second group there is a substantial difference between the

best and worst interpolants. The obvious choice for the interpolating function in one dimension is then the Lagrangian interpolator.

4.3.2 Two-dimensional interpolation

Here some interpolation methods for functions containing two input dimensions are described. For simplicity, every time a single interpolated space is shown a generic representation of said space, the unit square, is used.

Bilinear interpolation

This first method is an extension of the linear interpolation for functions spanning two axis. Like the rest of the presented methods, it works by executing the necessary interpolations first on one direction and then again on the remaining one. It should be noted that even though the interpolant performs two linear steps, hence the name, the interpolation as a whole is quadratic.

This process is the simplest one yielding a continuous function in the interpolated space. It requires just the values of the function at the corners of the desired space. For any point (x, y) inside the interpolated area, given by its limiting intervals $[x_0, x_1]$ and $[y_0, y_1]$, its attributed value is computed through the linear interpolation of its limits over the first axis,

$$\begin{cases} f(x, y_0) = \frac{x_1 - x}{x_1 - x_0} f(x_0, y_0) + \frac{x - x_0}{x_1 - x_0} f(x_1, y_0) \\ f(x, y_1) = \frac{x_1 - x}{x_1 - x_0} f(x_0, y_1) + \frac{x - x_0}{x_1 - x_0} f(x_1, y_1) \end{cases},$$

followed by the linear interpolation between these newfound points over the second axis

$$f(x, y) = \frac{y_1 - y}{y_1 - y_0} f(x, y_1) + \frac{y - y_0}{y_1 - y_0} f(x, y_1).$$

This process is illustrated in the figure below, with the original known points in red, the partial interpolated points in blue and the end result in green.

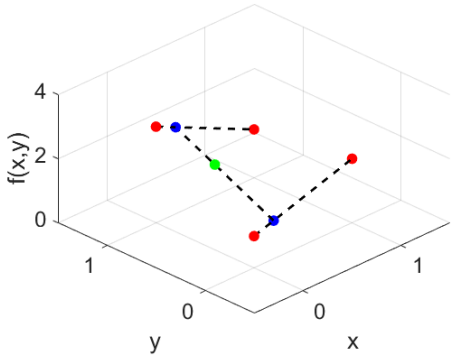


Figure 4.11: Bilinear interpolation of a single point.

There are a few alternative ways to describe this problem. One can define the desired point in terms

of its coordinates

$$f(x, y) \approx a_0 + a_1 x + a_2 y + a_3 xy,$$

with the a coefficients the solution for the linear system

$$\begin{bmatrix} 1 & x_0 & y_0 & x_0 y_0 \\ 1 & x_0 & y_1 & x_0 y_1 \\ 1 & x_1 & y_0 & x_1 y_0 \\ 1 & x_1 & y_1 & x_1 y_1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} f(x_0, y_0) \\ f(x_0, y_1) \\ f(x_1, y_0) \\ f(x_1, y_1) \end{bmatrix}.$$

One can also write it based on the function limits

$$f(x, y) \approx b_{00} f(x_0, y_0) + b_{01} f(x_0, y_1) + b_{10} f(x_1, y_0) + b_{11} f(x_1, y_1),$$

where the b coefficients are found by solving

$$\begin{bmatrix} b_{00} \\ b_{01} \\ b_{10} \\ b_{11} \end{bmatrix} = \left(\begin{bmatrix} 1 & x_0 & y_0 & x_0 y_0 \\ 1 & x_0 & y_1 & x_0 y_1 \\ 1 & x_1 & y_0 & x_1 y_0 \\ 1 & x_1 & y_1 & x_1 y_1 \end{bmatrix}^{-1} \right)^T \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix}.$$

Additionally, if the area is represented using the unit square, the interpolation function simplifies to

$$f(x, y) \approx f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy,$$

or, equivalently,

$$f(x, y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix}.$$

Bicubic interpolation

Like the previous method, this one is an extension of an interpolation process to the dimension above, the cubic interpolant. Just as its lower dimension counterpart, this procedure results in a surface with a smoother transition in and around the known data points, compared to the linear and piece-wise constant interpolation methods. Consequently, if said trait is of significant importance, it is preferable to use this method over the bilinear or nearest-neighbor options. The reason this solution bears a more fluid behaviour stems from the fact that it takes sixteen points when interpolating, opposed to the four necessary points used in the simpler interpolants. These additional twelve points serve to compute the partial and cross derivatives of the four corners surrounding the interpolated area. The partial derivatives, f_x and f_y, are obtained by measuring the slopes at each of the points over the respective axis, i.e. by using finite differences with the neighboring points. The cross derivative f_{xy} is the result of performing a partial derivative on the solution of the other one.

Using these sixteen values, one can determine the essential weights attributed in the interpolation of any given point (x, y). The equation that computes the function value is

$$f(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} x^i y^j. \tag{4.34}$$

Likewise, the expressions for the derivatives are

$$f_x(x,y) = \sum_{i=1}^{3}\sum_{j=0}^{3} a_{ij} i x^{i-1} y^j, \tag{4.35}$$

$$f_y(x,y) = \sum_{i=0}^{3}\sum_{j=1}^{3} a_{ij} x^i j y^{j-1}, \tag{4.36}$$

and

$$f_{xy}(x,y) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij} i x^{i-1} j y^{j-1}. \tag{4.37}$$

Together, these four equations provide the necessary information for the computation of each of the weight values. A concise form for the solution of this system is obtained, in matrix form, through

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) & f_y(0,0) & f_y(0,1) \\ f(1,0) & f(1,1) & f_y(1,0) & f_y(1,1) \\ f_x(0,0) & f_x(0,1) & f_{xy}(0,0) & f_{xy}(0,1) \\ f_x(1,0) & f_x(1,1) & f_{xy}(1,0) & f_{xy}(1,1) \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

Once the weights are determined, any point inside that area can be interpolated using

$$f(x,y) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ y \\ y^2 \\ y^3 \end{bmatrix}.$$

Lanczos interpolation

This last interpolation method, also named Lanczos filtering or Lanczos resampling after its inventor Cornelius Lanczos, uses a sinc based kernel (Lanczos kernel) to smoothly interpolate the value of a digital signal between its samples. This reconstrucion kernel $L(\cdot)$ is the normalized sinc function, windowed by the central lobe of a horizontally stretched sinc function

$$L(x) = \begin{cases} \text{sinc}(x)\text{sinc}(\frac{x}{a}) & \text{for} - a < x < a \\ 0, \text{otherwise.} \end{cases} \tag{4.38}$$

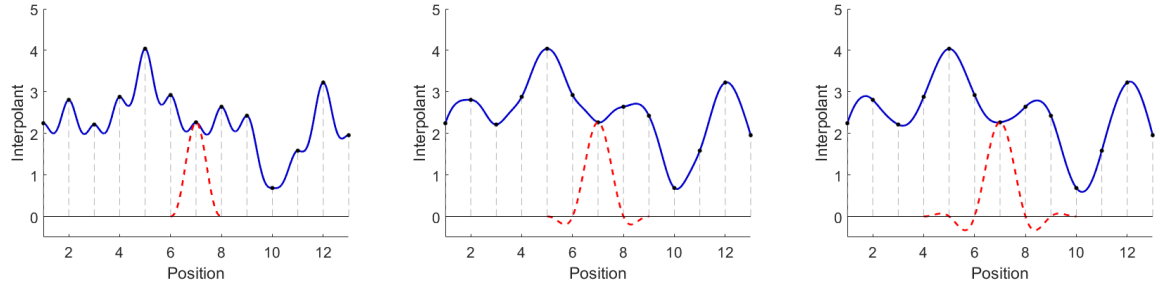
The size of the kernel is determined by a, which is a positive integer typically taking the value 2 or 3. Restricting the parameter to positive integers yields an interpolated signal with the guarantee of being

continuous and having a continuous derivative.

Once the kernel is defined, one can obtain the equation for the one-dimensional interpolant

$$f(x) = \sum_{i=\lfloor x \rfloor - a + 1}^{\lfloor x \rfloor + a} f(x_i) L(x - i), \tag{4.39}$$

with $\lfloor \cdot \rfloor$ the floor function that rounds the input to the nearest integer with a lower value than its own. As the value of a grows, so does the area of influence of each lobe. This area is the interval where the filter is active. For a determined parameter value, the range of a single filter is given by $2a - 1$ with the center at x_i. Having a bigger region of impact creates a smoother output at the cost of producing an oscillating behaviour. Furthermore, these oscillations can generate estimates with a negative value, so special care must be taken to ensure no unacceptable interpolations appear in the final product.



(a) Lanczos interpolant with $a = 1$. (b) Lanczos interpolant with $a = 2$. (c) Lanczos interpolant with $a = 3$.

Figure 4.12: Depiction of the interpolated function (blue) from the original points (black dots) with different sized nodes (red).

The generalization of (4.39) to a higher dimension consists in summing over all the possible combinations of the filter. Specifically, the equation for the interpolation function in two dimensions is

$$f(x, y) = \sum_{i=\lfloor x \rfloor - a + 1}^{\lfloor x \rfloor + a} \sum_{j=\lfloor y \rfloor - a + 1}^{\lfloor y \rfloor + a} f(x_i, y_j) L(x - i, y - j). \tag{4.40}$$

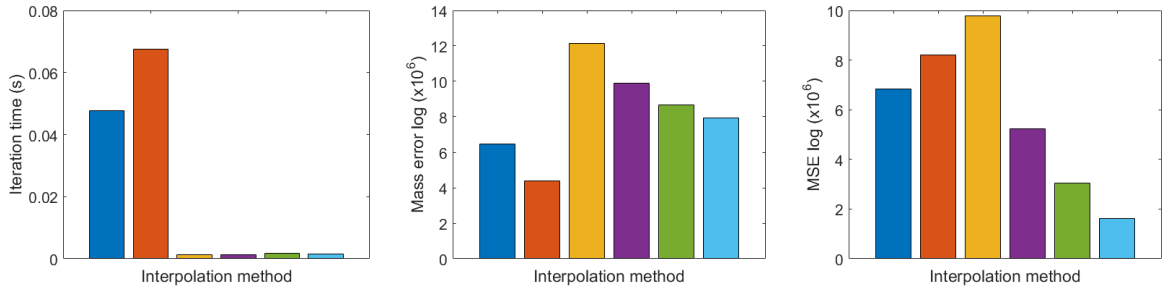
Fortunately, the multivariate Lanczos filter's kernel is separable, i.e. it can be written as the product of its parts. As a consequence, the bidimensional interpolant simplifies to a series of sums

$$f(x, y) = \sum_{i=\lfloor x \rfloor - a + 1}^{\lfloor x \rfloor + a} L(x - i) \left(\sum_{j=\lfloor y \rfloor - a + 1}^{\lfloor y \rfloor + a} f(x_i, y_j) L(y - j) \right). \tag{4.41}$$

Results

The performance of these three functions was tested in regards to three distinct aspects: computation time, mass error and mean squared error. This information will allow for the choice of the most convenient method. The goal is the attainment of an interpolating process that provides an adequate approximation within an acceptable time interval, and whose output density is not far from the input. Below, a series of bar charts depict the results of the conducted study. From left to right the interpolation methods are: Bilinear, Bicubic and Lanczos with the parameter a set to 1, 2, 3 and 4, respectively. Though the bilinear

and bicubic solutions outperformed the Lanczos interpolant in mass conservation, the latter surpassed them both in computation speed and greatly in the mean squared error.



(a) Computation time of one interpolation iteration.

(b) Absolute difference in mass error in natural logarithm base.

(c) Mean squared error in natural logarithm base.

Figure 4.13: Bar charts of the different studied aspects of the interpolants.

Whereas the nature of the Lanczos filter makes it more prone to higher errors of the outcome mass, it provides a faster more accurate estimate of the whole function. The thinnest filter's response is too rough to justify its usage, however, wider filters proved capable of handling the problem with an acceptable performance. While it obtained better results than its counterparts with lower parameter values, the oscillations produced by the broadest Lanczos filter tested ($a = 4$) in undesirable areas, namely areas with low density magnitude, were both higher in amplitude and numbers. This effect is aggravated in ampler filters. The author considered the Lanczos interpolant with $a = 3$ to be the best choice out of all the options.

Chapter 5

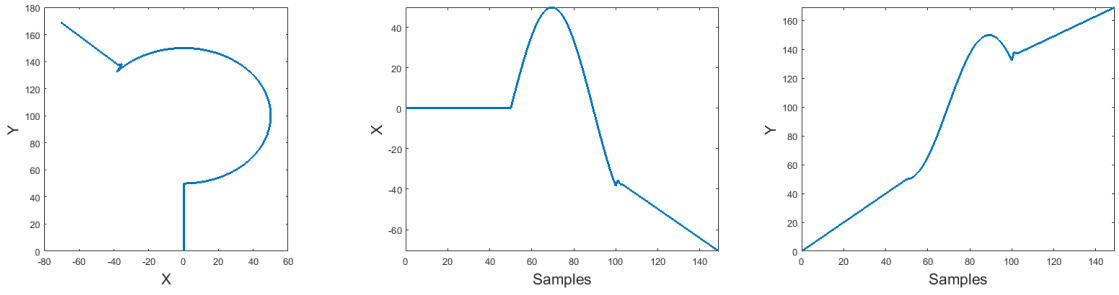
Results

This chapter is dedicated to the review and explanation of the results of important tests, conducted in an attempt to understand and evaluate the performance of the constructed algorithm.

5.1 Non-stationary signal estimation

The solution suggested in the second section of Chapter 3 was tested in a noiseless case scenario. The chosen trajectory was the one shown in figure 2.7. The set of H models used in the estimation consisted of three different models in total, namely a second and third order integrators, H_i^2 and H_i^3 respectively, and a periodic model with the period of the cosine used in the generation of the movement related to the roundabout itself, H_p^{50}.

The estimated trajectory was similar to the actual one, with the only perceivable difference being in the transition from the circular to the linear motions, as depicted in figure 5.2.



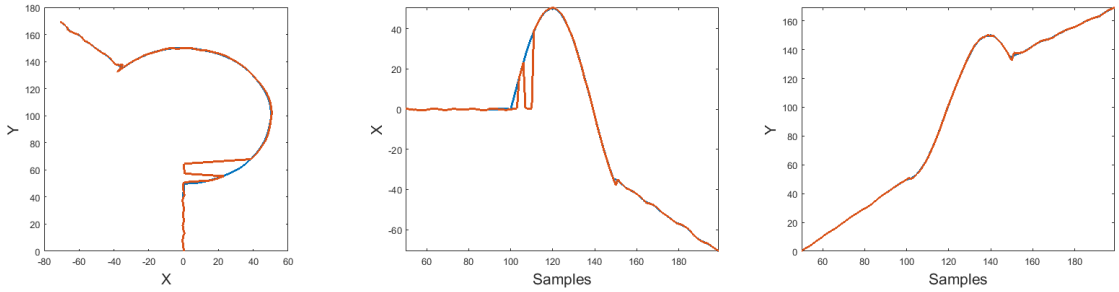
(a) Estimated motion of a vehicle in a roundabout.

(b) Estimated motion along the first coordinate.

(c) Estimated motion along the second coordinate.

Figure 5.1: Bi-dimensional estimate (a) and its one-dimensional counterparts (b) and (c) in a noiseless environment.

When some noise was added, however, the estimator did not behave so well. While it maintained the oscillatory behaviour observed in the previous results, the algorithm chose the periodic model to represent certain parts of the circular motion which, due to the initial conditions present, resulted in a poor representation of the motion in those points.



(a) Estimated motion of a vehicle in a roundabout.

(b) Estimated motion along the first coordinate.

(c) Estimated motion along the second coordinate.

Figure 5.2: Estimate of a bi-dimensional motion (orange) in the presence of noise and original motion (blue).

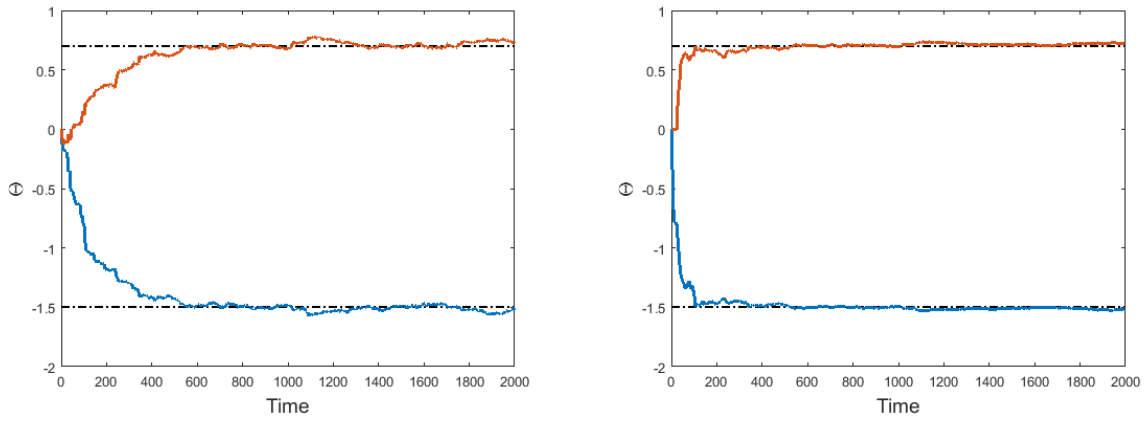
A few solutions have been thought of to solve the problem at hand, such as the introduction of models with static initial conditions, or the filtering of the signal prior to the estimation of the motions so as to smoothen it and attenuate the effect of the noise on the initial conditions. Further work is in order before advancing to tests using real data.

5.2 Stationary signal estimation

The performance of both the RLS and the RW-LASSO algorithms were tested in situations where the parameters were static and variable. In the first situation, the AR model was given by

$$\frac{C}{A} = \frac{1}{1 - 1.5z^{-1} + 0.7z^{-2}}, \tag{5.1}$$

with the parameters to be estimated being $\theta = [1.5 \quad 0.7]^T$. Supplying the algorithms with the correct order of the polynomial results in a good estimation of the parameters in both cases, with the RLS having a slower convergence and a bigger variation than the RW-LASSO, as can be seen in the figure 5.3 below. The RLS method has a fixed forgetting factor in the following cases, unless otherwise stated.

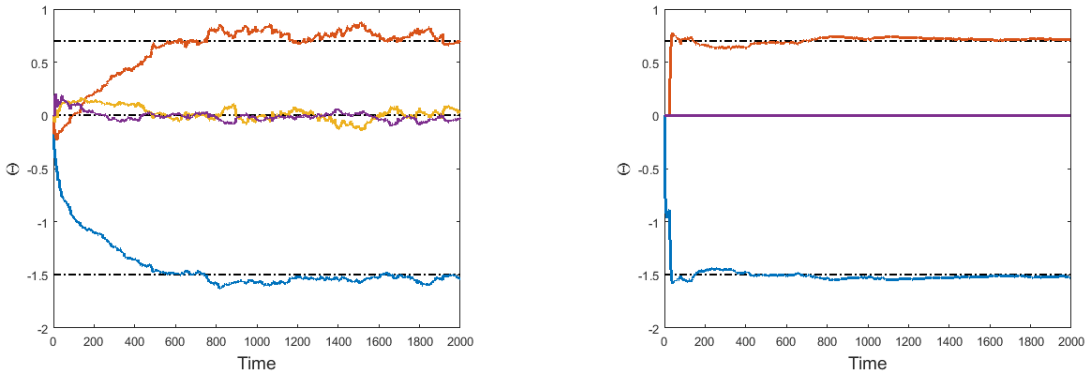


(a) RWL estimation.

(b) RW-LASSO estimation.

Figure 5.3: Estimation of parameters given the correct order of the polynomial.

Similar results were obtained when the input order was increased. The polynomial in this case was of order 4, two orders above the actual polynomials order. In figure 5.4 the main difference between these two methods can be perceived: their responses sparsity. While the LASSO correctly nullifies the extra parameters added, the RLS constantly varies their values around zero, never actually giving them a null value.



(a) RWL estimation.

(b) RW-LASSO estimation.

Figure 5.4: Estimation of parameters given a value superior to the actual order of the polynomial.

5.3 Target motion data

Once built and fine-tuned, the prediction algorithm was tested with real time-series taken from the Stanford Aerial Pedestrian Dataset [29]. This Dataset comprises various videos of different classes of targets, such as pedestrians, bikers, skateboarders, cars, buses, and golf carts in a variety of scenes located somewhere on the Stanford University campus. The Dataset is public and accessible at `https://cs.stanford.edu/~anenberg/uav_data/`. Along with the correspondent video showcasing the distinct target trajectories, an annotation file containing all the relevant information is available for every scenario. Inside this file are as many lines as there are observations of targets, each associated to the identification of an object in a specific frame. The structure of an annotation consists of the coordinate limits of the bounding box surrounding the object, namely the top-left and bottom-right points, the frame at which it was detected, a series of flags describing some features of the data, specifically if the current observation was outside of the field of view, occluded or automatically interpolated, and finally, the label of the target. The footage is obtained from a 4K camera attached to a quadcopter platform hovering directly above the scene, so there is no need to perform an homography on the data.

Although the files contain a multitude of acceptable time-series, not all of them are adequate for motion prediction. Due to the automatic nature of the object classification, there exist some incomplete sequences of observations and even cases of different trajectories being fused "end-to-start", resulting in drastic changes in coordinates that can deteriorate the performance of any study conducted on said data. It is only reasonable that a certain amount of data is compromised since, given the scale of the dataset, classifying all of the objects by hand would be, at the very least, considerably impractical. To reduce the number of plausible choices, a validation step is executed, filtering out every undesirable curve. The

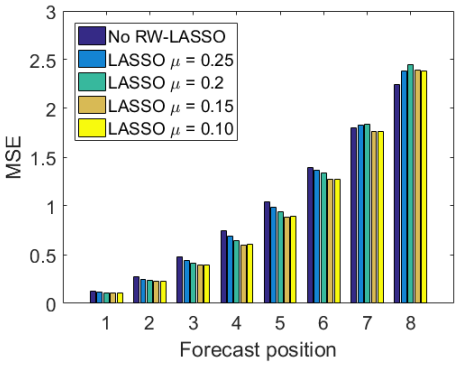
author thought it best to specify as acceptable criteria that each of the valid paths started and ended near the edges of the map and passed through a region around the center. This successfully eliminates any trajectory that inexplicably appears or ends in the middle of the map, and other undesired ones.

Another adverse aspect of the dataset is the error that arises from the extraction of the data in a finite state space. The fact that the target is portrayed in the pixels of an image deteriorates the accuracy of the observation due to the quantization effect. Since the silhouette of an object in a frame affects the overall shape of its associated bounding box, this discretization produces rougher transitions between consecutive points by rounding the corners to integer form. The object is described using the center of the rectangle encompassing it, allowing the coordinates to take values ending in ".5", resultant of being between two consecutive pixels. Even though this decision effectively doubles the available state space, which was already reasonably big to start with, at a resolution of 1400 by 1904, the time-series are still too rough to work with. A low-pass filter, similar to the one described in (3.3), is applied on the data to smooth it out. As a consequence of using such a filter, a delay on the positions of the target is observed. To avoid extending the delay too much and hoping to simulate as accurately as possible a real-time scenario, the order of the filter was kept to two.

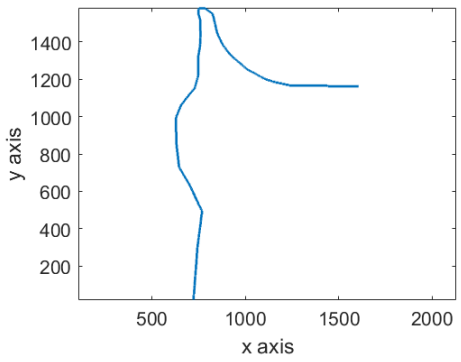
5.4 Single target forecasting

Despite calibrating beforehand, the chosen parameters resulted in estimates too sparse to influence the outcome of the system. In most cases, the AR models would only be active for less than ten percent of the whole path, usually peaking near the beginning and suddenly collapsing to a null estimate. This resulted in a predictor that relied almost always solely on the general motion of the target (H model). To counter this effect, the lower limit of the penalization function was lowered from 3.1 to allow smaller values of the coefficients to be considered. Three tests were conducted to ascertain the best value for the model estimation in this dataset time-series. The distinction between the performance of the methods with different coefficients is obtained through the comparison of their mean squared errors. The values used in these tests serve more as a guideline to the study of the effect of the parameters than a specific selection to obtain the best possible forecast. It is beneficial to select a value that provides adequate predictions for a larger number of targets over one that favours a specific type of paths.

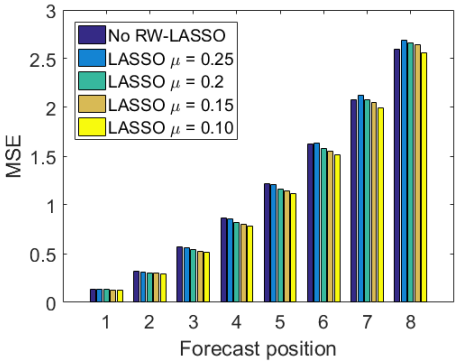
Figure 5.5 shows the results of the variation of μ_k on the chosen path predictions and the respective paths that generated those results. On the left a series of bars are displayed in groups. Each group corresponds to the mean squared error of the predictions at a determined number of epochs ahead. For each prediction step, the comparison between the various models with different AR estimation parameter λ and the simplified system using only the motion model is displayed in the bars with varying colors. On the right are depicted the respective trajectories that originated the graph bar on the same line.



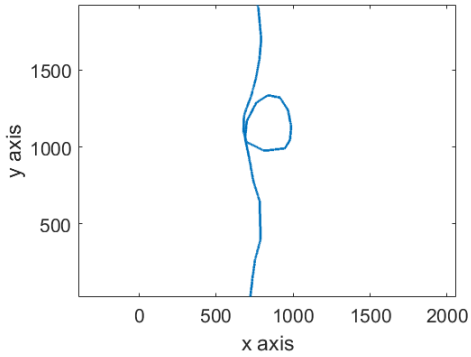
(a) First target forecasting MSE.



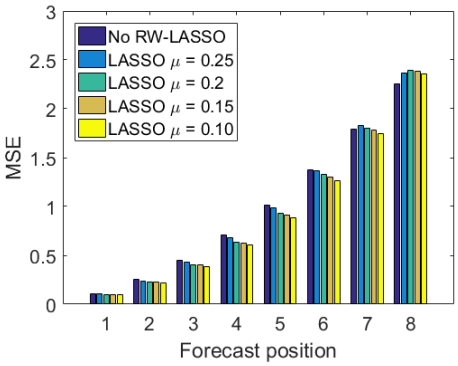
(b) Path travelled by the first target.



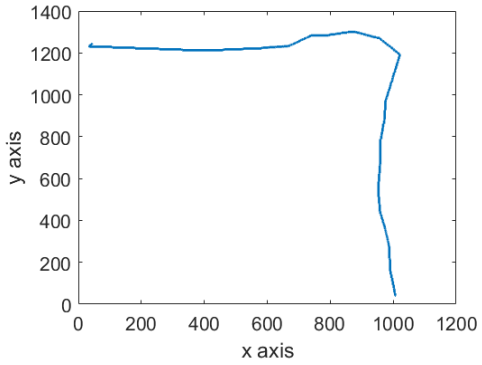
(c) Second target forecasting MSE.



(d) Path travelled by the second target.



(e) Third target forecasting MSE.



(f) Path travelled by the third target.

Figure 5.5: Mean squared error evolution of the first eight prediction steps.

The addition of the RW-LASSO algorithm in the forecasting process provides a positive impact on most cases up to predictions six epochs ahead. From there on, only a few configurations improve the MSE of the predictor. The decrease in magnitude of μ generally yields a more accurate guess, regardless of the number of forecast steps. However, the estimates produced with values below 0.15 tend to have little sparsity more often than not. Because of this fact, λ was adjusted so that $\mu_k = 0.15$.

Further tests were conducted with the modified parameter value to understand more extensively the impact of the stationary model estimator on the system. The chosen path (figure 5.6) was similar to the one travelled by the third target from the previous case study, both in the start and end coordinates, as well as in the overall shape of the trajectory.

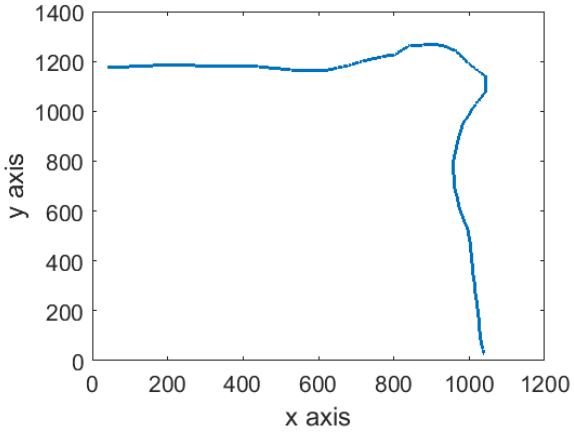
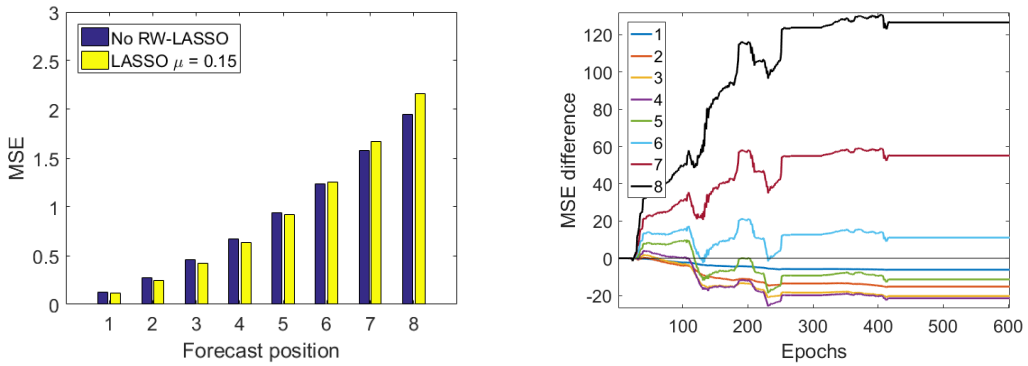


Figure 5.6: Path traveled by the fourth studied target.

The effect of the auto-regressive signal is however, not as positive as it was for the last test case, with the mean squared error of the predictions with the incorporation of the stationary signal input surpassing that of the estimates using solely the motion model from 6 forecasting steps forward.

Figure 5.7 (b) shows a more detailed perspective of the difference between these two predictions. Each of the colored signals represents the evolution of a particular forecast epoch MSE difference obtained by subtracting the predicted coordinates with the AR model aid to the more general predictions obtained with just the H block. While the behaviour of the errors is similar for every projection level, the amount of variation they display is much more accentuated in farther estimates.

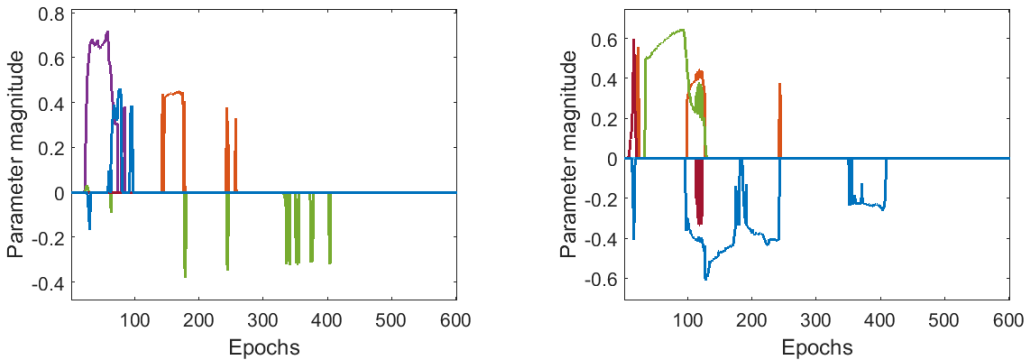


(a) Total MSE sum.

(b) MSE evolution in time.

Figure 5.7: Performance test on the impact of the AR estimator usage.

A close inspection of the correspondent AR parameter evolution, shown below, reveals that these abrupt changes in values result from an incorrectly estimated model. Usually, an overfit of the stationary model deteriorates the prediction more than an underfit. Note that the images illustrating the evolution of the AR coefficients all follow the same color pattern as the one displayed in 5.7 (b), with higher numbers referring to y values farther in the past.



(a) Coefficients pertaining to the xx axis.

(b) Coefficients pertaining to the yy axis.

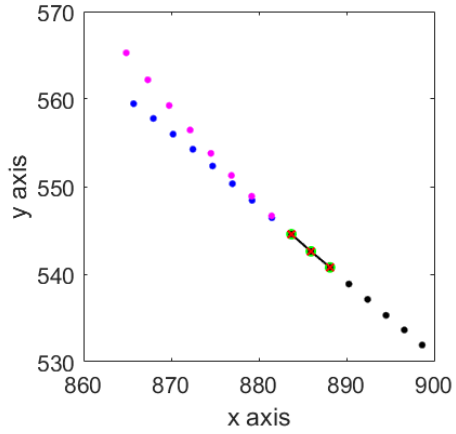
Figure 5.8: AR model estimated parameters.

Even with the reduction in the magnitude of λ, the coefficients that generate the stationary signals are all zero during a considerable part of the trajectory. In figure 5.8 (a) a somewhat extreme case is depicted, with the parameters possessing null value for more than half the study time, apart from the occasional sudden peaks. These peaks arise whenever the estimator encounters a possible sparse representation that cannot sustain itself for very long, resulting in an abrupt change in value from zero to non-zero and then, almost instantly, back to zero again. Albeit somewhat inconvenient, this option is not as harmful as the overfitting of the model that would come as a consequence of further lowering the limit of the penalization function.

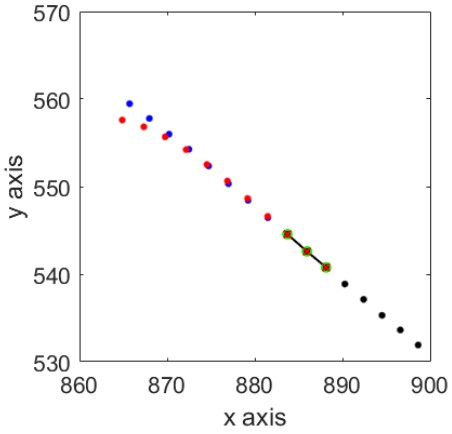
As informative as these graphics are, it is not as intuitive as examining the behaviour of the predictor in real time. Additional insight can be gained by inspecting the impact of the AR model addition to the signal with more context. For that purpose, a few snapshots of the execution of the algorithm are presented and analyzed. In the figures below, the actual positions of the target are depicted in black, for present and past observations, and blue, for future ones. The initial conditions that generate the prediction are coloured red with a green edge and linked with a black line. The predicted points are shown in magenta and red, pertaining to the predictions with and without the stationary input, respectively.

First, a scenario in which the addition of the stationary signal input improves the outcome of the predictor is considered. Here, the slight deviation from the truth obtained with just the motion model prediction is corrected, increasing the accuracy of the first six estimates to almost pinpoint precision and reducing the error of the last two forecast points to magnitudes comparable to that of the fourth and fifth guesses attained with the more general system.

In the second scenario, the algorithm predicts a continuous curve to the right, while the actual path transitions from a curve to a straight line after a few steps. The addition of the y signal reinforces this belief in the continuation of the curve. In fact, not only does this change result in a tighter curve, it also influences the speed at which the target moves, producing estimates that diverge from the actual trajectory much more than the previous, more general ones.

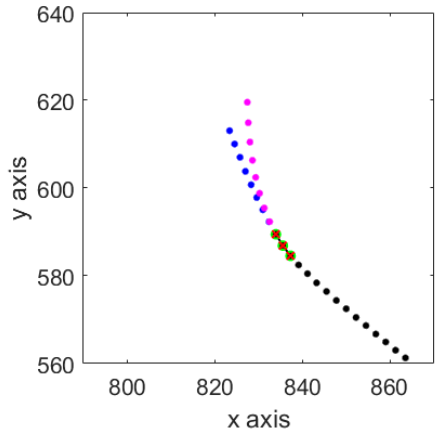


(a) Forecasting without the addition of any stationary input.

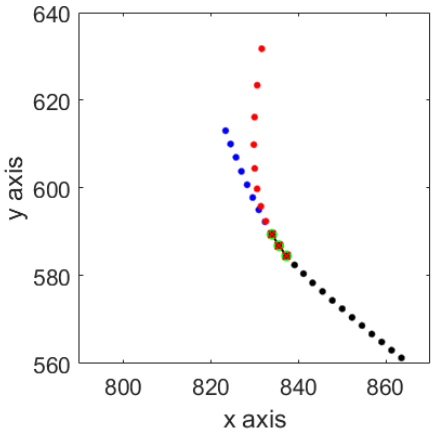


(b) Forecasting with the aid of the auto-regressive signal input.

Figure 5.9: Improvement of the predictions using the AR models estimated with the RW-LASSO algorithm.



(a) Forecasting without the addition of any stationary input.



(b) Forecasting with the aid of the auto-regressive signal input.

Figure 5.10: Deterioration of the predictions using the AR models estimated with the RW-LASSO algorithm.

In both situations, the active motion model is the third order integrator. This model is highly susceptible to changes in the initial conditions and tends to drift towards one side. As such, it is more likely to produce errors with a bigger magnitude than its lower order counterpart.

Further tests were conducted on targets with distinct path shapes and the results were somewhat consistent. While the results of introducing an auto-regressive input in the motion model differ with the chosen trajectories, the effect of the nature of the AR model is constant. Although the sparsity of the estimates plays an important role in the accuracy of the forecasts, it was perceived that higher order AR models tend to generate bigger errors than those containing just the first few coefficients active.

5.4.1 Group forecasting

The generalization of the predictor used above was tested in scenarios with multiple targets whose paths possess a similar outline. A noticeable difference between both the singular and multiple prediction cases

is the increase in computation time when considering more than one target. This gap is due to the expansion in the grid encompassing the targets. This grid needs to be dense enough to describe the targets position with adequate accuracy, however, if the distance between the objects is too big compared to the variance of their related uncertainty, some spots inside this area of interest are bound to have negligible values. In an attempt to reduce the toll imposed by this process, a separation of the region of interest into smaller areas focused on individual targets is considered.

A trio of cyclists was chosen as the targets for the first case study. These time-series displayed a concurrent behaviour, both temporal and spatially, indicating a high probability of belonging to the same group. Figure 5.11 illustrates the paths of the different objects, cropped to the same sample length. From the image it is possible to see some overlaps in coordinates, consistent with the expected behaviour of a set of cyclists travelling the same trajectory. This situation is favorable for testing the performance of the generalized algorithm, since it is assumed that a single motion model is sufficient for an adequate prediction of every target.

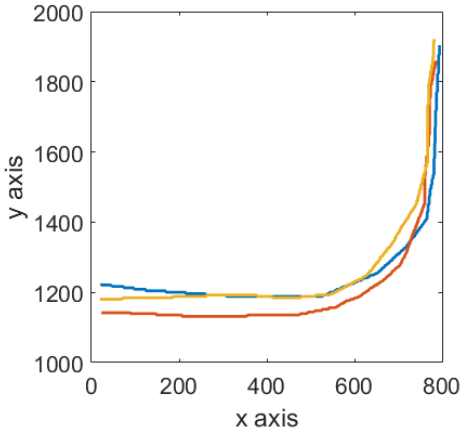
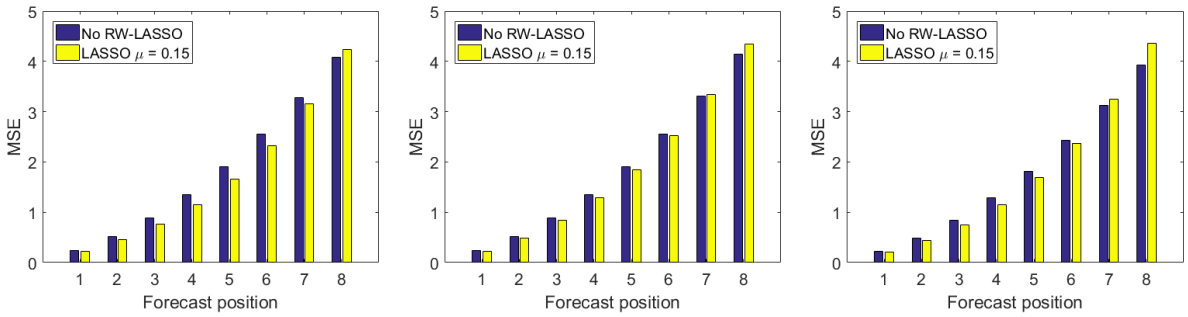


Figure 5.11: Paths travelled by a group of targets. Each distinctly coloured line represents an individual target trajectory.

The results of the test are positive and in accordance to what was previously observed in the single target experiments. The overall performance of the system with the interaction between the stationary and non-stationary models surpasses that of the system with just the motion model, for forecasts up to six epochs ahead. The difference in error is bigger for the time-series from which the motion model compared to the other two, as was anticipated, yet the disparity is not substantial.

The results depicted in figure 5.12 demonstrate that it is possible to simplify the generalization of the predictor for more complex situations by describing the general aspect of the paths with a single common model and obtain beneficial outcomes. It must not be forgotten that the conditions under which this conclusion holds are somewhat restrictive, nevertheless, it encourages the usage of the simplified version of the system.

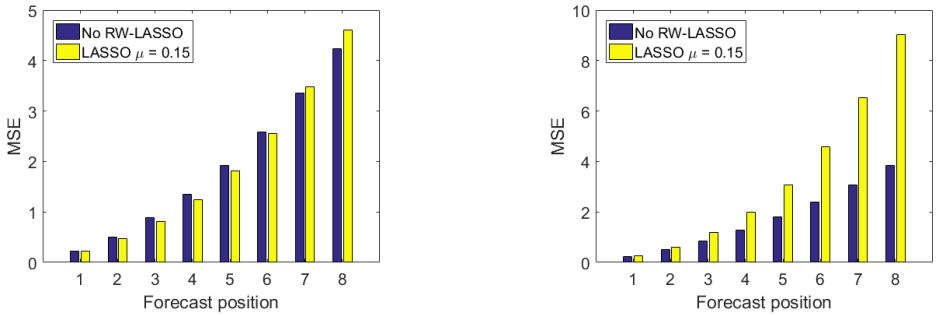
Regarding the detailed analysis of the MSE evolution, the reactions of the predictor to the addition of the stationary signal are in conformity with the former studies. Once again, the system appears to react more positively to sparser, lower order AR models opposed to the more abundant or higher order estimates. The graphics illustrating this behaviour are presented in appendix A.



(a) MSE of the first target. (b) MSE of the second target. (c) MSE of the third target.

Figure 5.12: Mean squared error of various targets using a single motion model, taken from the target represented in the leftmost image.

In another scenario, the gap between the effect of the auto-regressive signal on the predictions in different targets is significantly wider. Both cases exhibit a satisfactory response to the simplification of the motion model, however, the addition of the AR model results in very dissimilar outcomes. Figure 5.17 shows the differing outcome errors. Examining the auto-regressive coefficient evolution leads to the conclusion that the consequences of incorrectly estimating the parameter values are disastrous. Much like before, an overfit of the signal causes the MSE to more than double in value for the higher order forecasts.



(a) MSE of the first target. (b) MSE of the second target.

Figure 5.13: Mean squared error difference resultant of the incorrect estimation of the AR model.

In fact, on all of the conducted tests the dominant origin of large errors was this overfitting effect. Whenever this misestimation occurs during a period where the third order integrator motion model is active, the predictions seem to diverge from the truth disproportionately. Below, two snapshots of the forecast coordinates of three series in a group are depicted. These images follow the same color scheme of figures 5.9 and 5.10, with the addition of the uncertainty areas illustrating the regions with 99% probability of containing the actual observations. These zones are circular, centered in the correspondent forecast points and their borders are represented with a shade of blue that is brighter for in predictions farther in the future. The growth of these spaces is big compared to the distance travelled at each epoch, so it is common for the observations to fall within these areas. Yet, the excessive deviation induced by the stationary input is great enough to distort the predictions to a state where these uncertainty regions are not reliable.

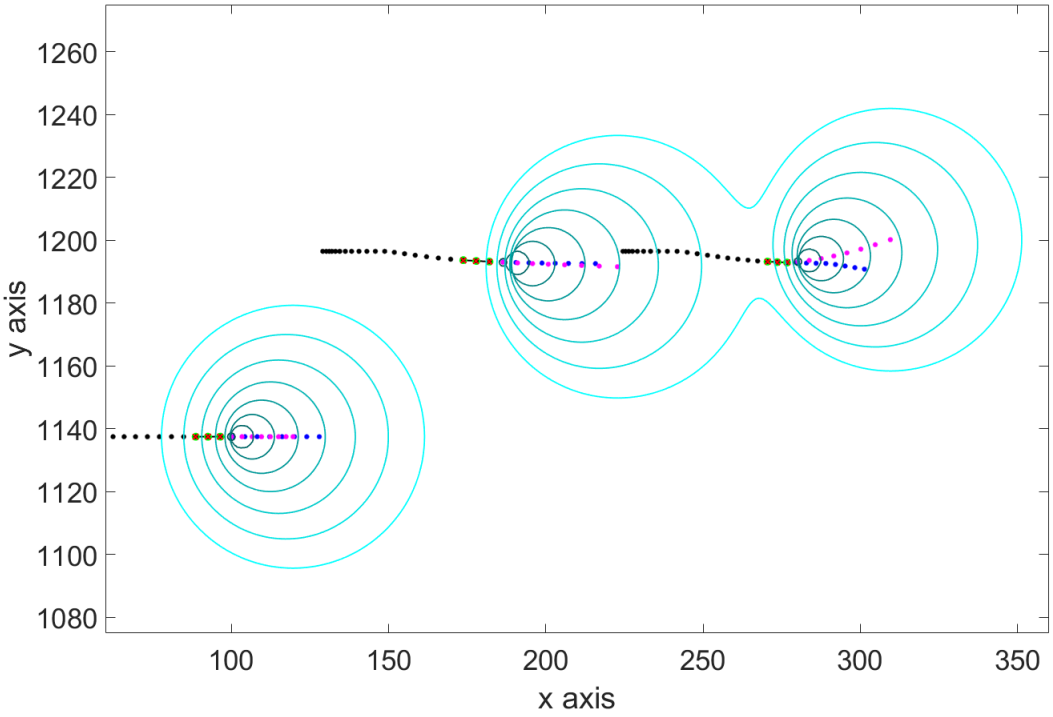


Figure 5.14: Group prediction without any stationary input.

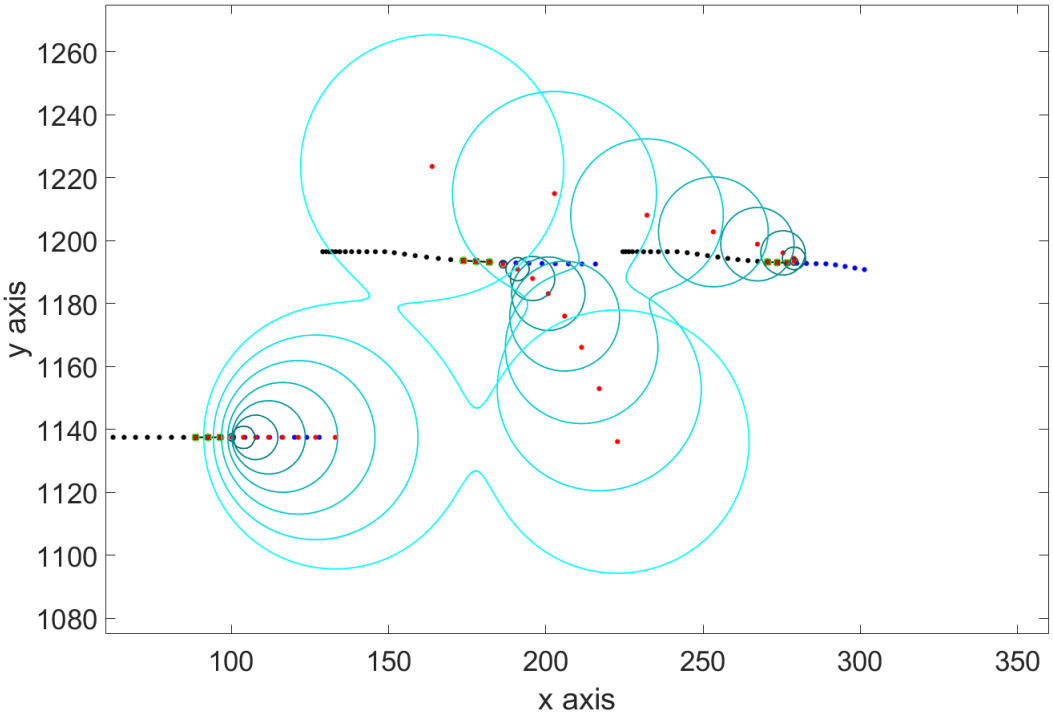


Figure 5.15: Group prediction with an incorrectly estimated stationary input.

A big part of the accuracy of the predictor appears to derive from the correct choice of the motion model. On the other hand, the employment of the AR model provides modest improvements, but carries with it the risk of notably worsening the forecasts if its coefficient values are severely misestimated. In

an effort to consolidate this hypothesis, a set of trajectories that did not belong to the same group was chosen for further testing the generalized version of the algorithm. This set (figure 5.16) consists of two targets that start in similar places and travel along the same direction for a while, splitting up in the end. Analyzing solely the first half of the paths, it is understandable that these targets might be considered as part of the same group.

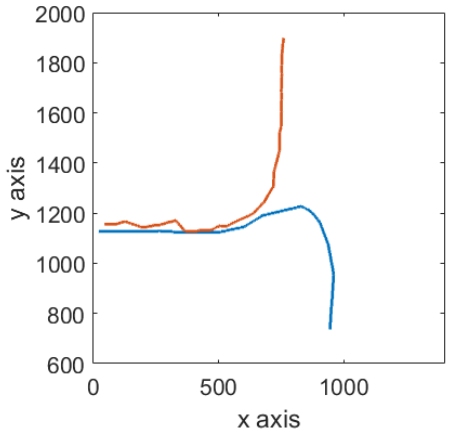
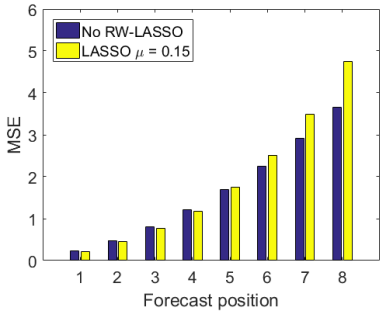
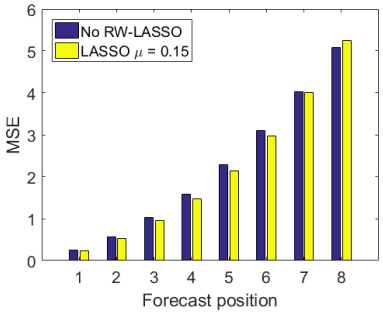


Figure 5.16: Paths travelled by two targets whose similarities could mistakenly classify them as belonging to the same group.

As was expected, forecasting the second target using the motion model extracted from the first one deteriorated the results of the algorithm. The general estimates suffered with the simplification derived from the assumption that these two targets belonged to the same class. Whereas the stationary signal input mostly lowered the accuracy of the predictor for the correctly modeled target, it improved a good part of the forecasts of the incorrectly modeled one. However, because of this simplification, the base prediction MSE of the second target was comparable to the worsened predictions of the first.



(a) MSE of the first target.



(b) MSE of the second target.

Figure 5.17: Mean squared error difference resultant of the incorrect application of the motion model on the second target.

A possible solution for this problem might be the adaptive classification of groups based on the similarities of the different series. To ensure the plausibility of the clustering, an evaluation of the groups in terms of the distance between targets and possibly even their past motion should be conducted from time to time.

Chapter 6

Conclusions

This report sought to study the effects of pursuing a relatively high degree of sparseness in certain key components of an online time-series prediction algorithm. The separation of the signal motion in two parts, one more general and the other pertaining to more detailed aspects of the trajectories, proved useful in the forecasting of the desired data.

The generalized description of the motions provided a moderately accurate representation of the actual future values of the input. The choice of lower order integrators for the representation of the underlying models played an important part in the performance of the predictor, with the third order one occasionally generating rougher estimates due to the accentuated curves the model naturally produces. In addition to this, the adoption of a single motion model for a group of targets proved adequate for series that exhibited similar behaviours. Nonetheless, the clustering of objects in groups is still a relevant concern, since the trajectories may be alike during a specific interval of time but different on the whole. This temporary closeness between the paths can lead to incorrect classifications as a result of not having the entire data sets beforehand. The fact that the algorithm is built to be executed in real-time demands that the pairing of different targets be checked often to ensure the validity of the groups and undone if said pairing is deemed incorrect.

The influence of the stationary signal on the forecast motion was demonstrated to be considerable. Due to the accuracy of the already existent predictions, the negative impacts of the addition of the correction component, i.e. the AR model, are more perceptible than the positive ones. This is notable whenever an overfit of the model occurs, especially when paired with a third order integrator model, since this combination usually distorts the estimates to an absurd degree. These results confirmed again the superiority of the RW-LASSO method over the RLS, for situations such as the ones described in this report. The system demonstrated a better response to sparser, lower order estimates for the auto-regressive model opposed to more complex or higher order ones.

All in all, the designed predictor showed a reliable short term forecast of both singular and multiple target time-series. Medium to long term forecasting was proven to be somewhat untrustworthy, given a multitude of factors such as the unpredictability of the motion models, the uncontrolled growth of the guess uncertainty, among others. The computational load of the algorithm was not found to be a

prominent burden for real-time applications, although large sets of data with multiple targets are yet to be studied.

This report serves as incentive to further research into the applications of sparseness in forecasting and control problems. The results of the work here described confirmed that there are advantages in exploring sparse approaches when devising solutions to these kinds of questions.

Future improvements may focus on the fine-tuning of the RW-LASSO parameters adaptively, instead of pre-processing large amounts of data to adjust these coefficients. This regulation is crucial to the pursuit of an acceptable outcome and, in addition to this, is problem dependent and observed to be even target dependent. Because of this last aspect, the parameters had to be occasionally readjusted to match the requirements of the input data, as the magnitude of the noise appeared to change with the outline of the trajectory.

Other enhancements stem from the addition of data regarding the environment in which the targets are situated, to help predict possible motion transitions due to target-target and target-object interactions.

Additionally, any decrease in computation time and effort is a compelling objective to strive for and should also be considered as an interesting change to the algorithm.

Bibliography

[1] F. J. Nogales, J. Contreras, A. J. Conejo, and R. Espínola. Forecasting next-day electricity prices by time series models. *IEEE Transactions on Power Systems*, 17(2):342–348, 2002. ISSN 08858950. doi: 10.1109/TPWRS.2002.1007902.

[2] Q. Song and B. S. Chissom. Forecasting enrollments with fuzzy time series - part II. *Fuzzy Sets and Systems*, 62(1):1–8, 1994. ISSN 01650114. doi: 10.1016/0165-0114(94)90067-1.

[3] J. G. Gooijer and R. J. H. 1. 25 years of time series forecasting. *International Journal of Forecasting*, 22(3): 443–473, 2006.

[4] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah. A review on time series forecasting techniques for building energy consumption. 2017.

[5] L. Cao. Support vector machines experts for time series forecasting. 2003.

[6] F. E. Tay and L. Cao. Modified support vector machines in financial time series forecasting. 2000.

[7] G. P. Zhang. Time series forecasting using a hybrid arima and neural network model. 1999.

[8] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley Sons, Inc., 2015.

[9] V. Ş. Ediger and S. Akar. Arima forecasting of primary energy demand by fuel in turkey. 2007.

[10] P. Vijaykumar, A. Kumar, and S. Bhatia. Latest Trends, Applications and Innovations in Motion Estimation Research. *International Journal of Scientific & Engineering Research*, 2(7):1–6, 2011. ISSN 2229-5518. URL `http://www.ijser.org`.

[11] J.-W. Kampoon, W. Okolo, S. A. Erturk, O. Daskiran, and A. Dogan. Wind Field Estimation and Its Utilization in Trajectory Prediction. *AIAA Atmospheric Flight Mechanics Conference*, (December), 2015. doi: 10.2514/6.2015-0756. URL `http://arc.aiaa.org/doi/10.2514/6.2015-0756`.

[12] J. B. Elsner. Tracking hurricanes. *Bulletin of the American Meteorological Society*, 84(3):353–356, 2003. ISSN 00030007. doi: 10.1175/BAMS-84-3-353.

[13] H. A. Ramsay, S. J. Camargo, and D. Kim. Cluster analysis of tropical cyclone tracks in the Southern Hemisphere. *Climate Dynamics*, 39(3):897–917, 2012. ISSN 09307575. doi: 10.1007/s00382-011-1225-8.

[14] B. Rosenhahn, U. G. Kersting, A. W. Smith, J. K. Gurney, and T. Brox. A system for marker-less human motion estimation. pages 230–237, 2005. doi: 10.1007/11550518.

[15] M. Gou, O. Camps, and M. Sznaier. MoM: Mean of Moments Feature for Person Re-identification. *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, 2018-Janua:1294–1303, 2018. doi: 10.1109/ICCVW.2017.154.

[16] H. Yao, A. Cavallaro, T. Bouwmans, and Z. Zhang. Guest Editorial Introduction to the Special Issue on Group and Crowd Behavior Analysis for Intelligent Multicamera Video Surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(3):405–408, 2017. ISSN 10518215. doi: 10.1109/TCSVT.2017.2669658.

[17] J. S. Marques and M. A. T. Figueiredo. Fast Estimation of Multiple Vector Fields: Application to Video Surveillance. 2011.

[18] N. Ferreira, C. Silva, J. T. Klosowski, and C. Scheidegger. Vector Field k-Means: Clustering Trajectories by Fitting Multiple Vector Fields. 32(3), 2012.

[19] J. C. Nascimento and J. S. Marques. Trajectory Analysis in Natural Images Usingmixtures of Vector Fields. pages 4353–4356, 2009.

[20] M. Barão and J. S. Marques. Gaussian Random Vector Fields in Trajectory Modelling. *Proceedings of the 19th Irish Machine Vision and Image Processing conference*, (1):211–216, 2017.

[21] L. Blackhall and M. Rotkowitz. Maximum a Posteriori vs Maximum Probability Recursive Sparse Estimation. pages 472–477, 2009.

[22] E. M. Eksioglu. Sparsity Regularized RLS Adaptive Filtering. *IET Signal Processing Journal*, 5(5):480–487, 2011.

[23] D. Chen, L. Bako, D. Chen, and L. Bako. A Recursive Sparse Learning Method : Application to Jump Linear Markov Systems. 2011.

[24] R. Brás. *Adaptive Control with Sparse Models*. PhD thesis, Instituto Superior Técnico, 2017.

[25] C. Paleologu, J. Benesty, and S. CiochiÅa. A robust variable forgetting factor recursive least-squares algorithm for system identification. *IEEE Signal Processing Letters*, 15(May 2014):597–600, 2008. ISSN 10709908. doi: 10.1109/LSP.2008.2001559.

[26] T. R. Fortescue, L. S. Kershenbaum, and B. E. Ydstie. Implementation of self-tuning regulators with variable forgetting factors. *Automatica*, 17(6):831–835, 1981. ISSN 00051098. doi: 10.1016/0005-1098(81)90070-4.

[27] D. Angelosante and G. Giannakis. RLS-weighted Lasso for adaptive estimation of sparse signals. *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1(2):3245–3248, 2009. URL `http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4960316`.

[28] J. M. Lemos, L. Rato, and J. Marques. Switching reconfigurable control based on hidden markov models. *European Control Conference (ECC)*, 1999.

[29] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning Social Etiquette: Human Trajectory Understanding in Crowded Scenes. 2016.

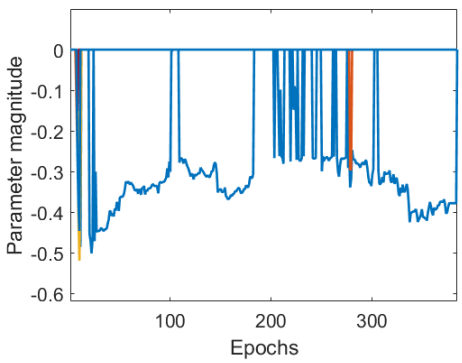
[30] D. Angelosante, J. A. Bazerque, G. B. Giannakis, S. Member, and G. B. Giannakis. Online Adaptive Estimation of Sparse Signals: Where RLS Meets the l1-Norm. *IEEE Transactions on Signal Processing*, 58 (7):3436–3447, 2010. ISSN 1053-587X. doi: 10.1109/TSP.2010.2046897. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5439789`.

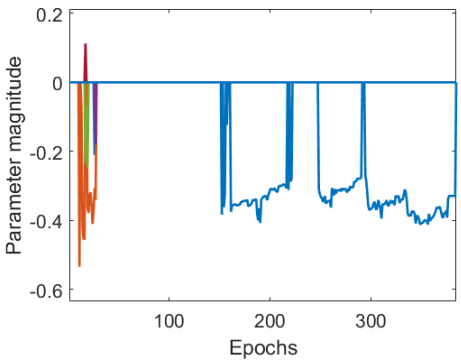
[31] H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101 (476):1418–1429, 2006. ISSN 01621459. doi: 10.1198/016214506000000735.

Appendix A

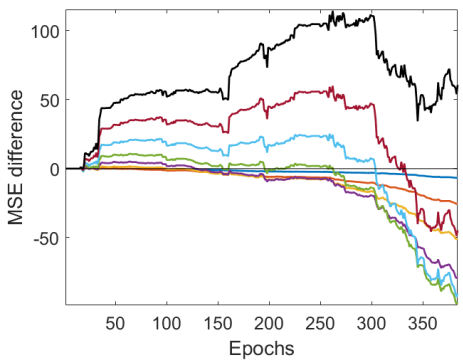
Supplementary results on the first group case study

Here, a detailed view of the mean squared error evolution related to the targets of the group depicted in figure 5.11 is provided.



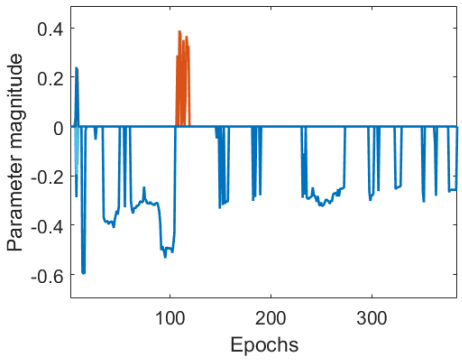


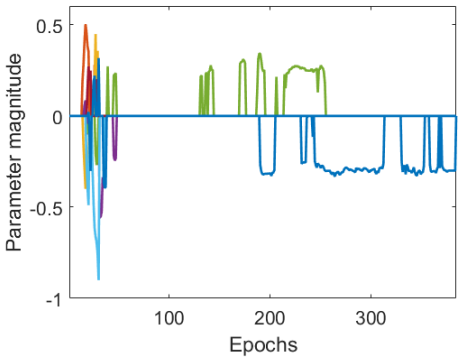
(a) AR model coefficients pertaining to the xx axis. (b) AR model coefficients pertaining to the yy axis.



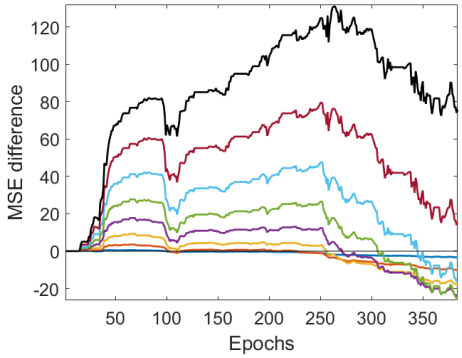
(c) MSE evolution in time.

Figure A.1: Mean squared error evolution of the first target of 5.11 (blue).



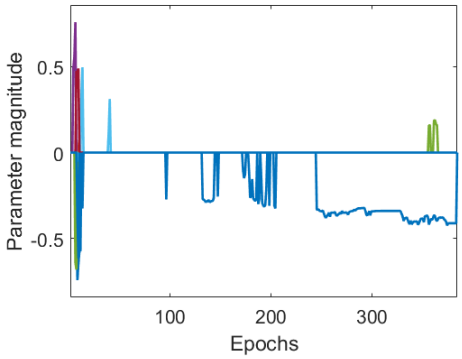


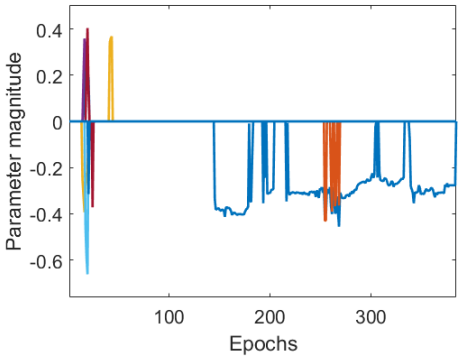
(a) AR model coefficients pertaining to the xx axis. (b) AR model coefficients pertaining to the yy axis.



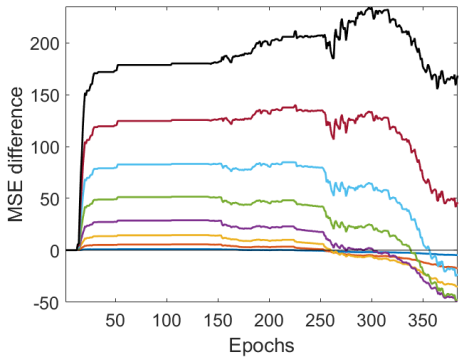
(c) MSE evolution in time.

Figure A.2: Mean squared error evolution of the second target of 5.11 (orange).





(a) AR model coefficients pertaining to the xx axis. (b) AR model coefficients pertaining to the yy axis.



(c) MSE evolution in time.

Figure A.3: Mean squared error evolution of the third target of 5.11 (yellow).