TÉCNICO LISBOA

Adaptive Control with Sparse Models

Rui Miguel Diogo Brás

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Prof. João Manuel Lage de Miranda Lemos

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. João Manuel Lage de Miranda Lemos
Members of the Committe: Prof. Mário Alexandre Teles de Figueiredo

September 2017

A plurality is not to be posited without necessity.

William of Ockham

Acknowledgments

I would like to start by thanking my supervisor, Prof. Dr. João Miranda Lemos for his guidance and invested time throughout this entire thesis. If it were not for his availability, knowledge, and wisdom, none of this work would have been possible.

I want to thank INESC-ID for their financial support and access to multiple resources and materials.

A special thanks to the Eng. Mário de Matos for providing the acquired data files from the electrical energy consumption of the buildings of IST.

I would also want to address and to thank my colleagues and friends, in particular Francisco Teles and Pedro Sá, who shared the most time with me during the entire course of this thesis.

Finally, I would like to express my sincere gratitude to my family, especially to my parents and sister, for always being supportive over these past 5 years.

The work in this dissertation was performed within the framework of project SPARSIS, Sparse Modeling and Estimation of Motion Fields, financed by FCT under contract PTDC/EEIPRO/0426/2014.

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

Abstract

The main motivation for the control strategies proposed in this work is to predict and estimate linear time-invariant and time-varying dynamical systems, while reducing the process model complexity. The identification problem studied aims at estimating the transfer function coefficients recursively and the order of the system, using classical methods of recursive model identification, such as Recursive Least Squares. Sparse constraints are appended to these methods such as to achieve subset model selection, and thus avoid model over-parameterization.

Owing to the nature of the ℓ_1 norm penalty, a sparse model parameter vector of estimates is obtained, and the true order of the system is revealed. The algorithms proposed are described and applied to the identification of sparse systems.

Two control strategies are considered in this work, one based on one-step ahead prediction, and other based on multi-step ahead prediction. These strategies are coupled to the algorithms described, and they are then tested and illustrated by simulation.

Keywords

Sparsity, Adaptive Control, System Identification, Prediction, Minimum Variance Control, Generalized Predictive Control.

Resumo

A principal motivação para as estratégias de controlo propostas é a predição e estimação da dinâmica de sistemas lineares invariantes no tempo e de sistemas lineares variantes no tempo.

O problema da identificação de sistemas estudado tem como objectivo a estimação recursiva dos coeficientes da função de transferência do sistema e respectiva ordem. Neste trabalho são utilizados métodos tradicionais para a realizar a identificação de sistemas de forma recursiva, como é o caso da versão recursiva do método dos mínimos quadrados. Estes métodos são depois aliados a restrições que enforçam a esparsidade das soluções, na busca pela obtenção de modelos simplificados, que evitem assim a sobre-parametrização de modelos.

Dada a natureza da norma ℓ_1, torna-se possível obter um vector de parâmetros esparso, que revela a correcta ordem do sistema. Os algoritmos propostos são descritos e aplicados através de simulações.

Neste trabalho, duas estratégias de controlo são consideradas, uma baseada em predição um passo à frente, e uma outra baseada em predição múltiplos passos à frente. Estas estratégias são acopladas aos algoritmos de estimação enunciados e são depois testadas e ilustradas em simulação.

Palavras Chave

Esparsidade, Controlo Adaptativo, Identificação de Sistemas, Predição, Controlo de Variância Mínima, Controlo Preditivo Generalizado.

Contents

List of Figures

List of Tables

Acronyms

ARMA	AutoRegressive Moving Average
ARMAX	AutoRegressive Moving Average with eXogeneous input
ARX	AutoRegressive with eXogeneous input
CARIMA	Controlled AutoRegressive Integrated Moving Average
CARMA	Controlled AutoRegressive Moving Average
LASSO	Least Absolute Shrinkage and Selection Operator
GPC	Generalized Predictive Control
LMS	Least Mean Square
LS	Least Squares
MPC	Model Predictive Control
MSE	Mean Square Error
MVC	Minimum Variance Control
NLMS	Normalized Least Mean Square
R-LASSO	Recursive LASSO
RLS	Recursive Least Squares
RW-LASSO	RLS-Weighted LASSO
RZA-NLMS	Reweighted Zero-Attractor Normalized Least Mean Square
ZA-LMS	Zero-Attractor Least Mean Square

1 Introduction

Contents

The purpose of this thesis is to develop control strategies for sparse linear systems. The problem of overestimation is common to a wide range of applications where the system dynamics is unknown or slowly-varying. The definition of the problem is presented in the following section, and a literature review on this matter is done afterwards. This chapter also includes the main contributions, the thesis outline, and some of the notational choices made.

1.1 Context and Motivation

Adaptive control is one of the broadly used control strategies to design advanced control systems. Originally conceived in the 1950s, adaptive controllers popularity and growth result from their clearly defined goal: to control plants with unknown or slowly varying parameters. Some mechanisms can create variations in process dynamics, and sometimes reducing these variations can be as simple as introducing nonlinear compensations in the controller. Different controller parameters can be designed for different operating points, if the process is well-known. However variations of process dynamics may come from many different sources of disturbances, most of them not fully understood. In these cases adaptive control can be much more advantageous, since searching for an explanation of the process variations sources can be unfeasible or not very economical.

The richness in terms of algorithms and design techniques comprises a vast number of applications such as chemical processes, ship steering, robotics [1], or biomedical systems, like anesthesia administration [2]. Another application of adaptive control is aircraft control. In fact, the design of autopilots for high-performance aircraft was one of the major motivations for the research and studies on adaptive control [3].

PID controllers are the standard tool in control and their purpose is to drive the error between a reference signal and the output of the plant to zero. They consist of three parts: the proportional part amplifies the error, the integral part eliminates the steady-state error, and the derivative part is used to reduce oscillations caused by the other two. The three signals resulting from each part are added together and plugged in the plant input. Herewith, each system requires manually tuning the gains of the PID. The parameters of a plant can change as the result of state changes, systems degradation or environmental issues, and whenever the parameters of the plant change, the gains of the PID need to be retuned. Adaptive control is prepared to deal with these changes by estimating the new plant parameters and recalculating the gains of the controller. In this work the focus is on Self-Tuning Adaptive Control [4, 5] even though there are numerous types of adaptive controllers [6].

Figure 1.1 shows the block diagram of Self-Tuning Adaptive Controllers. The Plant represents the dynamical system to be controlled while the Controller manipulates the system input u at each time instant k, to drive the system output y toward the reference r. Since the plant parameters may change continuously, the Self-Tuning Adaptive Controller needs to calculate these parameters, and thus the controller parameters online, which requires an estimation algorithm that updates the parameters recursively. Known as the certainty equivalence principle, the estimates are assumed to be equal to the true parameters of the process. The architecture consists of an Adapter and a

Controller. The Adapter itself is a combination of two other blocks, the Recursive Identifier and the Design Control blocks. The former is responsible for the identification procedure, at which a recursive estimation algorithm [7] is used to estimate the unknown plant parameters $\boldsymbol{\theta}$ given the plant inputs and outputs. The latter is a design method that recalculates the controller gains K according to the new estimates of the model. Estimation based designs are more versatile and applicable allowing a

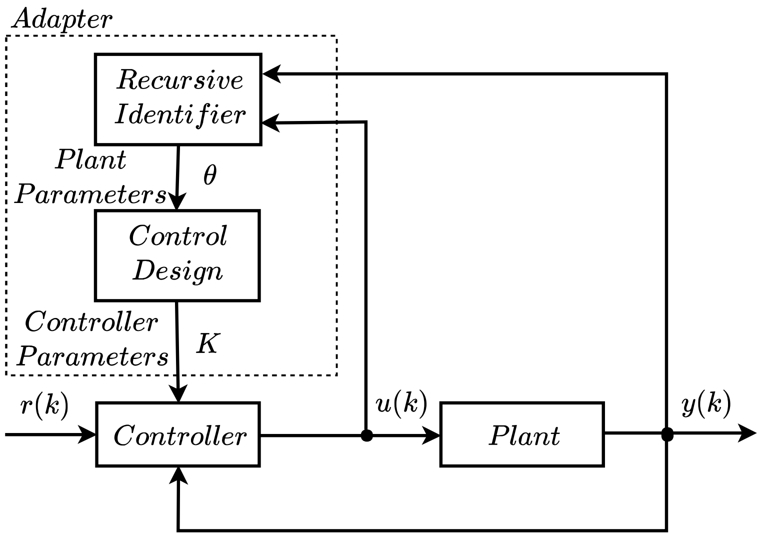

Figure 1.1: Self-Tuning Adaptive Controller architecture

choice of parameter update laws from a wide list of gradient and least squares optimization algorithms. This versatility owes to the fact that the Adapter can be analyzed as an independent module whose properties and conditions can be tested and guaranteed individually.

A basic principle in system identification is the parsimonious principle, which states that the final model should have a dimension only large enough to represent the underlying dynamics. According to this principle, selecting a simple model whose representation is enough to explain the system behavior becomes essential. This argument is even more meaningful when dealing with large data sets. Not only we want to consume less computational time but also save storage space. To cope with it, additional constraints, reflecting specific properties or assumptions, must be introduced.

A good fit to the measured data can not be achieved with few parameters (underfitting). On the other hand, if too many parameters are used, the fit to this data can be really good, but for a different dataset the fit may be very poor (overfitting). The simplest assumption that can be made about the problems structure is the solution sparsity, which assumes that only a relatively small subset of variables is critical in a specific context. The exploitation of sparsity to improve estimation performance has been attracting considerable interest and it is the leading motivation of this work [8, 9].

1.2 State Of The Art

In recent years, sparsity has become an important topic in estimation. The idea of adding a regularization term to cost functions, in order to penalize nonzero elements, arises in a plethora of problems, from system identification [10] to signal processing [11–13].

When designing controllers, models are a key component due to its heavy influence. Systems are often of low order or else they can be represented by a few number of parameters. While the proper representation may be known, the specific coefficients with nonzero value within it are unknown. Since there is no certainty about which parameters are nonzero, the parameter vector may be represented having an high dimension $-$ sparse system. After defining the model structure of the system, an appropriate method is used to identify its parameters. A basic technique to estimate the model parameters is the Least Squares (LS) method [14], which is fairly simple if the model is linear in the parameters.

However, LS estimates are nonzero, even of parameters that have zero as their true value, making the interpretation of the final model really challenging for large data sets. To estimate p parameters one should have far more n observations than p. In fact, if $p > n$, the LS estimates are not unique. There is an infinite set of solutions that minimize the objective function, almost surely leading these solutions to overfit the data. If $p \gg n$ and the underlying model is not sparse, then the number of observations n is too small to accurately estimate the parameters. Nonetheless if the true model is indeed sparse, it is then possible to effectively estimate the parameters using proper algorithms and methods.

In many scenarios of system identification, the system impulse response can be assumed to be sparse [10], containing only a few large parameters distributed among many minor ones. The basis of sparse system identification is to incorporate this sparse prior information to improve the estimation procedure.

The field of hybrid system identification has also benefited from new discoveries and methods using convex relaxations and sparsity [15, 16]. One example is model segmentation, a special case in which the system parameters are piecewise constant, and change only at rare time instants. In [17], a sparse representation aids in the identification of infrequently changing parameters.

Several techniques for sparse estimation have been successfully used for model subset selection [18–20] with the Least Absolute Shrinkage and Selection Operator (LASSO) [21] leading to a significant evolution and development of sparse representations in a broad range of problems.

In [22], Zhang *et al.* summarized a vast number of sparse representation algorithms and methods, from viewpoints of the mathematical and theoretical optimization, developed prior to 2015. Despite the clear merits of all the contributions, developing more efficient and robust sparse representation methods is still a challenging task.

Most of these algorithms are, however, unable to identify the unknown system/signal online, and the relevance of this work is precisely to solve this problem using recursive algorithms with ability exploit the sparse nature of the system/signal.

When systems/signals are time-varying or the available storage and resources are limited, online algorithms are of great importance. So far, only in a few recent contributions these adaptive methods have been addressed. In this respect, ℓ_0 regularized Least Mean Square (LMS) [23] and ℓ_1 regularized LMS [24] algorithms have arisen in the context of adaptive filters. In [25–27], ℓ_1 norm regularized

Recursive Least Squares (RLS) algorithms has been proposed. Due to the non differentiability of the ℓ_1, subgradient methods are used to provide the information for the following estimates. Similarly, Expectation-Maximization and Kalman filtering algorithms combined with ℓ_1 regularization are proposed in [28, 29]. A projection based adaptive algorithm is developed by Kopsinis *et al.* in [30], in which a performance comparison is done to online algorithms up to date. A different approach is proposed in [31], where a homotopy scheme is used to update the LASSO solution.

Sparse systems can be arbitrary sparse or exhibit additional structure. The latter refers to the case in which the impulse response of the system is composed of a few distinct clusters of nonzero coefficients $-$ group (or block) sparse system. Applications where group sparse models appear include multi-band signals [32] and wireless channels [33]. Examples of usage of $\ell_{1,2}$ and $\ell_{1,\infty}$ mixed norms to promote group sparsity can be found in [34, 35].

Not just in estimation but also in control theory, sparsity related works have been developed with the intention of reducing the size of control inputs and the number of control actions, for instance, to spare systems and allow them to last longer.

In recent works, variations of classical quadratic Model Predictive Control (MPC) methods have been developed. These alternatives use a combination of ℓ_1 and ℓ_2 penalties to obtain spatially or temporally sparse control signals [36, 37]. One application of this strategy is the minimum fuel control [38], in which the amount of fuel consumed is minimized using the ℓ_1 norm of the control action. A different example is the stop-start system of vehicles, in which the internal combustion engine is stopped when the vehicle stops or the speed is lower than a given threshold to reduce the amount of time the engine spends idling, thereby reducing fuel consumption and gases emissions. This feature is also present in hybrid electric vehicles, where the internal combustion engine is stopped and the electric motor is used as an alternative [39].

1.3 Thesis Contributions

The focus of this work is on the application of adaptive algorithms to control sparse systems. Based on RLS and LMS, several sparsity aware recursive algorithms are tested and compared with the aim of estimating the system parameters, thus achieving sparse solutions.

The objective of this dissertation is to develop and test a control strategy using one of the sparsity aware recursive algorithms. The problem of model over-parameterization is tackled by obtaining sparse models, enough to explain the systems dynamical behaviors.

The AutoRegressive Moving Average (ARMA) family models are used to describe all the systems presented in this work. Two control strategies are considered, Minimum Variance Control (MVC) that uses one-step ahead predictors, and Generalized Predictive Control (GPC), a multi-step ahead predictor.

The main contribution is then the application of these sparsity inducing recursive algorithms in control systems, and its demonstration in simulation using distinct models.

The proposal of time varying forgetting for recursive sparse estimation algorithms is also a contribution.

1.4 Thesis Outline

The dissertation is organized as follows. Chapter 1 includes an introduction to the work developed in this dissertation supported by the context and motivation, followed by a literature review, and a presentation of the main contributions.

In chapter 2 the concept of sparsity is explored as well as the advantages of its use.

In chapter 3 the general method for system identification is detailed.

Chapter 4 describes multiple sparsity enforcing algorithms. Simulations are made to analyze and compare them, and the respective results are discussed.

In chapter 5 the control strategy for identification and control of sparse models is detailed. Coupled with the algorithms previously shown, the developed framework is demonstrated through simulation tests. The results obtained are then presented and discussed.

Chapter 6 recaps the main results. Conclusions are then presented, followed by a summary of considerations that seem interesting for a future work.

1.5 Notation

In this section the notation used throughout this thesis is established. The following choices were made:

- Scalars and vectors are represented by lowercase letters, except for the Kalman gain vector which is represented by K.
- Matrices are represented by uppercase letters.
- To distinguish between the real parameters and the estimated parameters, the former are always referred to as the parameters of the plant or process, whilst the latter are referred to as the parameters of the model.
- The ℓ_1 norm of vectors is expressed by $\|\cdot\|_1$, while $|\cdot|$ denotes the absolute value of scalars. Whenever $|\cdot|$ is used as cardinality set a remark is made to discriminate the two.
- q^{-1} denotes the one-step delay operator satisfying $q^{-1}f(k) = f(k-1)$, for any time function $f(k)$.

2 Sparsity

Contents

Sparsity and its applications have been attracting researchers from a variety of fields, including signal processing, machine learning, image processing, and computer vision, since the second half of the 20^{th} century [40]. In these past decades, society and researchers in particular have benefited from the significant discoveries and advances in these areas.

One impressive sparsity-related breakthrough is compressive sensing [11], a technique used in signal processing to acquire and reconstruct signals. Results show that a sparse signal can be reconstructed with fewer samples than those required by the classical Shannon-Nyquist Theory [12].

The challenge of processing signals under the frameworks of the Shannon-Nyquist Sampling Theorem is to reduce signal acquisition costs, that is, sampling efficiently from large data sets without large memory requirements.

Generally, high-dimensional, limited sample inference is both underdetermined and computationally intractable, except if the problem has specific properties or structure, like sparsity. Saving both sampling time and sample storage space motivates the development of sparse representation techniques. Examples include finding a subset of genes responsible for specific diseases [21], reconstructing high-quality images from a compressed set of measurements [41], or in the context of this work estimating the parameters of a process in a high-dimensional but small sample statistical setting.

A sparse statistical model is one in which only a relatively small number of parameters (or predictors) plays an important role. This idea matches the philosophical principle of parsimony, also known as *Ockham's razor*, attributed to William of Ockham, which states that "entities should not be multiplied unnecessarily".

2.1 Sparsity and the ℓ_1 norm

Inferring quantities of interest from measured information is a common task in many practical problems. Consider a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and a vector $\mathbf{y} \in \mathbb{R}^m$. The vector $\mathbf{x} \in \mathbb{R}^n$ can be inferred either from noiseless observations of $\mathbf{y}$

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{2.1}$$

or from noisy observations

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}. \tag{2.2}$$

In the low dimensional case, when the number of observations is larger than the number of variables such that $m > n$, as in classical statistical setups, system identification and adaptive control, the desired $\mathbf{x}$ is the unique solution to the system of linear equation.

In the high dimensional case when $m < n$, the system is underdetermined (more unknowns than equations), which implies that the solution is not unique, provided that there is at least one. Assuming a matrix $\mathbf{A}$ with full-rank (its columns span the entire $\mathbb{R}^n$) guarantees that a solution can be sought to.

Typically, a single solution is desired and the fact that the system can have an infinite number of solutions is a major problem. In order to limit this number to one well-defined solution, additional properties are required. One approach to induce these properties is through *regularization*. Regularization is useful as it encodes additional criteria to the problem, and evaluates the desirability of a possible solution.

A constrained optimization problem can be defined by rewriting (2.1) as

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & r(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{y} = \mathbf{A}\mathbf{x} \end{aligned} \tag{2.3}$$

where $r(\mathbf{x})$ denotes the regularization term. Additionally, under noisy conditions, as happens with realistic measurements and signals, the problem is

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & r(\mathbf{x}) \\ \text{subject to} \quad & \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \leq \epsilon \end{aligned} \tag{2.4}$$

with error tolerance $\epsilon > 0$.

Recall the definition of convex sets and convex functions [42]:

Definition 1. ***(Convex Set)*** *A set $\mathcal{S}$ is convex if $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}$ and $\forall \alpha \in [0, 1]$ the convex combination $\mathbf{x} = \alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2$ is also in the set $\mathcal{S}$.*

A function $f(\mathbf{x})$: $\mathcal{S} \to \mathbb{R}$ defined on a convex set $\mathcal{S}$ in a vector space is convex if

$$f((1-\alpha)\mathbf{x}_1 + \alpha\mathbf{x}_2) \leq (1-\alpha)f(\mathbf{x}_1) + \alpha f(\mathbf{x}_2), \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{S}, \forall\alpha \in [0,1] \tag{2.5}$$

Regarding (2.5), the left-hand side of the inequality is the function f evaluated at the convex combination of $\mathbf{x}_1$ and $\mathbf{x}_2$, that corresponds to a point on the line segment between $\mathbf{x}_1$ and $\mathbf{x}_2$. This quantity must be less or equal to f evaluated at the same convex combination of the points at the endpoints. Geometrically this means that the cord connecting $\mathbf{x}_1$ and $\mathbf{x}_2$ must lie above the graph of f as represented in Figure 2.1. If equation (2.5) is a strict inequality, then f is called strictly convex. Although a convex function may not have a global minimum, if it has a local minimum then this point is

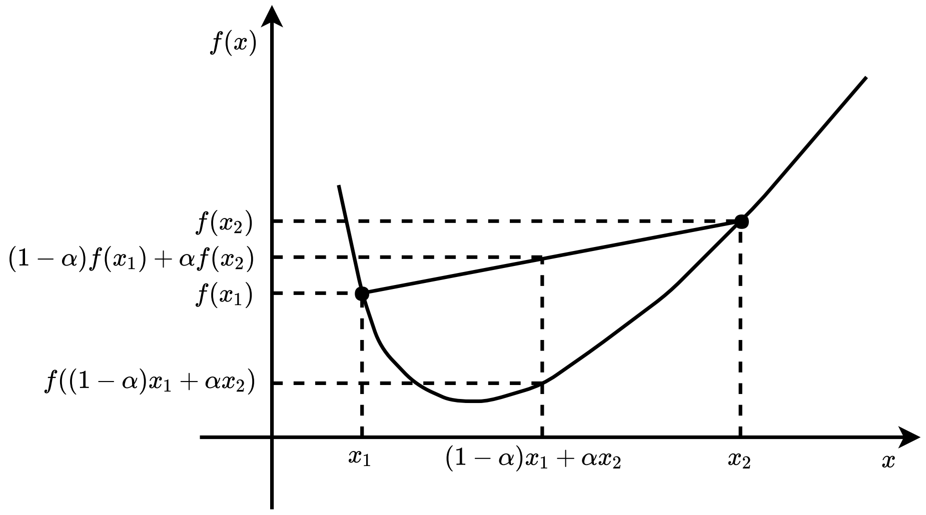
$f(x)$
$f(x_2)$
$(1-\alpha)f(x_1) + \alpha f(x_2)$
$f(x_1)$
$f((1-\alpha)x_1 + \alpha x_2)$
x_1
$(1-\alpha)x_1 + \alpha x_2$
x_2
x

Figure 2.1: Convex function illustration

also a global minimum. The squared Euclidean norm $\|\mathbf{x}\|_2^2$ is a well-known regularization term since it is strictly convex and thus always has a unique minimum. In addition, the solution is available in closed form.

The solution of a constrained optimization problem like (2.3) can often be found by using the so-called Lagrange multipliers method. The Lagrangian is defined as follows

$$\mathcal{L}(\mathbf{x}, \lambda) = \|\mathbf{x}\|_2^2 + \lambda^T(\mathbf{A}\mathbf{x} - \mathbf{y}), \tag{2.6}$$

being λ the Lagrange multiplier for the constrained set. Taking the derivative of $\mathcal{L}(\mathbf{x}, \lambda)$ with respect to $\mathbf{x}$ yields

$$\frac{\partial \mathcal{L}(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = 2\mathbf{x} + \mathbf{A}^T\lambda. \tag{2.7}$$

Equating (2.7) to zero the solution $\hat{\mathbf{x}}$ is then

$$\hat{\mathbf{x}} = -\frac{1}{2}\mathbf{A}^T\lambda, \tag{2.8}$$

and replacing $\mathbf{x}$ in (2.1) by (2.8) yields

$$\lambda = -2(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{y}, \tag{2.9}$$

which plugged in (2.8)

$$\hat{\mathbf{x}} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{y}. \tag{2.10}$$

Recall that $\mathbf{A}$ was assumed full-rank and therefore $\mathbf{A}\mathbf{A}^T$ is positive-definite and invertible. The simplicity of the illustrated closed-form and unique solution is the reason for the common use of the ℓ_2 norm, yet it does not necessarily make the Euclidean norm the best choice for regularization [40, Chapter 1]. Different convex, or strictly convex, functions should be considered. Interesting choices of regularization functions are the so-called ℓ_p norms. Consider an n-dimensional vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$ in an Euclidean space $\mathcal{V}$. A norm is a function f: $\mathcal{V} \to \mathbb{R}$ that satisfies the following properties:

i) $f(\mathbf{x}) \geqslant 0$ for all $\mathbf{x} \in \mathcal{V}$

ii) $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{V}$ (triangle inequality)

iii) $f(\lambda\mathbf{x}) = |\lambda| f(\mathbf{x})$ for all $\lambda \in \mathbb{C}$ and $\mathbf{x} \in \mathcal{V}$ (positive homogeneity)

iv) $f(\mathbf{x}) = 0 \Leftrightarrow \mathbf{x} = 0$

The ℓ_p norm of $\mathbf{x}$ is given by

$$\|\mathbf{x}\|_p = \begin{cases} |j \in \{j, \dots, n\} : x_j \neq 0| \text{ , if } p = 0 \\ \\ (\sum\limits_{j=1}^{n} |x|^p)^{\frac{1}{p}} \text{ , if } 0 < p < \infty \\ \\ \max\limits_{j=1,\dots,n} |x_j| \text{ , if } p = \infty \end{cases} \tag{2.11}$$

where $|\cdot|$ denotes set cardinality.

For $0 < p < 1$, the ℓ_p "norm" violates the triangle inequality condition and hence it is called pseudo-norm. For $p = 0$ the homogeneity property, when $\lambda \neq 0$, is not satisfied, and thus ℓ_0 is also called pseudo-norm. As shown in equation (2.11) the cardinality of the vector is the ℓ_0 pseudo-norm. Its association with sparsity is evident as it intuitively corresponds to the number of nonzero elements of $\mathbf{x}$. Defining $\text{supp}(\mathbf{x}) \triangleq \{j \in \{j, \ldots, n\} : x_j \neq 0\}$ then sparsity amounts to having $|\text{supp}(\mathbf{x})| \ll n$. The problem of the ℓ_p pseudo-norms, with $0 \leq p < 1$, is their non-convexity, nonsmoothness, and global nondifferentiability.

The ℓ_1 norm corresponds to the sum of the absolute values of the elements in $\mathbf{x}$, and it is convex, nonsmooth and globally nondifferentiable. The ℓ_2 norm is convex, smooth and globally differentiable.

The optimal solution of a norm minimization problem is, by inflating the so-called norm ball, the first point of the corresponding norm ball to hit the hyperplane containing the solutions to the problem [22]. As it may be useful to take a look at the 2D-space example, Figure 2.2 shows the norm balls of different ℓ_p norms.

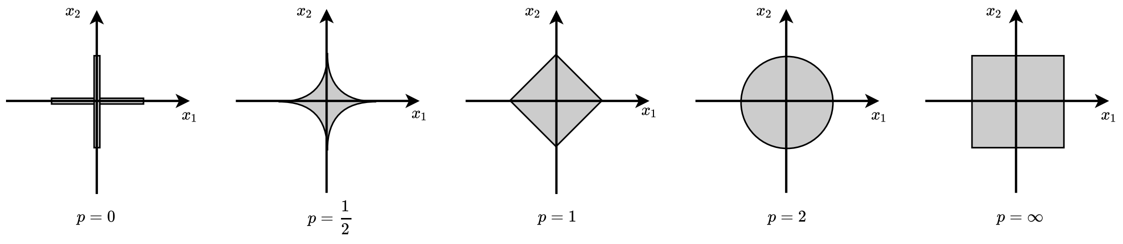

Figure 2.2: Geometric interpretation of the ℓ_p norm balls in 2D-space. The graph axes represent the x_1 and x_2 components of $\mathbf{x}$.

To understand how these different norm balls affect the solutions of said norms minimization and which of them can, in fact, induce sparse solutions, the geometry solution via ℓ_1 and ℓ_2 minimizations for two variables is depicted in Figure 2.3, from left to right.

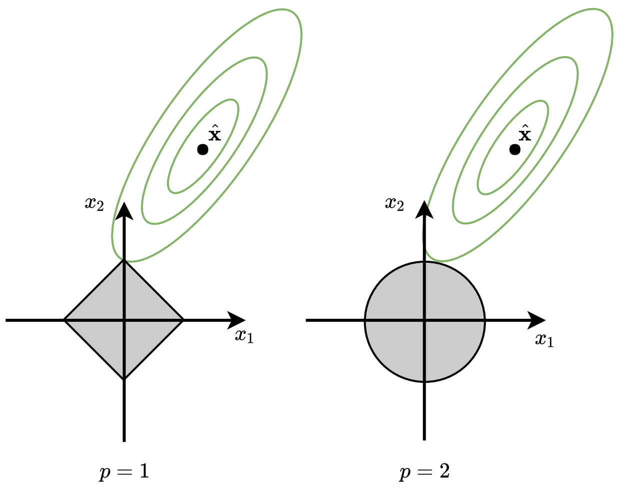

Figure 2.3: Geometry solutions of ℓ_1 and ℓ_2 norms minimization in 2D-space.

The gray areas correspond to the constraint regions while the green ellipses are the level sets of the objective function $\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$, where $\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$ is the minimum.

The reasoning behind some norms being more suitable for generation of sparse solutions lies in the location of the intersection between the norm ball and the set of elliptical contours. When the penalty function is nondifferentiable at points where $x_j = 0$, it forms a corner in the axes, promoting that the loss and the penalty functions meet at the axis. When they intersect on one of the coordinate axes, the solution is sparse (the other component is zero). Notice that the diamond-shaped region tends to meet the ellipses at the vertices (Figure 2.3 shows the 2D case, but it also generalizes to higher dimensions), which favours sparse solutions. Similarly, for $0 \leq p < 1$, the ℓ_p pseudo-norms are sparsity inducing. On the other hand, only ℓ_p norms with $p \geq 1$ are convex.

The ℓ_2 norm optimum will only be on one of the axes (that is, be sparse) if the minimum Mean Square Error (MSE) point $\hat{\mathbf{x}}$ is also on one of the axes, and this event will happen with probability zero. Because of its sharp contour at the axes, on the ℓ_1 regularized system the optimum can be on the axes even when the minimum MSE point $\hat{\mathbf{x}}$ is not. For the scenario depicted in Figure 2.3 it is clear that the two surfaces intersect at one of the axes, but typically a question arises "what if $\hat{\mathbf{x}}$ is located at a different position, will the two surfaces boundaries still intersect at one of the axes?"

Figure 2.4 shows different configurations of the tangent points of the ℓ_1 norm ball and the least squares error surfaces, while their centers remain fixed.

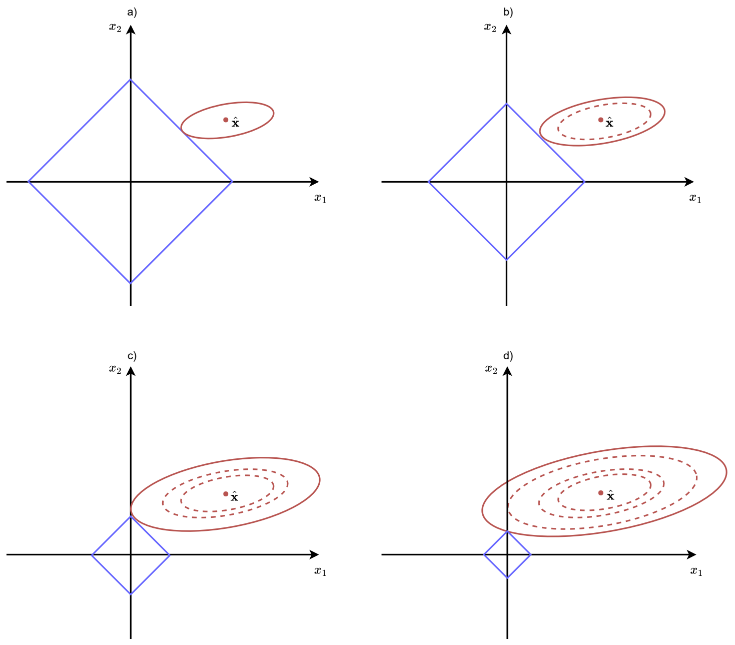

Figure 2.4: Distinct configurations of tangent points of the ℓ_1 norm ball and the least squares error surfaces, keeping their centers fixed.

To understand how Figures 2.4 a), b), c) and d) differ from each other it is important to recall that in the LASSO objective there is a parameter that sets how important the ℓ_1 norm term is, relative to the ℓ_2 norm term. If this regularization parameter is large, then the ℓ_1 norm term becomes the dominant contributor to the LASSO objective, and the minimizer will then have to make the ℓ_1 norm term really small. As the least squares error surface size increases relative to the ℓ_1 norm ball, the point of tangency gets closer to one of the vertices of the ℓ_1 norm ball. Therefore, at the minimum the ℓ_1 norm ball will be very small if the regularization parameter is very large. The adjustment of the regularization parameter value allows moving between the distinct scenarios of Figure 2.4, and a favourable choice will lead to a sparse solution.

The biggest advantage of the ℓ_1 norm is the success of achieving variable selection without surrendering the computational complexity of convexity.

2.2 LASSO

Replacing $r(\mathbf{x})$ with $\|\mathbf{x}\|_0$ in (2.4)

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \|\mathbf{x}\|_0 \\ \text{subject to} \quad & \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 \leqslant \epsilon \end{aligned} \tag{2.12}$$

yields the optimization problem for sparse signal recovery for the noisy observations case. Finding the sparsest solution is considered computationally intractable (NP-hard problem) [43], due to its non-convex combinatorial nature. Consider the simplest case where $\mathbf{x}$ is a 1-sparse vector [1] of size n and the position of the nonzero entry is unknown. There are $\binom{n}{1}$ possibilities that need to be searched in order to find the unique minimizer. As k grows the number of $\binom{n}{k}$ possibilities grows too. If k is not known *a priori* then all n possible values of k have to be considered, and the resulting complexity is $\sum\limits_{i=1}^{n}\binom{n}{i}$.

One way to bypass this problem is resorting to approximations such as addressing the problem via approximate methods, or replace it with convex relaxations. The latter is done by replacing the ℓ_0 pseudo-norm with a convex approximation. Any of the ℓ_p pseudo-norms $(0 \leq p \leq 1)$ is a good option to seek sparse solutions. From this set, however, the only convex one is the ℓ_1 norm. Accordingly, the problem (2.12) is transformed into a convex optimization problem [42], that is efficiently solvable by standard convex optimization techniques. The relaxation is then represented as

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \|\mathbf{x}\|_1 \\ \text{subject to} \quad & \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 \leqslant \epsilon. \end{aligned} \tag{2.13}$$

Moreover equation (2.13) can be written in two additional forms

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 \\ \text{subject to} \quad & \|\mathbf{x}\|_1 \leqslant d \end{aligned} \tag{2.14}$$

and

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1, \tag{2.15}$$

where d and λ are nonnegative real parameters uniquely defined by ϵ [44, Proposition 3.2.]. Problem (2.13) is a quadratically constrained linear program called *Basis Pursuit Denoising* [19]. When $\epsilon = 0$, problem (2.13) corresponds to a linear program known as as *Basis Pursuit* [18]. Finnaly, problem (2.15) is an unconstrained optimization problem known as LASSO. The sparse estimation methods of Chapter 4 are based on the LASSO problem.

[1] A k-sparse vector is characterized by k nonzero entries.

The first desirable property of a sparse estimator $\hat{\mathbf{x}}_t$ (estimator of $\mathbf{x}$ after t observations) concerns *support consistency*, as the estimator should asymptotically identify the right subset model

$$\lim_{t\to\infty} \Pr[\mathrm{supp}(\hat{\mathbf{x}}_t) = \mathrm{supp}(\mathbf{x})] = 1, \tag{2.16}$$

where $\Pr[\Omega]$ denotes the probability of the event Ω and $\mathrm{supp}(\mathbf{x})$ denotes the set of nonzero entries of $\mathbf{x}$ (the support of $\mathbf{x}$). In other words, when the regularization parameter λ is correctly chosen, the parameters which have nonzero components are correctly estimated with probability tending to 1.

The second property demands weak *estimation consistency* or optimal estimation rate

$$\sqrt{t}\,[\mathcal{S}(\hat{\mathbf{x}}_t) - \mathcal{S}(\mathbf{x})] \rightarrow_d \mathcal{N}(0, \Sigma^{-1}), \tag{2.17}$$

where $\rightarrow_d$ denotes convergence in distribution. The operator $\mathcal{S}(\hat{\mathbf{x}}_t) : \mathbb{R}^n \rightarrow \mathbb{R}^{|\mathrm{supp}(\mathbf{x})|}$ selects the entries of $\hat{\mathbf{x}}_t$ corresponding to $\mathrm{supp}(\hat{\mathbf{x}}_t)$, and $\hat{\mathbf{x}}_t$ is a consistent estimator of $\mathbf{x}$ with normal limit distribution with Σ^{-1} as the covariance matrix knowing the true subset model. These two properties are known as *oracle properties* [45] since an estimator which meets them is, asymptotically, as good as if $\mathrm{supp}(\mathbf{x})$ was known beforehand. The ideal procedure for variable selection should account for three different conditions [46]:

- 1. Unbiasedness - The resulting estimator should be nearly unbiased when the true unknown parameter is large to avoid excessive modeling bias.
- 2. Sparsity - The resulting estimator should be a thresholding rule, automatically estimating small parameters as zero to reduce model complexity.
- 3. Continuity - The resulting estimator should be continuous in data to avoid model prediction instability.

The sparse estimators described in Chapter 4 have sparsity imposing constraints, and consequently, as some of the system parameters are reduced to zero, the resulting bias is also reduced.

Even though sparse estimators variable selection may be inconsistent in certain scenarios, the oracle properties can be achieved using weighted ℓ_1 norm constraints. Withal, the oracle properties do not automatically result in optimal prediction performance [45].

2.3 Proximity operator of the ℓ_1 norm

The proximity operator is a natural generalization of the projection operator [47]. It has similar behavior to a gradient descent step for a given function f. Proximal algorithms are algorithms used for solving convex optimization problems. They work by iterating a sequence of steps in which proximity operators of the functions involved in the minimization are evaluated. Although these subproblems may be solved with standard methods, they often can be solved in closed form solutions or at least very quickly with simple specialized methods.

Proximal algorithms have several main advantages, among which the fact that they can be applied to nonsmooth functions or functions that are otherwise difficult to handle. Their simple nature allows them to be applied not only to very particular problems but also to large scale problems (as in distributed optimization [48, Chapter 5]).

Many optimization problems that arise in statistics take the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad \Phi(\mathbf{x}) := l(\mathbf{x}) + r(\mathbf{x}) \tag{2.18}$$

where $l(\mathbf{x})$ is a convex and smooth function, dependent of the observed data, and $r(\mathbf{x})$ is a lower semi-continuous, convex, and nonsmooth regularization function. For example, if $l(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ and $r(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, then equation (2.18) similarly corresponds to the LASSO problem.

Definition 2. *(**Lower semi-continuous funtion**) Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ and a point $\mathbf{x}_0 \in \mathbb{R}^n$. The function f is said to be lower semi-continuous at $\mathbf{x}_0$ if*

$$f(\mathbf{x}_0) \le \lim_{\mathbf{x}\to\mathbf{x}_0} \inf f(\mathbf{x}). \tag{2.19}$$

Furthermore, a function $f : \mathbb{R}^n \to \mathbb{R}$ is lower semi-continuous if and only if its epigraph is closed, that is, the set of points lying on or above its graph contains all its limit points [49, 50, Chapter 3]. Figure 2.5 depicts in blue a lower semi-continuous function f.

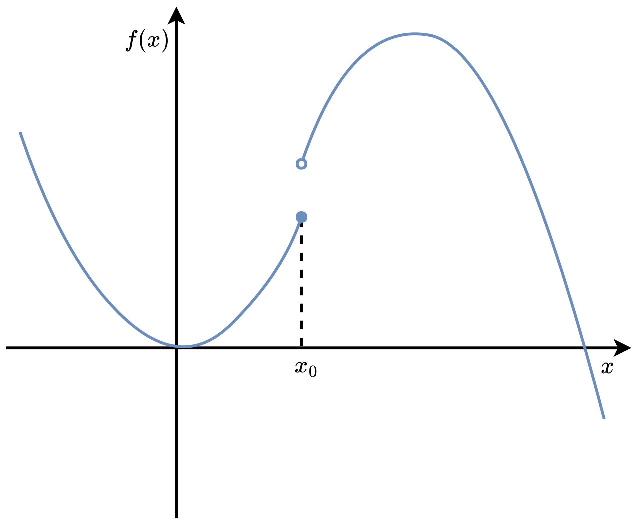

Figure 2.5: Lower semi-continuous function f.

Definition 3. *(**Moreau envelope**) Let $f(\mathbf{x})$ be a lower semi-continuous function, and let $\alpha > 0$ be a scalar. The Moreau envelope $f_\alpha(\mathbf{x})$ is*

$$f_\alpha(\mathbf{x}) := \inf_{\mathbf{z}} \left\{ f(\mathbf{z}) + \frac{1}{2\alpha}\|\mathbf{z} - \mathbf{x}\|_2^2 \right\} \le f(\mathbf{x}). \tag{2.20}$$

Evaluating the proximity operator can be viewed as a gradient-descent step for a regularized version of the original function, with α as a step-size parameter controlling the tradeoff between minimizing f and being close to $\mathbf{x}$.

The Moreau envelope is a regularized version of f, approximating it from below as shown in Figure 2.6. It has domain $\mathbb{R}^n$ and it is continuously differentiable, even when f is not.

The yellow line in Fig. 2.6 represents the function $f = |z| + |z-2| + |z+2| - |z-1|$, while the blue and red lines depict the Moreau envelope of f for $\alpha = 1$ and $\alpha = 0.5$, respectively. The significant point is that both f and f_α have the same set of minimizers. Also, for a smaller value of α, f is better approximated by the Moreau envelope.

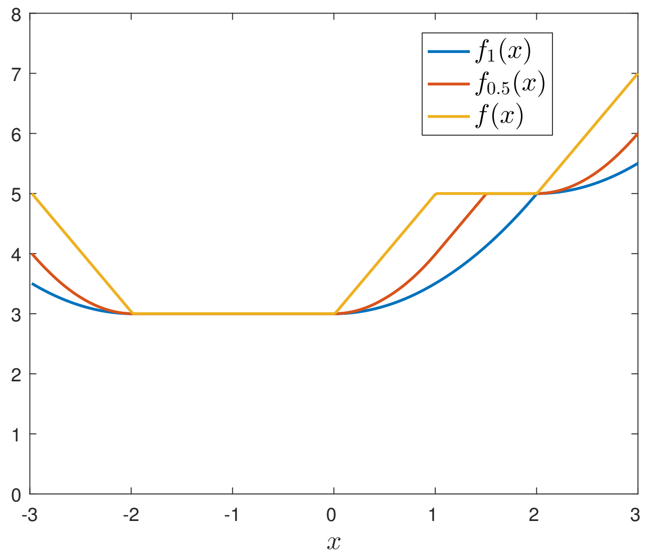

Figure 2.6: Function f and its Moreau envelope f_α, for α equal to 0.5 and 1.

Definition 4. *(**Proximity operator**) Let $f(\mathbf{x})$ be a lower semi-continuous function, and let $\alpha > 0$ be a scalar. The proximity operator prox$_{\alpha f}(\mathbf{x})$ of f with parameter α is*

$$\operatorname{prox}_{\alpha f}(\mathbf{x}) = \arg\min_{z} \left\{ f(\mathbf{z}) + \frac{1}{2\alpha} \|\mathbf{z} - \mathbf{x}\|_2^2 \right\}, \tag{2.21}$$

The proximity operator $\operatorname{prox}_{\alpha f}$ in Definition 4 specifies the value that solves the minimization problem introduced by the Moreau envelope definition.

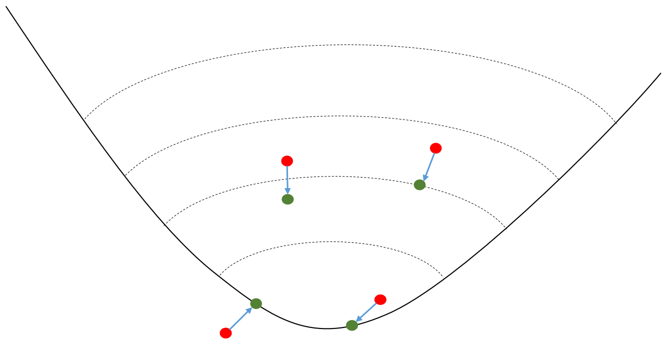

Figure 2.7: Proximity operator step at different points.

Figure 2.7 shows what happens when the proximity operator is applied. The black line corresponds to the boundary of the domain of f while the dashed lines are its level curves. Picking the red points and applying the proximity operator results in the corresponding green points. After the red points already inside the domain of f being mapped by the proximity operator, the corresponding green points remain inside its domain and move towards the minimum. In the situation of the red point outside the domain of f, it is first moved to the boundary of the domain and then moved towards the minimum. The parameter α controls the arrows length, with small values conferring smaller steps towards the minimum of f, and large values allowing bigger steps.

One property of proximity operators is the separable sum. If $f(\mathbf{x}_1, \mathbf{x}_2) = f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2)$, then

$$\mathrm{prox}_f(\mathbf{x}) = (\mathrm{prox}_{f_1}(\mathbf{x}_1), \mathrm{prox}_{f_2}(\mathbf{x}_2)). \tag{2.22}$$

Hence, the proximity operator of a separable function can be computed by calculating the proximity operator of each of the separable parts of the function independently. Evaluating the proximity operator of a fully separable function reduces to calculating the proximity operator of scalar functions.

The proximity operator can be seen as a gradient-descent step for the Moreau envelope of f known as the Proximal Gradient Descent. The derivative of the Moreau envelope is

$$\begin{aligned} \partial f_\alpha(\mathbf{x}) &= \partial \inf_{\mathbf{z}} \left\{ f(\mathbf{z}) + \frac{1}{2\alpha}\|\mathbf{z} - \mathbf{x}\|_2^2 \right\} = \\ &= \frac{1}{\alpha}\left[\mathbf{x} - \mathrm{prox}_{\alpha f}(\mathbf{x})\right], \end{aligned} \tag{2.23}$$

and thus that $\mathrm{prox}_{\alpha f}(\mathbf{x}) = \mathbf{x} - \alpha \partial f_\alpha(\mathbf{x})$.

Proximity operators and fixed-point theory are not entirely unrelated. In fact, $\mathrm{prox}_{\alpha f}(\mathbf{x}^*) = \mathbf{x}^*$ if and only if $\mathbf{x}^*$ is a minimizer of $f(\mathbf{x})$. This means that starting at an initial point and continuously applying the proximity operator such that

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \nabla f_\alpha(\mathbf{x}^k), \tag{2.24}$$

the minimum of the Moreau envelope is reached at convergence. Therefore a minimum of f is also reached, at which $\nabla f_\alpha(\mathbf{x}^k) = 0$, and consequently $\mathrm{prox}_{\alpha f}(\mathbf{x}^*) = \mathbf{x}^*$.

Considering $\Phi(\mathbf{x})$ from equation (2.18), the proximal gradient consists of only two steps that must be iterated until convergence.

- In the first step, a point $\mathbf{v}_k$ is defined by calculating the gradient step with respect to the differentiable part of Φ:
$$\mathbf{v}_k = \mathbf{x}_k - \alpha \nabla l(\mathbf{x}_k). \tag{2.25}$$
- The second step is the computation of the proximity operator of the nondifferentiable part of Φ at $\mathbf{v}_k$:
$$\begin{aligned} \mathbf{x}_{k+1} &= \mathrm{prox}_{\alpha r}(\mathbf{v}_k) \\ &= \mathrm{prox}_{\alpha r}(\mathbf{x}_k - \alpha \nabla l(\mathbf{x}_k)). \end{aligned} \tag{2.26}$$

Proof. If $\mathbf{x}^*$ minimizes Φ then

$$\frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}^*\|_2^2 + \Phi(\mathbf{x}) \geq \Phi(\mathbf{x}^*) = \frac{1}{2\alpha}\|\mathbf{x}^* - \mathbf{x}^*\|_2^2 + \Phi(\mathbf{x}^*),\ \forall \mathbf{x}. \tag{2.27}$$

Therefore $\mathbf{x}^*$ minimizes $\frac{1}{2\alpha}\|\mathbf{x} - \mathbf{x}^*\|_2^2 + \Phi(\mathbf{x})$, and $\mathbf{x}^* = \mathrm{prox}_\Phi(\mathbf{x}^*)$. On the other hand, the point $\bar{\mathbf{x}}$ minimizes $\frac{1}{2\alpha}\|\mathbf{x} - \mathbf{z}\|_2^2 + \Phi(\mathbf{x})$ if and only if $0 \in \partial\Phi(\bar{\mathbf{x}}) + (\bar{\mathbf{x}} - \mathbf{z})$. Considering $\bar{\mathbf{x}} = \mathbf{z} = \mathbf{x}^*$, $0 \in \partial\Phi(\mathbf{x}^*)$ thus $\mathbf{x}^*$ minimizes Φ. □

In this way, the minimizers of Φ correspond to fixed points of prox_{Φ}, and Φ can be minimized just by finding a fixed point of its proximity operator.

To further illustrate the operation of these operators, the proximity operator of the ℓ_1 norm is exemplified in Figure 2.8. The yellow line represents the function $f(x) = |x|$, while the red line

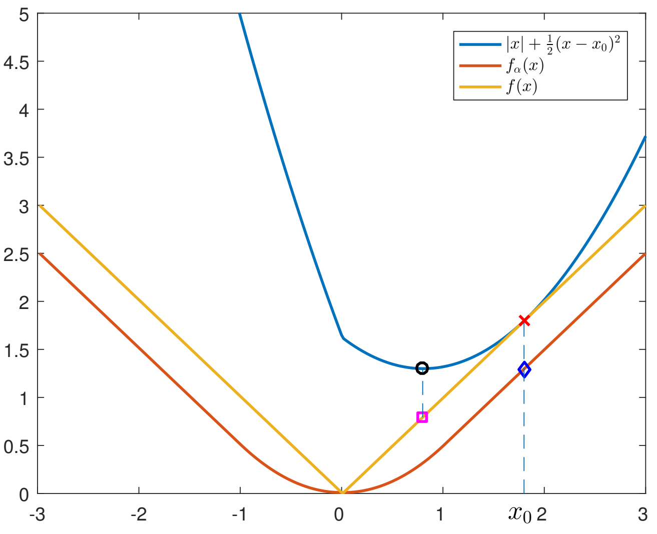

Figure 2.8: Proximity Operator and Moreau envelope (first iteration).

represents the corresponding Moreau envelope $f_\alpha(x)$ for $\alpha = 1$. The blue line is the function $|x| + \frac{1}{2}(x - x_0)^2$ with $x_0 = 1.8$. The minimum of this function, represented by a black circle, defines the proximity operator, and its coordinates are $(\mathrm{prox}_f(x_0), f_\alpha(x_0)) = (0.8, 1.3)$, which is closer to the minimum of f (at $x = 0$) than x_0. The magenta square is the point that characterizes x_1, the next point to iterate through. Figure 2.9 depicts the second iteration of the algorithm, wherein x_1 is taken as x_0 in Figure 2.8. This time the black circle is closer to the minimum of f than x_1.

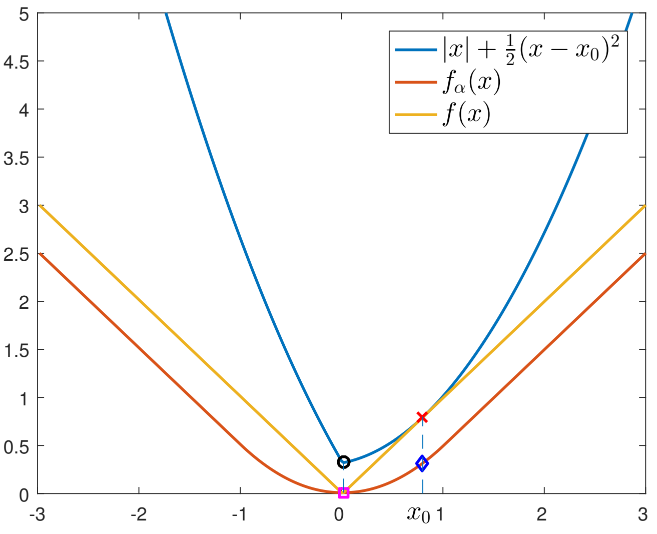

Figure 2.9: Proximity Operator and Moreau envelope (second iteration).

The optimization problem is

$$\mathrm{prox}_{\alpha\|\cdot\|_1}(\mathbf{x}) = \arg\min_{z} \left\{ |\mathbf{z}| + \frac{1}{2\alpha}\|\mathbf{z} - \mathbf{x}\|_2^2 \right\}, \tag{2.28}$$

which is separable, and thus equation (2.28) can be written as follows

$$\left\{\operatorname{prox}_{\alpha\|\cdot\|_1}(\mathbf{x})\right\}_i = \arg\min_{z_i}\left\{|\mathbf{z}_i| + \frac{1}{2\alpha}(\mathbf{z}_i - \mathbf{x}_i)^2\right\}. \tag{2.29}$$

Applying the optimality condition yields

$$\{\operatorname{prox}_{\alpha\|\cdot\|_1}(\mathbf{x})\}_i = \begin{cases} x_i - \alpha & \text{if} x_i > \alpha \\ 0 & \text{if} |x_i| \leq \alpha \\ x_i + \alpha & \text{if} x_i < -\alpha \end{cases}. \tag{2.30}$$

Equation (2.30) is known as the element-wise Soft Thresholding function, and can be written as $\{S_\alpha(\mathbf{x})\}_i = \operatorname{sign}(x_i)(|x_i| - \alpha)_+$, where $(\cdot)_+$ is the nonnegative part of the function inside parentheses.

For the LASSO problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1, \tag{2.31}$$

the objective is split in $l(\mathbf{x}) = \dfrac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ with gradient $\nabla l(\mathbf{x}) = \mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{y})$, and $r(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$ whose proximal operator $\operatorname{prox}_{\alpha r}(\mathbf{x}) = S_{\lambda\alpha}(\mathbf{x})$ defined as in equation (2.30).

3 System Identification

Contents

System identification is the experimental approach for modeling a process from input/output data. Its main steps involve the selection of a suitable model structure, the process model parameters estimation, and model validation.

The structure of a discrete linear, time-invariant system can be described by its input and output relationship as follows

$$A(q^{-1})y(k) = \frac{B(q^{-1})}{F(q^{-1})}u(k) + \frac{C(q^{-1})}{D(q^{-1})}e(k), \tag{3.1}$$

where

$$\begin{aligned}
A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a}, \\
B(q^{-1}) &= b_0 + b_1 q^{-1} + \ldots + b_{n_b} q^{-n_b},\ b_0 \neq 0 \\
C(q^{-1}) &= 1 + c_1 q^{-1} + \ldots + c_{n_c} q^{-n_c}, \\
D(q^{-1}) &= 1 + d_1 q^{-1} + \ldots + d_{n_d} q^{-n_d}, \\
F(q^{-1}) &= 1 + f_1 q^{-1} + \ldots + f_{n_f} q^{-n_f},
\end{aligned} \tag{3.2}$$

are polynomials defined in the backward shift operator q^{-1}, and $y(k)$, $u(k)$ and $e(k)$ are the output, input, and zero mean white noise respectively.

3.1 Model Selection

Several linear models can be derived from equation (3.1), but this thesis focuses on the AutoRegressive with eXogeneous input (ARX) and the AutoRegressive Moving Average with eXogeneous input (ARMAX) models that model adequately most situations of interest in Adaptive Control.

The AutoRegressive with eXogeneous input (ARX) model admits the following structure

$$A(q^{-1})y(k) = B(q^{-1})u(k) + e(k), \tag{3.3}$$

hence, $C = D = F = 1$. The parameters a_i and b_i in the model are the system coefficients. Using equation (3.3), the time-domain output at time k is

$$\begin{aligned}
y(k) = -a_1 y(k-1) - \ldots - a_{n_a} y(k-n_a) + \\
+ b_0 u(k) + b_1 u(k-1) + \ldots + b_{n_b} u(k-n_b) + e(k),
\end{aligned} \tag{3.4}$$

that can be written in vector form as

$$\begin{aligned}
y(k) &= \begin{bmatrix} -y(k-1) & \ldots & -y(k-n_a) & u(k) & u(k-1) & \ldots & u(k-n_b) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{n_a} \\ b_0 \\ b_1 \\ \vdots \\ b_{n_b} \end{bmatrix} + e(k) \\
&= \varphi^T(k)\theta(k) + e(k).
\end{aligned} \tag{3.5}$$

The regression vector $\varphi(k)$ consists of lagged outputs and inputs, and the vector θ contains the unknown parameters, that are to be estimated.

The ARMAX model is represented by the following structure

$$A(q^{-1})y(k) = B(q^{-1})u(k) + C(q^{-1})e(k), \tag{3.6}$$

where the difference to the ARX model is the polynomial $C(q^{-1})$.

Many industrial applications are affected by non stationary disturbances (random steps occurring at random times and Brownian motion), and it has been argued that integrated models like Controlled AutoRegressive Integrated Moving Average (CARIMA) are more appropriate as they intrinsically account for integral action. This model is further explained in Chapter 5.

3.2 Parameter Estimation

A process is described by a set of discrete-time measurements that characterizes the time evolution of its output in response to a known input. This section provides the generic approach for parameters estimation in system identification using the two most known recursive algorithms, the RLS and the LMS. They are important not only to understand what is the usual approach for system identification, but also to serve as a baseline for comparison with other algorithms.

3.2.1 Recursive Least Squares

The LS method consists of driving the parameter estimates of a given model to best fit a data set according to a quadratic criterion. Consider the ARX model in equation (3.5). Given n observations the LS method optimum solution

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{2} \sum_{k=1}^{n} \| y(k) - \varphi^T(k)\theta \|_2^2, \tag{3.7}$$

corresponds to the minimum of the sum of the squared errors, defined as the difference between the true value of the parameter and the estimated by the method. Provided that the inverse exists, a closed form solution is obtained as

$$\hat{\theta} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\boldsymbol{Y}, \tag{3.8}$$

where

$$\begin{aligned} \mathbf{\Phi} &= \begin{bmatrix} \varphi^T(1) & \dots & \varphi^T(n) \end{bmatrix}^T, \\ \boldsymbol{Y} &= \begin{bmatrix} y(1) & \dots & y(n) \end{bmatrix}^T. \end{aligned} \tag{3.9}$$

Yet this batch LS method has the disadvantage of being an offline method, requiring all the data to be known before the estimation process which is not admissible in the context of this work. Furthermore, each time that new information is available, the model parameters need to be calculated by recomputing (3.8). As time progresses, this procedure becomes unreasonable both because it turns computationally impractical, and the memory required to store the full data set grows linearly with it. To deal with these negative considerations a RLS method can be defined, combining previous estimates with new data acquired.

As new observations become available, the issue is how to find the current estimate $\hat{\theta}(k)$ as a function of the previous estimate $\hat{\theta}(k-1)$, the current system output $y(k)$, and the system input $u(k-1)$ in a recursive fashion.

Let $\mathbf{\Phi}(k)$ and $\boldsymbol{Y}(k)$ be

$$\begin{aligned} \mathbf{\Phi}(k) &= \begin{bmatrix} \mathbf{\Phi}(k-1) & \varphi^T(k) \end{bmatrix}^T, \\ \boldsymbol{Y}(k) &= \begin{bmatrix} \boldsymbol{Y}(k-1) & y(k) \end{bmatrix}^T. \end{aligned} \tag{3.10}$$

The LS closed form solution in (3.8) can also be written as

$$\hat{\theta}(k) = \Lambda^{-1}(k) \sum_{t=1}^{k} y(t)\varphi(t), \tag{3.11}$$

or

$$\Lambda(k)\hat{\theta}(k) = \sum_{t=1}^{k} y(t)\varphi(t), \tag{3.12}$$

where

$$\Lambda(k) = \sum_{t=1}^{k} \varphi(t)\varphi^T(t) \tag{3.13}$$

is the information matrix that verifies

$$\Lambda(k) = \Lambda(k-1) + \varphi(k)\varphi^T(k). \tag{3.14}$$

Combining (3.12), (3.13), and (3.14) leads to an equation of the form

$$\hat{\theta}(k) = \hat{\theta}(k-1) + \Lambda^{-1}(k)\varphi(k)\left[y(k) - \varphi^T(k)\hat{\theta}(k-1)\right]. \tag{3.15}$$

As it stands, the algorithm is not well suited for computation given that at each time instant, a matrix needs to be inverted. To avoid the matrix inversion in each iteration the covariance matrix can be computed in spite of the information matrix

$$P(k) = \left[\Lambda(k-1) + \varphi(k)\varphi^T(k)\right]^{-1}. \tag{3.16}$$

Applying the Matrix Inversion Lemma (see Appendix 6) with

$$\mathbf{A} = P(k-1), \quad \mathbf{B} = \varphi(k), \quad \mathbf{C} = 1, \quad \mathbf{D} = \varphi^T(k), \tag{3.17}$$

and a few matrix manipulations, the RLS equations are then

$$\begin{aligned} \varepsilon(k) &= y(k) - \varphi^T(k)\hat{\theta}(k-1), \\ \hat{\theta}(k) &= \hat{\theta}(k-1) + K(k)\varepsilon(k), \\ K(k) &= P(k)\varphi(k), \\ P(k) &= P(k-1) - \frac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{1 + \varphi^T(k)P(k-1)\varphi(k)}, \end{aligned} \tag{3.18}$$

where $\varepsilon(k)$ is known as the prediction error, and K is the so-called Kalman Gain that measures how much the estimate should be changed given a certain measurement. If $P(k)$ is large, it means that the state is estimated to change a lot which requires changing the estimates with new observations. As a result the Kalman Gain is high. On the other hand, if $P(k)$ is small then the state does not change much therefore the Kalman Gain is low. A common choice for the initial values is to take $P(0) = cI$, and $\hat{\theta}(0) = 0$ where c is a large constant and I the Identity Matrix. Initial estimates of $P(k)$ and $\hat{\theta}(k)$ are enough to run RLS. The block diagram with the setup previously detailed is depicted in Figure 3.1.

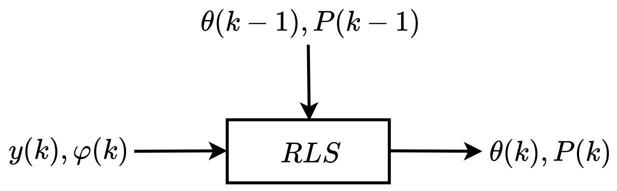

Figure 3.1: RLS Block Diagram

3.2.2 Variable Forgetting Factor

When the RLS converges, the Kalman gain $K(k)$ decreases, and the algorithm becomes insensitive to parameter variations when identifying time-varying systems. To tackle this difficulty, a cost function with weights exponentially decaying in the past information direction is used

$$E(k) = \frac{1}{2}\sum_{i=0}^{k} \beta^{k-i}\left(y(i) - \varphi^T(i)\theta(i)\right)^2, \tag{3.19}$$

where $0 < \beta < 1$ is known as the forgetting factor. When the forgetting factor is very close to one, the algorithm achieves low mis-adjustment and good stability, but its tracking capabilities are reduced. On the other hand, a smaller value of the forgetting factor improves tracking performance but also increases the mis-adjustment which may affect the stability of the algorithm. The asymptotic memory N gives the idea of the number of past data affecting the current estimate $N = \dfrac{1}{1-\beta}$. The RLS with exponential forgetting factor equations are

$$\begin{aligned}
\varepsilon(k) &= y(k) - \varphi^T(k)\hat{\theta}(k-1), \\
\hat{\theta}(k) &= \hat{\theta}(k-1) + K(k)\varepsilon(k), \\
K(k) &= P(k)\varphi(k), \\
P(k) &= \beta^{-1}\left[P(k-1) - \frac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{\beta + \varphi^T(k)P(k-1)\varphi(k)}\right].
\end{aligned} \tag{3.20}$$

A fixed exponential forgetting factor can still be inadequate as the algorithm is prone to estimator windup or covariance blow-up. Under normal operating conditions the forgetting factor acts to increase the size of the covariance matrix $P(k)$ while the regression vector $\varphi(k)$ acts to decrease it, achieving a balance between alertness to parameters variations, and convergence. However when the input is not persistent or it decreases significantly in amplitude, below a given level where the system is not excited anymore (system is at equilibrium), the regression vector $\varphi(k)$ holds no new information. In this situation, $P(k) = \beta^{-1}P(k-1)$ and since $\beta < 1$ the covariance matrix increases in size leading to unreliable and biased estimates. There also exist some problems with exponential forgetting which can cause strong misfitting of the estimations. One of these problems is when the parameters are kept constant over a long period of time and then an abrupt variation occurs. At the time of the variation, the gain is too low and the estimates converge too slowly to their true values.

Defining a measure of the information content as the weighted sum of the squares of the *a posteriori* errors

$$\Sigma(k) = \beta(k)\Sigma(k-1) + (1 - \varphi^T(k)K(k))\varepsilon^2(k). \tag{3.21}$$

A well-known method of choosing the forgetting factor, described in [51], consists of keeping the information content of the algorithm constant $\Sigma(k) = \Sigma(k-1) = \ldots = \Sigma_0$.

Knowing that $\beta(k) = 1 - \frac{1}{N(k)}$, and using equation (3.21) results in $N(k) = \frac{\Sigma_0}{[1 - \varphi^T(k)K(k)]\,\varepsilon^2(k)}$. The value of Σ_0 is set equal to the expected measurement noise variance (assuming there is some kind of knowledge about the process) times a nominal asymptotic memory length, $\Sigma_0 = \sigma_e^2 N_0$.

If $\beta(k) < \beta_{\min}$ then $\beta(k) = \beta_{\min}$. Therefore, at each time instant, the forgetting factor is as follows

$$\beta(k) = \max\left\{\beta_{\min}, 1 - \left[1 - \varphi^T(k)K(k)\right]\frac{\varepsilon^2(k)}{\Sigma_0}\right\}, \tag{3.22}$$

3.2.3 Least Mean Square

Whereas the RLS is based on the minimization of a linear least squares, the LMS algorithm [52] derives from the minimization of the cost

$$J(k) = \frac{1}{2}|e(k)|^2, \tag{3.23}$$

where $e(k)$ is the instantaneous error, equal to the difference of the desired and the estimated output

$$e(k) = y(k) - \varphi^T(k)\theta(k). \tag{3.24}$$

The LMS is very simple to implement, requiring low computational complexity. However, compared to the RLS, it has a lower convergence rate, and it behaves poorly in presence of coloured signals.

Since this thesis deals with system identification where the input signal is always real-valued, the use of the conjugate operator is omitted and the non-conjugate matrix transpose ($(\cdot)^T$) is used instead of the Hermitian transposition ($(\cdot)^H$).

The gradient vector of $J(k)$ can be expressed as

$$\frac{\partial J(k)}{\partial \theta(k)} = -r(k) + R(k)\theta(k), \tag{3.25}$$

where $r(k)$ and $R(k)$ are, respectively, the cross-correlation vector between the regression vector and the output, and the regression vector correlation matrix. The gradient vector is then characterized using instantaneous estimations of $r(k)$ and $R(k)$

$$\begin{aligned} r(k) &= \varphi(k)y(k), \\ R(k) &= \varphi(k)\varphi^T(k). \end{aligned} \tag{3.26}$$

Thus, equating equation (3.25) to zero leads to a parameter vector update given by

$$\begin{aligned} \theta(k) &= \theta(k-1) - \mu\frac{\partial J(k)}{\partial \theta(k)} \\ &= \theta(k-1) + \mu e(k)\varphi(k), \end{aligned} \tag{3.27}$$

where μ is the step size that balances the algorithm convergence and steady-state error. The convergence condition of LMS is imposed by

$$0 < \mu < \frac{1}{\lambda_{\max}}, \tag{3.28}$$

where λ_{max} is the maximum eigenvalue of the regression vector covariance matrix.

From equation (3.27) it is visible that the parameters adjustment is directly proportional to the regression vector $\varphi(k)$. When $\varphi(k)$ is large, the LMS undergoes a gradient noise amplification. This problem is circumvented if the update applied to the parameter vector at each time step is normalized, with respect to the squared euclidean norm of $\varphi(k)$. This modification entails the well-known Normalized Least Mean Square (NLMS) algorithm

$$\theta(k) = \theta(k-1) + \mu \frac{e(k)\varphi(k)}{\alpha + \varphi^T(k)\varphi(k)}, \tag{3.29}$$

where the regularization parameter α is used to prevent division by zero, especially during initialization when $\varphi(k) = 0$.

4 Sparsity Aware Recursive Algorithms

Contents

In this chapter the estimation process of a sparse parameters vector performed by several recursive algorithms is described.

4.1 RZA-NLMS

Motivated by LASSO, the ℓ_1 norm optimization strategy is incorporated in the NLMS cost function in order exploit the sparsity of the system. The objective is to minimize the following cost function

$$J(k) = \frac{1}{2}e^2(k) + \gamma\|\theta(k)\|_1. \tag{4.1}$$

Calculating the subgradient of $J(k)$, and equating it to zero gives an update equation for the parameter vector, of the form

$$\begin{aligned}\hat{\theta}(k) &= \hat{\theta}(k-1) - \mu\frac{\partial J(k)}{\partial\hat{\theta}(k)} \\ &= \hat{\theta}(k-1) - \rho\,\mathrm{sgn}\left(\hat{\theta}(k-1)\right) + \mu\frac{e(k)\varphi(k)}{\alpha + \varphi^T(k)\varphi(k)},\end{aligned} \tag{4.2}$$

where $\mathrm{sgn}\left(\cdot\right)$ is the sign function, and $\rho = \mu\gamma$ controls the effect of the additional term $-\rho\,\mathrm{sgn}\left(\hat{\theta}(k-1)\right)$. First introduced in [24], this term is responsible for attracting the coefficients of θ to zero, hence the name zero-attractor.

Intuitively, the zero-attractor speeds-up convergence when most of the coefficients of θ are zero (sparse system) and ρ is a controllable parameter that determines the degree of zero attraction of the ℓ_1 norm for the parameter vector coefficients.

Based on the Zero-Attractor Least Mean Square (ZA-LMS), this algorithm gives biased estimates. Notwithstanding, an appropriate choice of ρ can be shown to give ZA-LMS results with a lower mean squared error in comparison to the standard LMS when sparse systems are considered [23, 24].

Regardless, forcing all the parameters to zero uniformly may be problematic, especially for less sparse systems, in which relevant coefficients should preserve their relevance. Weighting the zero-attractor can, however, avoid uniform shrinkage. A weighted algorithm can be derived either using the weighted ℓ_1 norm directly or through the log-sum penalty [53]. The Reweighted Zero-Attractor Normalized Least Mean Square (RZA-NLMS) will be derived using the latter while the following sections will address the former. Hence, the following cost is considered

$$J(k) = \frac{1}{2}e^2(k) + \gamma\sum_{j=1}^{n}\log\left(1 + \frac{|\hat{\theta}_j(k)|}{\epsilon}\right). \tag{4.3}$$

Here, the penalty term is no longer the ℓ_1 norm but the log-sum penalty, that behaves more similarly to the ℓ_0 pseudo-norm. Figure 4.1 shows the plots of the ℓ_0 pseudo-norm, the ℓ_1 norm, and the log-sum function with a value of ϵ such that they all intersect at the coordinates where the ordinate is equal to 1. In [54] it was shown that when $\epsilon = 0$, the log-sum penalty function is essentially the same as the ℓ_0 pseudo-norm. Thus, it is expected that it behaves similarly to the ℓ_0 pseudo-norm when ϵ is small. Furthermore, as $\epsilon \to 0$, it is more likely to converge to an undesirable local minimum, therefore adding the log-sum penalty function turns the problem into a non convex optimization problem with no guarantee that the algorithm converges to a global minimum. Subsequently, it is important to choose a suitable initial point if possible. In fact, the probability of finding a global minimum can be improved by starting from a different initialization points and choosing the converged point that achieves the minimum objective function value.

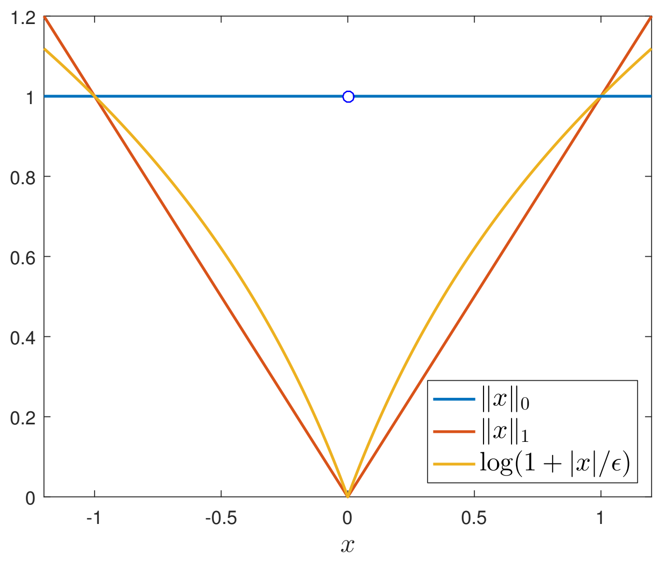

Figure 4.1: At the origin, the ℓ_0 pseudo-norm (in blue) is better approximated by the log-sum penalty (in yellow) function than by the traditional ℓ_1 norm (in red).

Additionally, a monotonically decreasing sequence $\{\epsilon^{(k)}\}$ can be considered, starting with a relatively large value to provide a stable estimate, and then its value can be decreased.

Using the log-sum penalty function the update equation for the parameter vector is then

$$\hat{\theta}(k) = \hat{\theta}(k-1) - \rho \frac{\operatorname{sgn}\left(\hat{\theta}(k-1)\right)}{1+|\hat{\theta}(k-1)|/\epsilon} + \mu \frac{e(k)\varphi(k)}{\alpha + \varphi^T(k)\varphi(k)}, \tag{4.4}$$

where $\rho = \dfrac{\mu\gamma'}{\epsilon'}$ and $\epsilon = \dfrac{1}{\epsilon'}$. The parameters with magnitudes comparable to $\dfrac{1}{\epsilon}$ are affected by the zero attractor, while on parameters with magnitudes larger than $\dfrac{1}{\epsilon}$ the zero attractor has small effect, introducing less bias in the estimates.

The algorithm outlined above is summarized in Algorithm 1.

Algorithm 1 RZA-NLMS

```
    $\mu$, $\rho$, $\alpha$                                            ▷ inputs
    $\gamma(0)$, $\beta$, $y(k)$, $\hat{\theta}(0) = 0$                  ▷ initialization
1: for $k = 1, 2, \ldots$ do                                   ▷ time recursion
2:     $\hat{\theta}(k) = \hat{\theta}(k-1) - \rho \frac{\operatorname{sgn}\left(\hat{\theta}(k-1)\right)}{1+|\hat{\theta}(k-1)|/\epsilon} + \mu \frac{e(k)\varphi(k)}{\alpha + \varphi^T(k)\varphi(k)}$
3: end for                                                  ▷ end of recursion
```

4.2 ℓ_1-RLS

The RLS cost function of equation (3.19) is modified as follows

$$J(k) = E(k) + \gamma(k) f\left(\hat{\theta}(k)\right), \tag{4.5}$$

where $f: \mathbb{R}^n \to \mathbb{R}$ is a convex function, and $\gamma(k) \geq 0$ is a regularization parameter that trades-off the amount of regularization and the estimation error.

Assuming that f is a generic nondifferentiable function, and that $E(k)$ is differentiable everywhere, then the subgradient is a possible alternative for the gradient computation. Recalling the definition, one valid subgradient vector of the cost function in equation (4.5) is

$$\nabla^s J(k) = \nabla E(k) + \gamma(k) \nabla^s f\left(\hat{\theta}(k)\right). \tag{4.6}$$

To find the optimal $\hat{\theta}(k)$, the subgradient $\nabla^s J(k)$ is set to zero. The following equation shows the relation for the i^{th} term

$$\sum_{m=0}^{k} \beta^{k-m} \left\{ y(m) - \sum_{j=1}^{n} \hat{\theta}_j(k) \varphi(m-j) \right\} \varphi(m-i) = \gamma(k) \left\{ \nabla^s f\left(\hat{\theta}(k)\right) \right\}_i. \tag{4.7}$$

The same equation can be written for all $i = 1, \ldots, n$ and put in matrix form as a system of modified normal equations

$$\Phi(k)\hat{\theta}(k) = r(k) - \gamma(k) \nabla^s f\left(\hat{\theta}(k)\right). \tag{4.8}$$

where $\Phi(k) \in \mathbb{R}^{n \times n}$ and $r(k) \in \mathbb{R}^n$ are, respectively, the deterministic autocorrelation matrix for the regression vector $\varphi(k)$ and the deterministic cross-correlation estimate vector between $y(k)$ and $\varphi(k)$:

$$\begin{aligned} \Phi(k) &= \sum_{m=0}^{k} \beta^{k-m} \varphi(m) \varphi^T(m) = \beta \Phi(k-1) + \varphi(k)\varphi^T(k), \\ r(k) &= \sum_{m=0}^{k} \beta^{k-m} y(m) \varphi(m) = \beta r(k-1) + y(k)\varphi(k). \end{aligned} \tag{4.9}$$

Defining a new variable $\Theta(k) = \Phi(k)\hat{\theta}(k)$, equation (4.8) becomes

$$\Theta(k) = r(k) - \gamma(k) \nabla^s f\left(\hat{\theta}(k)\right). \tag{4.10}$$

Replacing $r(k)$ in equation (4.10) with the update recursion defined by equation (4.9) results in an update equation for $\Theta(k)$. The assumption that $\gamma(k-1)$ and $\nabla^s f\left(\hat{\theta}(k-1)\right)$ do not change substantially each time instant, allows for an update equation of the form

$$\Theta(k) \approx \beta \Theta(k-1) + y(k)\varphi(k) - \gamma(k-1)(1-\beta) \nabla^s f\left(\hat{\theta}(k)\right). \tag{4.11}$$

Note that this assumption implies time invariant and slow variation scenarios only. Using the Matrix Inversion Lemma (see appendix 6), the covariance matrix $P(k)$, inverse of the autocorrelation matrix $\Phi(k)$, yields the well-known update equation

$$P(k) = \beta^{-1} \left[P(k-1) - K(k) \varphi^T(k) P(k-1) \right], \tag{4.12}$$

where the Kalman gain vector is $K(k) = \dfrac{P(k-1)\varphi(k)}{\beta + \varphi^T(k)P(k-1)\varphi(k)}$. The relation $\Theta(k) = \Phi(k)\hat{\theta}(k) \Leftrightarrow \hat{\theta}(k) = P(k)\Theta(k)$ and the update in equation (4.11) allows writing an estimate, at each time instant, for $\theta(k)$

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)e(k) - \gamma(k-1)(1-\beta)P(k)\nabla^s f\left(\hat{\theta}(k)\right) \tag{4.13}$$

Generally, the suitable values of γ are found through repeated trials so as to reduce the steady-state error. To avoid tuning the regularization parameter for every distinct simulation, [27] proposes an automatic selection criterion for the value of $\gamma(k)$ when white a input is used.

At this point, the algorithm is generically detailed remaining only the choice of f. The true measure of sparsity is the non convex ℓ_0 pseudo-norm but, as seen in chapter 2, the ℓ_1 norm is known to be a great convex approximation. In this case, $f(\theta(k)) = \|\theta(k)\|_1 = \sum\limits_{i=1}^{n} |\theta_i(k)|$ whose subgradient is $\nabla^s\|\theta(k)\|_1 = \operatorname{sgn}(\theta(k))$, the sign function. Again, the use of the ℓ_1 norm penalty lead to biased estimates by uniformly affecting all the parameters, which prompts the development of a weighted ℓ_1 norm scheme

$$\|W\hat{\theta}(k)\|_1 = \sum_{j=1}^{n} w_j|\hat{\theta}_j(k)|. \tag{4.14}$$

The values $w_j,\ j = 1, 2, \ldots, n$ are positive weighting factors that affect the corresponding n parameters. W is a diagonal matrix with the values w_j on its main diagonal. The subgradient of the weighted ℓ_1 norm is then

$$\nabla^s\|W\hat{\theta}(k)\|_1 = W^T\operatorname{sgn}\left(W\hat{\theta}(k)\right), \tag{4.15}$$

but since W is a positive valued diagonal matrix, equation (4.15) becomes

$$\nabla^s\|W\hat{\theta}(k)\|_1 = W\operatorname{sgn}\left(\hat{\theta}(k)\right). \tag{4.16}$$

Finally, the weighting matrix is selected. In order to approximate the weighted ℓ_1 norm to the ℓ_0 pseudo-norm, the weights can be picked as inversely proportional to the actual value of the parameters. However, the parameters value are unknown and the best approximation available are their last estimates. Thereby,

$$w_j = \frac{1}{|\hat{\theta}_j(k-1)| + \epsilon}. \tag{4.17}$$

The parameter ϵ is positive and its purpose is to avoid numerical errors resulting from division by zero. In [53], it is shown that a value of lower order than the nonzero estimates is the more suitable choice. Equation (4.17) goes to show the relationship with the log-sum penalty mentioned earlier.

The algorithm outlined above is summarized in Algorithm 2.

Algorithm 2 ℓ_1-RLS

```
    $\beta$, $\epsilon$, $\varphi(k)$, $y(k)$                                  ▷ inputs
    $\gamma(0)$, $\hat{\theta}(0) = 0$, $P(0) = a\mathrm{I}$, $a = \text{constant}$      ▷ initialization
1: for $k = 1, 2, \ldots$ do                                   ▷ time recursion
2:     $K(k) = \frac{P(k-1)\varphi(k)}{\beta + \varphi^T(k)P(k-1)\varphi(k)}$
3:     $\varepsilon(k) = y(k) - \varphi^T(k)\hat{\theta}(k-1)$
4:     $P(k) = \beta^{-1}\left[P(k-1) - K(k)\varphi^T(k)P(k-1)\right]$
5:     $\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\varepsilon(k) - \gamma(k-1)(1-\beta)P(k)\frac{\operatorname{sgn}\left(\hat{\theta}(k-1)\right)}{|\hat{\theta}(k-1)| + \epsilon}$
6: end for                                                     ▷ end of recursion
```

The vector division operation in state 5 of Algorithm 2 denotes a simple element-wise division.

4.3 RLS-Weighted LASSO

The batch LASSO [21] in the lagrangian form yields an estimate given by

$$\hat{\theta}(k) = \arg\min_{\theta} \ \frac{1}{2}\|\boldsymbol{\mathcal{Y}}(k) - \boldsymbol{\Phi}(k)\theta\|_2^2 + \lambda(k)\|\theta\|_1, \tag{4.18}$$

for a sparse vector θ in a regression model of the type $\boldsymbol{\mathcal{Y}}(k) = \boldsymbol{\Phi}(k)\theta + \boldsymbol{e}(k)$ where the observations up until k are $\boldsymbol{\mathcal{Y}}(k) = \begin{bmatrix} y^T(1) & \ldots & y^T(k) \end{bmatrix}^T$ and $\boldsymbol{\Phi}(k) = \begin{bmatrix} \varphi^T(1) & \ldots & \varphi^T(k) \end{bmatrix}^T$. Even though it is non-differentiable, LASSO is also convex, hence being possible to obtain a global minimum through linear programming techniques. However as the value of k increases, the size of $\boldsymbol{\mathcal{Y}}(k)$ and $\boldsymbol{\Phi}(k)$ also increases, which leads to high complexity along with large memory requirements and computation time. In that regard, a recursive version is developed to process the observations sequentially. The Recursive LASSO (R-LASSO) [25, 26] objective is then formulated

$$J(k)^{\mathrm{R-LASSO}} = \frac{1}{2}\sum_{i=1}^{k} \beta^{k-i}\|y(i) - \varphi(i)\theta\|_2^2 + \lambda(k)\|\theta\|_1. \tag{4.19}$$

Recalling the setup for the RLS, (4.19) is equivalent to

$$\begin{aligned} \hat{\theta} &= \arg\min_{\theta} \ J(k)^{\mathrm{R-LASSO}} \\ &= \arg\min_{\theta} \ \frac{a(k) + \theta^T R(k)\theta - 2\theta^T r(k)}{2} + \lambda(k)\|\theta\|_1, \end{aligned} \tag{4.20}$$

with

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

Analogous to RLS the quantities in (4.20) can be recursively updated.

Proof.

$$\begin{aligned}
r(k+1) &= \sum_{i=1}^{k+1} \beta^{k+1-i}\varphi^T(i)y(i) = \sum_{i=1}^{k+1} \beta\beta^{k-i}\varphi^T(i)y(i) = \beta\sum_{i=1}^{k} \beta^{k-i}\varphi^T(i)y(i) + \beta\beta^{-1}\varphi^T(i)y(i)\\
&= \beta r(k) + \varphi^T(i)y(i),\\
R(k+1) &= \sum_{i=1}^{k+1} \beta^{k+1-i}\varphi^T(i)\varphi(i) = \sum_{i=1}^{k+1} \beta\beta^{k-i}\varphi^T(i)\varphi(i) = \beta\sum_{i=1}^{k} \beta^{k-i}\varphi^T(i)\varphi(i) + \beta\beta^{-1}\varphi^T(i)\varphi(i)\\
&= \beta R(k) + \varphi^T(i)\varphi(i).
\end{aligned}$$

Note that $a(k)$ is just a scalar therefore its updating is not relevant for the minimization procedure. □

Unlike RLS, gradient based minimization of $J(k)^{\mathrm{R-LASSO}}$ is impossible due to the non-differentiability of the ℓ_1 norm. To overcome it, the authors of [25, 26] use subgradient based iterative minimizers:

$$\theta(k)^{i+1} = \theta(k)^i - \alpha_i \check{\nabla} J(k), \tag{4.21}$$

where $\check{\nabla} J(k)$ denotes a sub-gradient of $J(k)^{\mathrm{R-LASSO}}$ at $\theta(k)^i$, being $\theta(k)^i$ the i^{th} iterate of $\theta(k)$ and $\alpha_i > 0$ the step size, which should be chosen with guarantee of convergence to the global minimum. Having $\alpha_i = \frac{\alpha}{\sqrt{i}}$ or $\alpha_i = \frac{\alpha}{i}$ guarantees it as $i \to \infty$, whilst $\alpha_i = \alpha$ guarantees convergence within some α-dependent range of the optimal value. The subgradient vector is then given by

$$\{\check{\nabla} J(k)^{\mathrm{R-LASSO}}\}_n = \begin{cases} \{\nabla L(k)\}_n + \lambda(k)\mathrm{sgn}(\theta_n), \text{ if } \theta_n \neq 0\\ \{\nabla L(k)\}_n - \lambda(k), \text{ if } \theta_n = 0, \{\nabla L(k)\}_n > \lambda(k)\\ \{\nabla L(k)\}_n + \lambda(k), \text{ if } \theta_n = 0, \{\nabla L(k)\}_n < -\lambda(k)\\ 0, \qquad \text{if } \theta_n = 0, -\lambda(k) < \{\nabla L(k)\}_n < \lambda(k) \end{cases}, \tag{4.22}$$

where $L(k) = \frac{\theta^T R(k)\theta - 2\theta^T r(k)}{2}$ is the differentiable LS objective function and $\nabla L(k) = R(k)\theta - r(k)$ its gradient. Granted that $\nabla L(k)$ can be recursively updated, the sub-gradient iterates can be updated with reasonable complexity at each time instant k. Even so sub-gradient methods have slow convergence, $\mathcal{O}\left(\frac{1}{\sqrt{k}}\right)$, and it is not guaranteed that the limit of $\hat{\theta}(k)$ necessarily converges to the true θ as $k \to \infty$. An alternative could be the use of proximal gradient methods.

LASSO continuously shrinks the coefficients toward 0 as λ increases, and some coefficients are exactly shrunk to 0 when λ is large enough. Continuous shrinkage can improve estimation accuracy due to the bias-variance tradeoff. It has been shown that variable selection with LASSO can be consistent if the underlying model satisfies specific conditions and that LASSO can perform automatic variable selection because the ℓ_1 penalty is singular at the origin [45]. However LASSO shrinkage produces biased estimates for the large coefficients, and thus it could be sub-optimal in terms of estimation risk, hence the oracle properties do not hold.

The R-LASSO cannot guarantee the recovery of the correct support and consistently estimate the non-zero entries of θ at the same time.

If $\sqrt{k} < \lambda(k) < k$ and the irrepresentable condition [55] is met, R-LASSO guarantees (2.16) but not (2.17). This happens because the rate of convergence is $\frac{k}{\lambda(k)}$ which is slower than $\sqrt{k}$ as in the second oracle property, thus $\left[\mathcal{S}(\hat{\theta}(k)) - \mathcal{S}(\theta)\right]$ diverges.

If $\lim_{k\to\infty} \frac{\lambda(k)}{\sqrt{k}} = \lambda_0 \geq 0$ then $\lim_{k\to\infty} \Pr[\mathrm{supp}(\hat{\theta}(k)) = \mathrm{supp}(\theta)] = g(\lambda_0) < 1$ with $g(\lambda_0)$ being an increasing function of λ_0. Since $g(\lambda_0)$ increases with λ_0, for $\lambda(k) \propto \sqrt{k}$ the limit in (2.16) is strictly less than one. In this case the oracle properties are not fulfilled yet the estimates are at least asymptotically unbiased.

There is no $\lambda(k)$ for which the oracle properties simultaneously hold therefore the R-LASSO cannot simultaneously estimate the signal support and the non-zero parameters values consistently. To bypass these considerations, the ℓ_1 weighted version of the R-LASSO is shown next.

To hinder the negative results of R-LASSO a penalty function is sought to. It should weigh the ℓ_1 norm term $|\theta_n|$ individually and be signal dependent. A desirable function [25] is then

$$w_{\mu(k)}(|\theta|) = \frac{(a\mu(k) - |\theta|)_+}{\mu(k)(a-1)} u(|\theta| - \mu(k)) + u(\mu(k) - |\theta|), \tag{4.23}$$

with $u(\cdot)$ being the step function and $(\cdot)_+$ the non-negative part of the function in parentheses. In [25] the parameter a is set to 3.7 as in the Smoothly Clipped Absolute Deviation penalty [46]. A proper choice of values for $\mu(k)$ and a grant the opportunity to set unwanted or irrelevant coefficients to zero, while not affecting the remaining ones.

While on R-LASSO all the elements $|\theta_n|$ are identically weighted ($w = 1$), this weight function proves the estimator with higher weights on small entries, and lower weights on entries with large amplitudes. More precisely, as depicted in Figure 4.2, elements of size less than $\mu(k)$ are penalized as in R-LASSO, elements between $\mu(k)$ and $a\mu(k)$ have a linearly decreasing penalization, and elements larger than $a\mu(k)$ are not penalized at all. The weight function is updated using RLS estimates that are computed alongside.

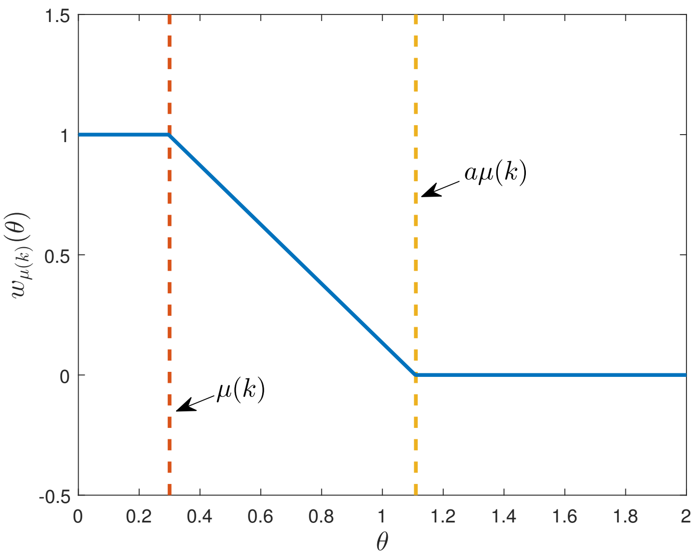

Figure 4.2: Weight function of equation (4.23) for values of $\mu(k) = 0.3$ and $a = 3.7$.

Plugging (4.23) in the cost function of the R-LASSO gives

$$J(k)^{\mathrm{RW-LASSO}} = \frac{1}{2}\sum_{i=1}^{k} \beta^{k-i} \|y(i) - \varphi(i)\theta\|_2^2 + \lambda(k) \sum_{n=1}^{N} w_{\mu_k}(|\widehat{\theta}_n^{\mathrm{RLS}}(k)|)|\theta_n|, \tag{4.24}$$

where $J(k)^{\mathrm{RW-LASSO}}$ is the new objective function to minimize. The now called RLS-Weighted LASSO (RW-LASSO) holds an estimate given by

$$\hat{\theta} = \arg\min_{\theta} \quad J(k)^{\mathrm{RW-LASSO}}, \tag{4.25}$$

which can also be updated similarly to R-LASSO. Although they involve affordable complexity, sub-gradient methods have slow convergence, the proximity operator referenced in Chapter 2 is applied instead.

For the specific case of LASSO where $h(\theta) = \lambda\|\theta\|_1$, the main step consists of computing

$$\mathrm{Prox}_{\alpha_i\|\cdot\|_1}(\theta^i - \alpha_i \nabla g(\theta^i)) = \mathcal{S}_\eta(\theta^i - \alpha_i \nabla g(\theta^i)) \tag{4.26}$$

where $[\mathcal{S}_\eta(\theta)]_n = \mathrm{sign}(\theta_n)(|\theta_n| - \eta)_+$ with $\eta = \alpha_i\lambda$ known as the element-wise Soft-Thresholding function. At each iteration the cost function is linearized around the current point. Since the weight function in (4.23) yields values between 0 and 1, if it is used in the ℓ_1 weighted-norm $\lambda\|w\theta\|_1$ then we can admit a new regularization parameter equal to $\gamma = \lambda w$, and $h(\theta) = \gamma\|\theta\|_1$. The proximal operator of $h(\theta)$ is the element-wise Soft-Thresholding function now with $\tau = \gamma\alpha = \lambda w\alpha$ for the ℓ_1 weighted recursive LASSO.

Still it is somewhat more complex than the R-LASSO since at each time instant k, an RLS estimate must be computed to update the weights which will be used at the same instant. With a forgetting factor $\beta = 1$, a penalty parameter $\sqrt{k} < \lambda(k) < k$ and a parameter $\mu_k = \dfrac{\lambda(k)}{k}$, the RW-LASSO fulfill the oracle properties [25]. Choosing $\beta < 1$ enables both R-LASSO and RW-LASSO with tracking capabilities for time-varying parameter vectors, yet the oracle properties are not met in this scenario.

The algorithm outlined above is summarized in Algorithm 3.

Algorithm 3 RLS-Weighted LASSO

```
    $\beta$, $\epsilon$, $\varphi(k)$, $y(k)$, $a$, $\alpha$, $i_{\max}$                                      ▷ inputs
    $r(0) = 0$, $R(0) = 0$, $\lambda(0)$, $\hat{\theta}(0) = 0$, $P(0) = \delta\mathrm{I}$, $\delta = \text{constant}$          ▷ initialization
 1: for $k = 1, 2, \ldots$ do                                                  ▷ time recursion
 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:     $K(k) = \dfrac{P(k-1)\varphi(k)}{\beta + \varphi^T(k)P(k-1)\varphi(k)}$
 6:     $P(k) = \beta^{-1}\left[P(k-1) - K(k)\varphi^T(k)P(k-1)\right]$
 7:     $\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)\varphi(k)\varepsilon(k)$
 8:     $w_{\mu(k)}(|\theta|) = \dfrac{(a\mu(k) - |\theta|)_+}{\mu(k)(a-1)} u(|\theta| - \mu(k)) + u(\mu(k) - |\theta|)$
 9:     for $i = 1, \ldots, i_{\max}$ do                                      ▷ iterative method
10:         $\mathrm{Prox}_{\alpha_i\|\cdot\|_1}(\theta^i - \alpha_i \nabla g(\theta^i)) = \mathcal{S}_\eta(\theta^i - \alpha_i \nabla g(\theta^i))$
11:     end for                                                       ▷ end of iterative method
12: end for                                                           ▷ end of recursion
```

4.4 Results

The performance of the algorithms adopted in this chapter is now evaluated against both time invariant and time varying systems. Since no control strategy is being considered yet, the algorithms are analyzed *per se*. Consequently, all simulations are performed using stable systems. To generate this type of systems, the MATLAB command *drss* is used.

The input signals applied to the systems may vary between Gaussian, binary, and uniformly distributed pseudo random signals. Gaussian signals are generated using the MATLAB command *randn*, while the command *idinput* generates the others.

To simulate Gaussian white noise with zero mean the command *randn* is used. The noise sequence is then added to the noiseless output signal as discussed in Chapter 3.

Unless stated otherwise, the models used to validate the sparsity enforced feature of the algorithms in this section will be of order 8, that is, their transfer function has 17 parameters:

$$H(q^{-1}) = \frac{b_0 + b_1q^{-1} + b_2q^{-2} + b_3q^{-3} + b_4q^{-4} + b_5q^{-5} + b_6q^{-6} + b_7q^{-7} + b_8q^{-8}}{1 + a_1q^{-1} + a_2q^{-2} + a_3q^{-3} + a_4q^{-4} + a_5q^{-5} + a_6q^{-6} + a_7q^{-7} + a_8q^{-8}}. \tag{4.27}$$

4.4.1 Case I: Time Invariant Scenario

The first system has the following transfer function

$$H(q^{-1}) = \frac{-0.7336 - 0.3455q^{-1} - 0.0605q^{-2}}{1 + 0.5171q^{-1} + 0.0966q^{-2}}, \tag{4.28}$$

and it is therefore a second order system. The sparse system to be identified has a total of 17 parameters and 5 of them are nonzero. The input signal is assumed to be white, and Gaussian white observation noise with variance $\sigma_e^2 = 0.01$ is added to the system output. The algorithms are simulated over 1000 data points.

The parameters for the different algorithm are chosen as below:

- RZA-NLMS: $\mu = 1.7$, $\rho = 1.2 \times 10^{-3}$
- ℓ_1-RLS : $\beta = 0.9995$, $\gamma = 0.3$, $\epsilon = 0.1$
- RW-LASSO: $\beta = 0.999$
- RLS: $\beta = 1$

The estimation results are depicted in Table 4.1.

The first comment about the final estimated parameters is related to the algorithms that do not show zero valued estimates. For the RLS it was already expected a no zero value to parameters that actually vanish, opposed to the RZA-NLMS algorithm. Based on the LMS filter, they both have similar convergence speed, and since the number of data points is not high enough for it to reach the convergence values, a truly sparse solution is not achieved. The ℓ_1-RLS and the RW-LASSO both have sparse solutions. As it can be seen, RW-LASSO not only shrinks the parameters that were supposed to be zero, but also a few of those that have insignificant values. In this regard,

	True Values	RZA-NLMS Estimated Values	ℓ_1-RLS Estimated Values	RW-LASSO Estimated Values	RLS Estimated Values
Numerator Parameters (b_i)	-0.7336	-0.7042	-0.7324	-0.7319	-0.7334
	-0.3455	-0.1322	-0.2562	-0.2497	-0.3124
	-0.0605	0.0615	0.0042	0	-0.0104
	0	-0.0406	0	0	0.0383
	0	0.0384	0	0	-0.0101
	0	0.0249	0	0	-0.0329
	0	0.0732	0	0	0.0011
	0	-0.0666	0	0	0.0259
	0	0.0238	0	0	0.0196
Denominator Parameters (a_i)	0.5171	0.3764	0.3960	0.3876	0.4720
	0.0966	0.0094	0.0047	0	0.0317
	0	0.0036	0	0	-0.0519
	0	0.0229	0	0	0.0117
	0	0	0	0	0.0448
	0	0.0241	0	0	0.0019
	0	-0.0346	0	0	-0.0372
	0	0.0113	0.0023	0	0.0089

Table 4.1: Estimation results of a second order system after 1000 data points.

the shrinkage of these low valued parameters or not, depends critically on the choice of the tuning parameters, as a more fine tuning can lead to different outcomes.

Figure 4.3 displays the impulse response of the system from equation (4.28) and the impulse response of the system estimated by the RW-LASSO algorithm.

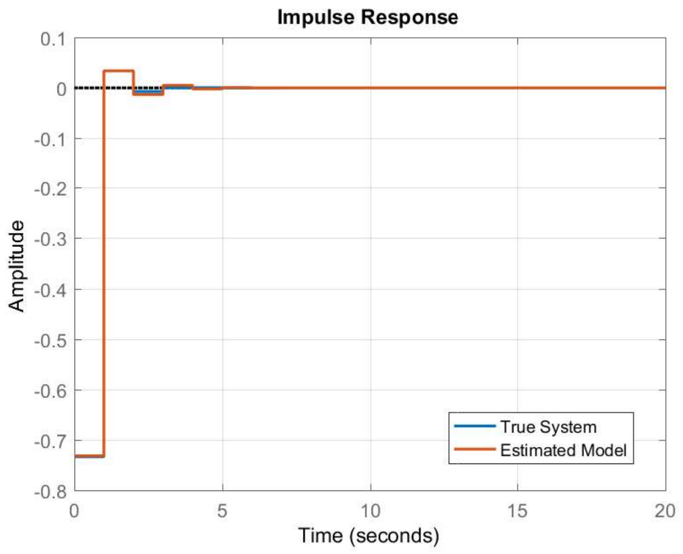

Figure 4.3: Impulse response of the system. The blue color represent the system of equation (4.28) and the orange color represent the estimated system using the RW-LASSO algorithm.

The analysis of the impulse responses should take into account two considerations: the first is the order of the system, and second the similarity between the real and the estimated coefficients. Figure 4.3 shows that the two impulse responses match one another. Therefore the systems highly match due to correct order estimation and correct transfer function parameters.

To also analyze the performance of the algorithms for different levels of sparsity, a new system of order 4 is generated. The system admits the following transfer function

$$H(q^{-1}) = \frac{0.1630 - 0.1051q^{-1} + 0.0938q^{-2} - 0.1762q^{-3} + 0.0595q^{-4}}{1 - 0.8674q^{-1} - 0.5737q^{-2} + 0.6447q^{-3} - 0.1360q^{-4}}. \tag{4.29}$$

In this simulation the input signal is a Pseudo Random Binary Signal the other inputs remain equal.

The values of the parameters of each algorithm were obtained via repeated simulations. The parameters for the different algorithm are as follows:

- RZA-NLMS: $\mu = 0.8$, $\rho = 5 \times 10^{-4}$
- ℓ_1-RLS : $\beta = 0.9995$, $\gamma = 0.3$, $\epsilon = 0.05$
- RW-LASSO: $\beta = 0.999$
- RLS: $\beta = 1$

The estimation results are depicted in Table 4.2. They corroborate the poor ability of RZA-NLMS to shrink the parameters exactly to zero.

	True Values	RZA-NLMS Estimated Values	ℓ_1-RLS Estimated Values	RW-LASSO Estimated Values	RLS Estimated Values
Numerator Parameters (b_i)	0.1630	0.1409	0.1634	0.1651	0.1648
	-0.1051	-0.1251	-0.1005	-0.0840	-0.1109
	0.0938	0.0738	0.0896	0.0794	0.0997
	-0.1762	-0.1387	-0.1737	-0.1728	-0.1854
	0.0595	0.0326	0.0505	0.0376	0.0666
	0	0.0277	0	0	-0.0064
	0	-0.0321	0.0013	0	0.0054
	0	0.0198	0.0045	0	0.0032
	0	-0.0227	0	0	-0.0120
Denominator Parameters (a_i)	-0.8674	-0.8463	-0.8168	-0.7674	-0.8790
	-0.5737	-0.5244	-0.6302	-0.6499	-0.5540
	0.6447	0.6092	0.6026	0.5573	0.6116
	-0.1360	-0.1149	-0.0920	-0.0746	-0.1279
	0	0.0402	0	0	0.0152
	0	0.0103	0	0	-0.0026
	0	-0.0202	0	0	0.0155
	0	-0.0187	0	0	-0.0152

Table 4.2: Estimation results of a fourth order system after 1000 data points.

Figure 4.4 displays the impulse response of the system from equation (4.29) and the impulse response of the system estimated by the RW-LASSO algorithm. In this case, they do not perfectly match. A logical explanation is the underestimation of the system parameters. Taking into account the values observed in Table 4.2, it is clear that the order of the system match the true order but there is a difference between the real and the estimated coefficients values, that results in a difference of impulse responses. If the two impulse responses match one another then the systems match either due to correct order estimation and correct transfer function parameters.

Consistent with the results of Table 4.1, the ℓ_1-RLS yields the best performance among all algorithms followed by the RW-LASSO. These results confirm that the sparse inducing algorithms outperform regular RLS, with the LMS based one being an exception.

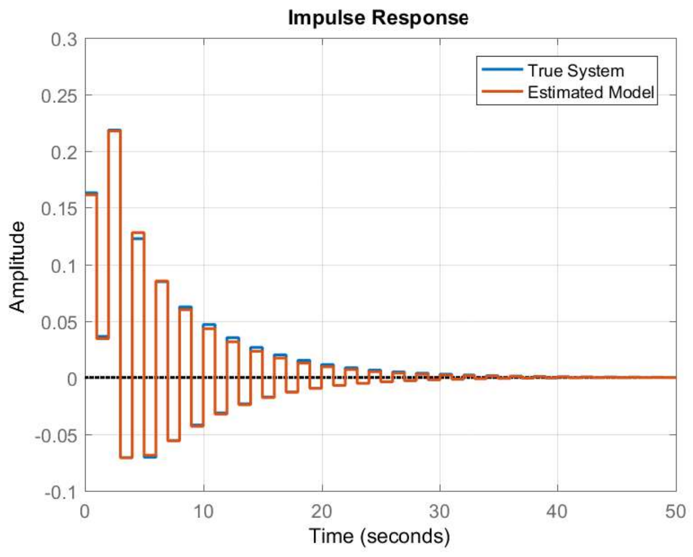

Figure 4.4: Impulse response of the system. The blue color represent the system of equation (4.29) and the orange color represent the estimated system using the RW-LASSO algorithm.

Usually, the bias of an estimator is a decreasing function of model complexity, while the variance is an increasing function of the model complexity. Figure 4.5 shows values of bias and variance according to different model complexities, in particular different sizes of the parameter vector.

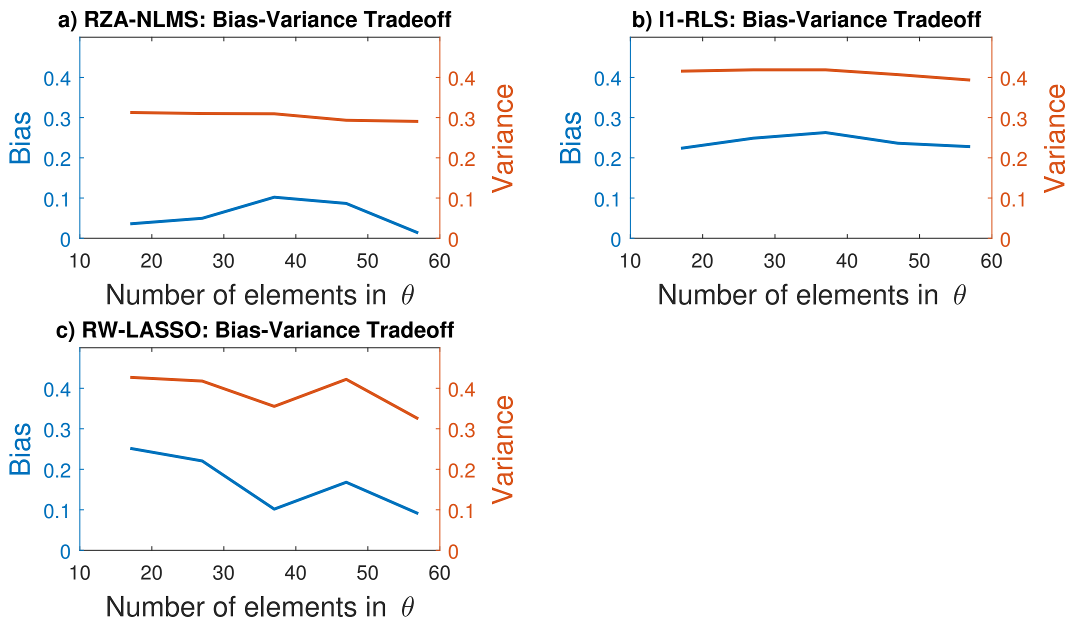

Figure 4.5: Estimators bias and variance as function of θ size.

Even though vectors of higher dimensions are considered, both bias and variance do not change considerably when sparse aware estimators are employed. The explanation lies in the zero attractor terms that reduce the estimates degrees of freedom. In that end, it is possible to assert that for sparse aware algorithms, the use of a large model or a small one, does not have a significant impact on the bias and variance tradeoff.

The same system is estimated with observations affected by different noise levels. Figure 4.6 depicts the MSE and the ℓ_1 error of θ for different values of noise variance σ_e^2. Figure 4.6 a) corresponds

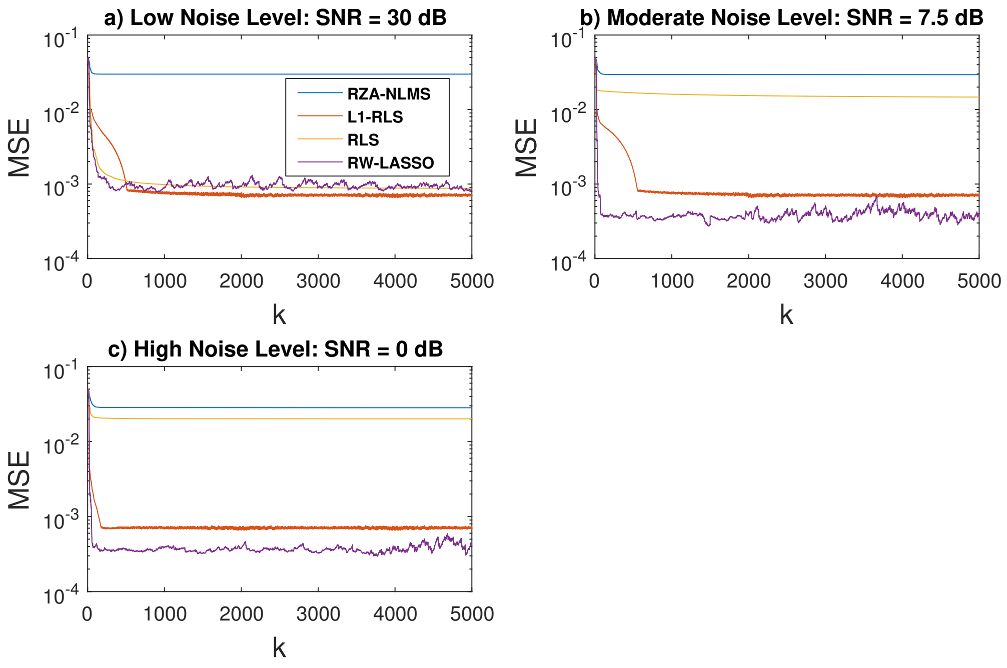

Figure 4.6: MSE values of the algorithms for different levels of noise.

to high noise level of $\mathrm{SNR} = 0\,\mathrm{dB}$ while Figures 4.6 b) and c) correspond to moderate and low noise levels, respectively, $\mathrm{SNR} = 7.5\,\mathrm{dB}$ and $\mathrm{SNR} = 30\,\mathrm{dB}$. In all scenarios, the RZA-NLMS has the highest MSE values. Followed by RLS, only in the low noise case, it becomes near the other two algorithms.

The percentage of correctly identified zero elements, as a function of the number of parameters of θ is given in Figure 4.7. Figure 4.7 also portrays the ℓ_1 parameter estimation error ($\|\hat{\theta} - \theta_0\|_1$) for each of the three SNR values mentioned previously.

In regards to complexity, the RZA-NLMS shows the least burden computationally ($\mathcal{O}(P)$ operations). The ℓ_1-RLS has similar complexity to RLS ($\mathcal{O}(P^2)$). The most expensive is the RW-LASSO with complexity proportional to $\mathcal{O}(rP^2)$, where r is the number of iterations used in the iterative method.

In summary, the results obtained in the previous simulations demonstrated that, even though it outperforms LMS, the RZA-NLMS algorithm acts poorly in comparison to RLS based methods.

When comparing ℓ_1-RLS and RW-LASSO, it seems that the former is better in respect to tracking performance and complexity. However the latter is better regarding the subset selection, mainly because of the soft-thresholding rule introduced by the proximity operator.

4.4.2 Case II: Time varying Scenario

The tracking capability and the convergence speed of an algorithm do not relate, at all, to each other. An algorithm with good converging properties can have terrible tracking capabilities in time varying scenarios, or the contrary. Thereby the performance of the adopted algorithms to track time varying sparse systems needs to be analyzed. The analysis includes the systems withstanding changes in their parameters. These variations can be slow varying, or either sudden changes of

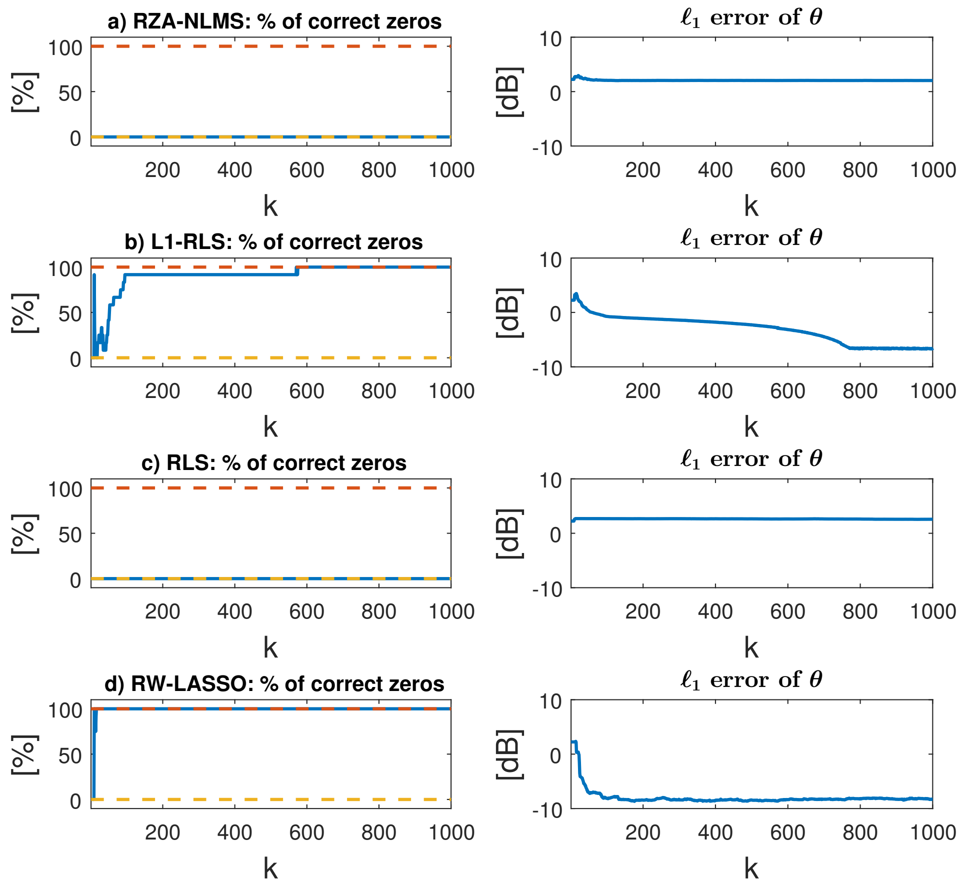

Figure 4.7: Percentage of correctly estimated zero elements of θ and ℓ_1 parameter estimation error.

the parameters values or changes of the support of the sparse system.

The following two simulations intend to illustrate the effect of variable forgetting, mentioned in Chapter 3. In this case, the generated system admits the following transfer function

$$H(q^{-1}) = \frac{1.6131q^{-1} - 1.0263q^{-2} + 0.1140q^{-3}}{1 - 0.8103q^{-1} + 0.1746q^{-2} - 0.0114q^{-3}}. \tag{4.30}$$

Here the parameter a_1 is time-varying. Even though the transfer function is constantly changing, the system is always stable. The blue line in Figure 4.8 is the real value of the parameter throughout the simulation. The red and yellow lines are the estimated coefficient values when using a variable forgetting factor and a fixed one (equal to 0.9995), respectively. Figure 4.8 also displays the variation of β according to equation (3.22). As expected, when the variable forgetting factor is applied, the estimate has a small drift. Notwithstanding, it follows the real value whereas using a fixed forgetting factor takes too long to make the estimate change. The downside of this approach is that the estimated parameter vector loses its sparsity feature. The variations of β lead also to small fluctuations in the estimated values of the supposedly zero valued parameters.

Again, estimators respond much more quickly to model variations when a variable forgetting factor is used. The response can be faster or slower, depending on the parameter used in equation (3.22), although with faster response (and more alertness to variations) comes a greater steady state error.

Figure 4.9 shows the variance of the estimate as function of the rising time, in terms of the number of iterations it takes for the algorithm to react to a change. For a fixed forgetting factor, as its value

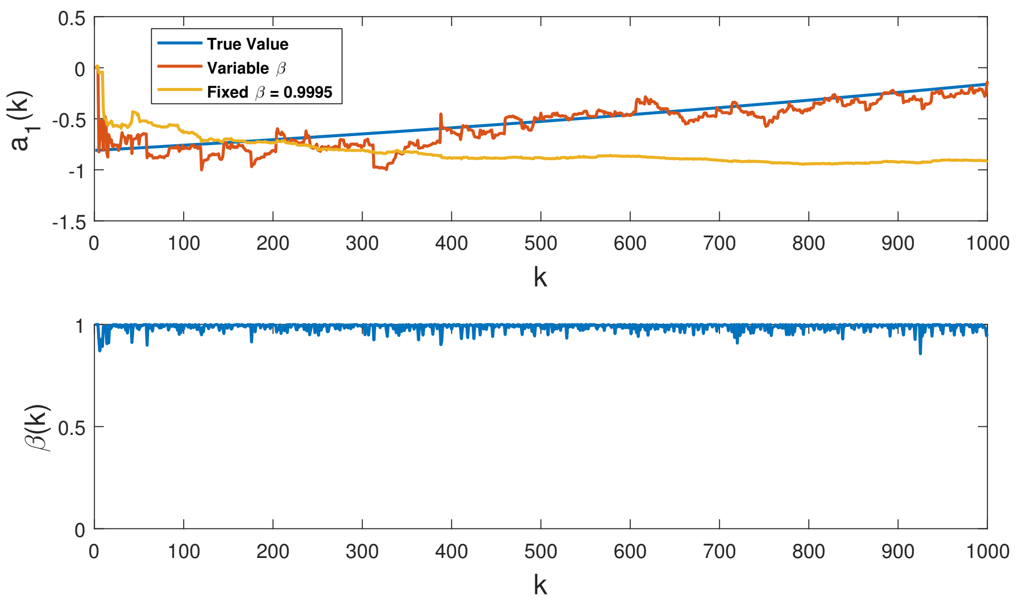

Figure 4.8: Time evolution of the system parameter a_1 during Simulation III (upper plot) and time evolution of the variable forgetting factor, using the sparsity-aware algorithm.

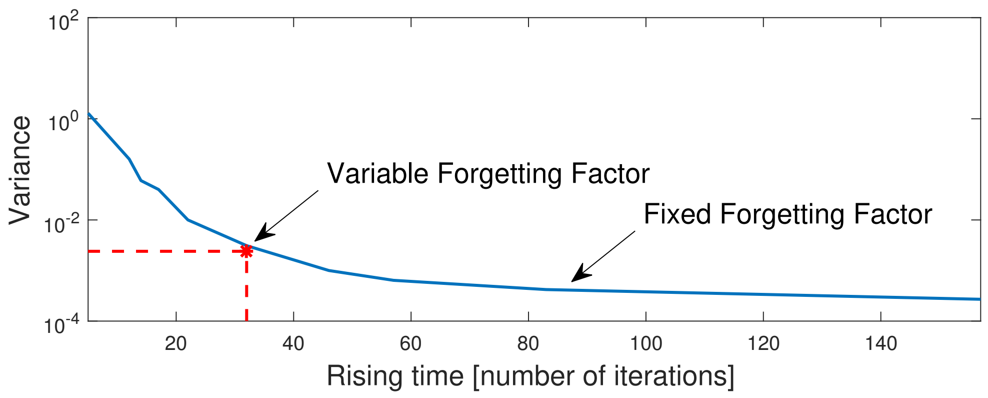

Figure 4.9: Variance as a function of the rising time.

becomes closer to one, the rising time increases and the variance decreases. As depicted in Figure 4.9, a variable forgetting factor can be more useful to the estimation as it exhibits a compromise between the rising time and the variance.

Hybrid systems concern dynamical systems governed by differential or difference equations that exhibit both continuous and discontinuous behavior [56]. In many cases, the discontinuous behavior comes from switches or interactions between distinct physical systems or parts of them.

Consider the following hybrid system identification example corresponding to the RC series circuit depicted in Fig. 4.10.

The system has one switch that allows shifting between one of two distinct states. The switch is either open and the output, $\mathrm{V_{out}}$, is measured at point A, or the switch is closed and $\mathrm{V_{out}}$ is measured at point B. The two states correspond to a first and a second order system, respectively.

The aim is to identify at each time instant which of the operating states is active, and it is expected that, using a sparse representation, the time instants of these transitions become evident. The knowledge of the first order system having less coefficients than the second order system aids

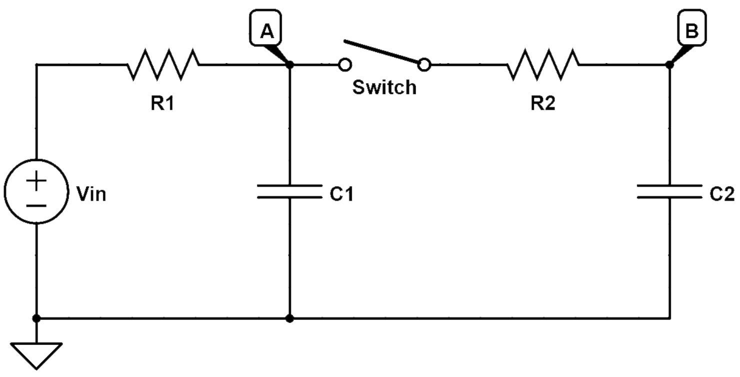

Figure 4.10: Second order RC series circuit.

the identification of the active state of the hybrid system at each time instant.

Let the measurements of $\mathrm{V_{in}}$ and $\mathrm{V_{out}}$ consist of samples at time instants $k = nT,\ n = 1, 2, ...,$ where T is the sampling time. The parameter estimation is updated for each time k. The regression vector $\varphi(t)$ is of the form $[\mathrm{V_{out}}(k-1)\ \mathrm{V_{out}}(k-2)\ ...\ \mathrm{V_{in}}(k)\ \mathrm{V_{in}}(k-1)\ \mathrm{V_{in}}(k-2)\ ...]$.

The data was acquired for values of $\mathrm{R_1} = 100\,\mathrm{k\Omega}$, $\mathrm{R_2} = 500\,\mathrm{k\Omega}$, $\mathrm{C_1} = 100\,\mathrm{mF}$, and $\mathrm{C_2} = 5\,\mu\mathrm{F}$, with a sampling rate lesser than the time constant of the system.

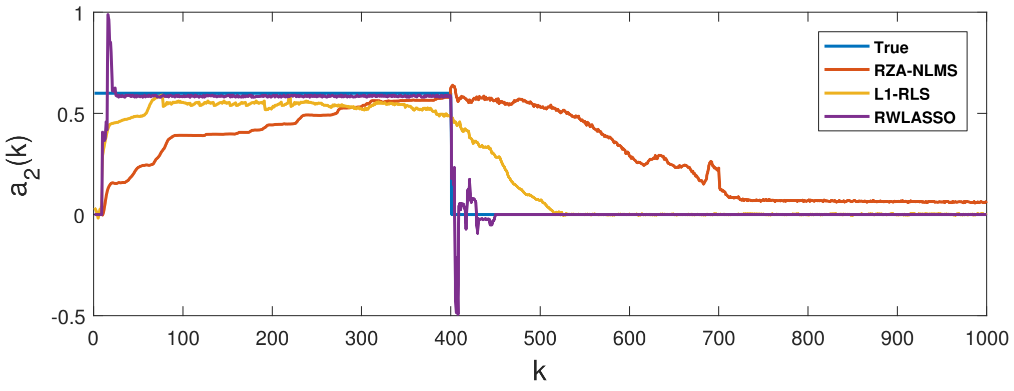

Figure 4.11: Time evolution of the system parameter a_2.

Figure 4.11 shows the parameter a_2 and the algorithms estimates $\theta_2(t)$ of a_2, that is the parameter associated to $\mathrm{V_{out}}(k-2)$. The true value of this parameter, represented by the blue line, is zero in state A and 0.6 in state B.

Analyzing Figure 4.11, the initial system corresponds to state B and around $k = 400$ a sudden change is registered, where a drop in the absolute value of a_2, and thus the operating system no longer corresponds to the second order series RC circuit, as the order of the system decreases. This decrement of order is reflected in the number of nonzero coefficients of the regression vector. Thereby, as soon as the switch opens, the dependency on the measurement of $\mathrm{V_{out}}(k-2)$ vanishes, and thus a_2 becomes zero.

In respect to the algorithms, RW-LASSO is faster to react to the change and it has the most accurate estimate. It also has lower variance comparing to the other algorithms. Besides being the slowest, the RZA-NLMS algorithm does not even converge to the exact value of zero after the sudden parameter variation.

4.4.3 Case III: Using Sparse Estimators to avoid instability

The aim of the last simulation is to demonstrate that sparse estimation can be used not only to reveal the true order of the system. In fact, whether the order is known or not, this example illustrates that sparse estimation may be used to estimate systems parameters that by standard means are not possible to estimate.

The system with the following transfer function

$$H(q^{-1}) = \frac{-1.4 + 1.3q^{-1} - 0.6q^{-2}}{1 - 0.7q^{-1} + 068q^{-3}}, \tag{4.31}$$

is unstable. Besides a pair of complex conjugate zeros at $z_1, z_2 = 0.4643 \pm 0.4615j$, the system has poles located at $p_1, p_2 = 0.6988 \pm 0.6975j$ and $p_3 = -0.6975$. The pair of complex conjugate poles have an absolute value equal to 0.9873, and they are therefore inside the unit circle. The Z plane with the location of zeros and poles of the system can be observed in Figure 4.12 a). In Figure 4.12 b) is depicted the impulse response of the system.

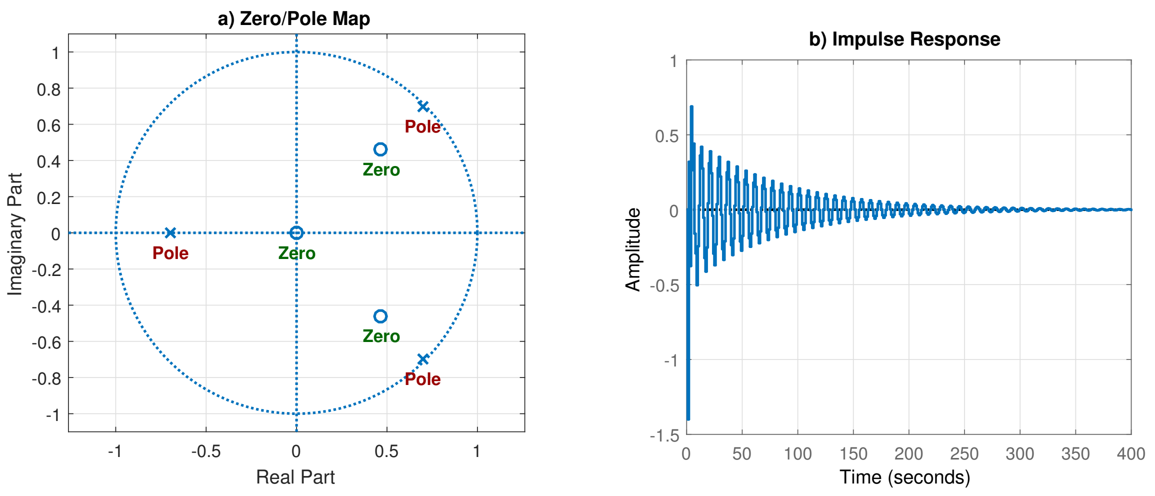

Figure 4.12: System from equation (4.31) a) impulse response , and b) Z plane with the pole/zero locations.

Be aware of equation (4.31), the coefficient a_2 is equal to zero. The estimated $\hat{a}_2$, using RLS, is expected to be different from zero. As some of the poles are located near the unit circle boundary, slightly different estimates can shift the poles to outside the unit circle. Consequently, the system becomes unstable, the output grows without bound, and sooner or later the estimation procedure is compromised.

Using sparsity-aware estimators, it is possible to reduce the value of $\hat{a}_2$ towards zero, thereby having no effect on the system stability. Figure 4.13 has the map of poles and zeros of the systems resulting from a) RLS estimation and b) sparse-based estimation. The algorithms were simulated also considering a model of eighth order. That is the reason for the high number of poles and zeros. The sparsity-aware estimator reduced the irrelevant coefficients to zero, hence the number of poles and zeros in the figure match the number of poles and zeros of the system from equation (4.31). Observe the two poles that were slightly moved over the boundary of the unit circle in the first case.

Figure 4.14 has the impulse responses of the systems resulting from a) RLS estimation and b) sparse-based estimation. In the first case, the response grows unlimited while the response displayed in the second case is similar to that of the system given by equation (4.31).

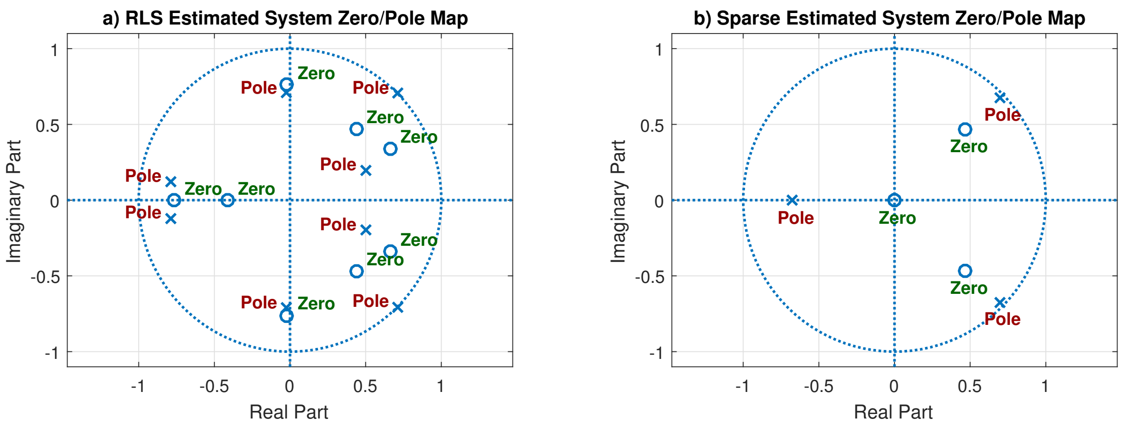

Figure 4.13: Z plane with the pole/zero locations of a) the system from equation (4.31), and b) the system resulting from sparse estimation.

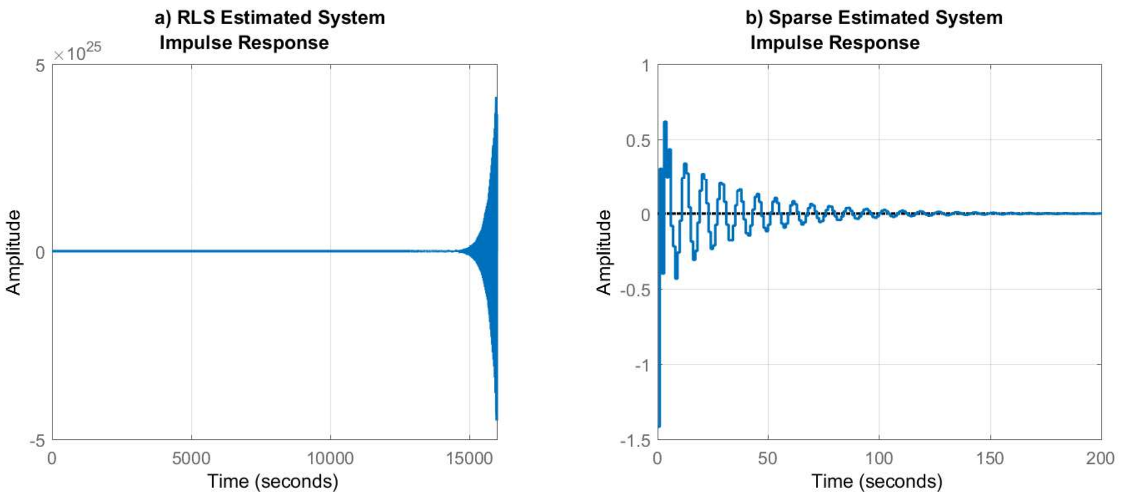

Figure 4.14: Impulse responses of a) the system from equation (4.31), and b) the system resulting from sparse estimation.

Even though this example might seem of no relevance, since unstable systems might be controlled with no problems, in the following chapter a similar example is illustrated, where for a given system, the controller only works if a sparse estimator is used.

Besides ARX models, ARMAX models were also used to test the algorithms in a setup similar to the Extended-RLS, however the estimation results were not good. A possible explanation for this is related to the estimation of the c_i parameters. These parameters are estimated using regression vectors that contain samples of the prediction errors. The effect of zero attractor terms and shrinkage induced by regularization also affects the c_i parameters, for instance, if these parameters are shrunk to zero then the model reduces to ARX and the remaining parameters estimates will be polarized.

5 Adaptive control with sparse estimation

Contents

MPC comprises a wide range of control methods designed around common ideas. These ideas involve the use of a model to predict the process output along future time instants, the minimization of a cost function in order to calculate a control sequence, and a receding strategy stating that only the first element of the calculated control sequence is applied to the process at each time instant.

The MPC advantages over other control methods are related to the ability to control a large number of process types, including non-minimum phase, unstable, or those with significant dead-time. Furthermore, the resulting control laws are easy to implement and embed in a natural way constraints. One of the main advantages of predictive control is the system reaction before an effective change happens if the future evolution of the reference signal is known. In fact, many applications have *a priori* knowledge of the reference signal which can be used to neglect the effects of the system dead time. The main drawbacks are the the number of parameters to choose and the need for an appropriate model, following a high dependency on the model prediction.

The expression for the general cost function is

$$J(N_1, N_2, N_u) = E\left\{\sum_{j=N_1}^{N_2} \delta(j)\left[y(k+j|k) - y_{ref}(k+j)\right]^2 + \sum_{j=1}^{N_u} \rho(j)\left[\Delta u(k+j-1)\right]^2\right\}, \tag{5.1}$$

where $E\{\cdot\}$ is the expectation operator, $y(k+j|k)$ is the j-step ahead prediction of the system output, and $y_{ref}(k+j)$ is a future reference.

A few methods do not consider the second term (control effort), as is the case of the MVC presented in Section 5.1, while other methods consider values of the control signal ($u(k)$) instead of its increments ($\Delta u(k)$) such as in the detuned MVC. The weighting sequences $\delta(j)$, $\rho(j)$ can also assume different values based on the approach used. The GPC, detailed in Section 5.2, regards the exact equation (5.1) as the cost function (with $\delta(j) = 1$ and $\rho(j)$ constant for $j = 1, \ldots, N_u$). This large set of available choices lead to a larger number of different methods.

The parameters N_1 and N_2 are the minimum and maximum output horizons. These values correspond to the limits of the time instants for which the output y needs to follow the reference y_{ref}. The first output to be affected by the input $u(k)$ is $y(k + d_0)$, so that if the dead time d_0 is known then there is no reason for $N_1 < d_0$. If the dead time is not known, $N_1 = 1$ is a proper choice. In this case, the degree of $B(q^{-1})$ should be increased as to include all possible values of d_0, resulting in a controller more robust to changes in the dead time. N_2 should be larger than the degree of $B(q^{-1})$ so that the maximum output horizon is longer than a possible negative-going non minimum phase transient.

The parameter N_u is the control horizon and its value depends on the type of process. For simpler processes $N_u = 1$ may be enough to achieve great results. For more complex processes such as unstable systems, N_u should be, at least, equal to the number of unstable poles.

The parameter ρ can be tuned to cover a large scope of processes and its value must be chosen according to the the particular process. For example, for an unstable open loop process, a large value of ρ will lead to an unstable closed loop process.

Fig. 5.1 shows how MPC is applied. For simplicity $N_2 = N_u = N$ is assumed. The objective is to make the process output (yellow signal) follow the reference (blue signal). Applying the first element of the predicted input signal (orange signal), and introducing the analyzed output in the feedback provides the controller with robustness to errors. At each time instant k, the process input is calculated based on a prediction horizon N. According to a receding horizon strategy, this operation can be performed at each time instant.

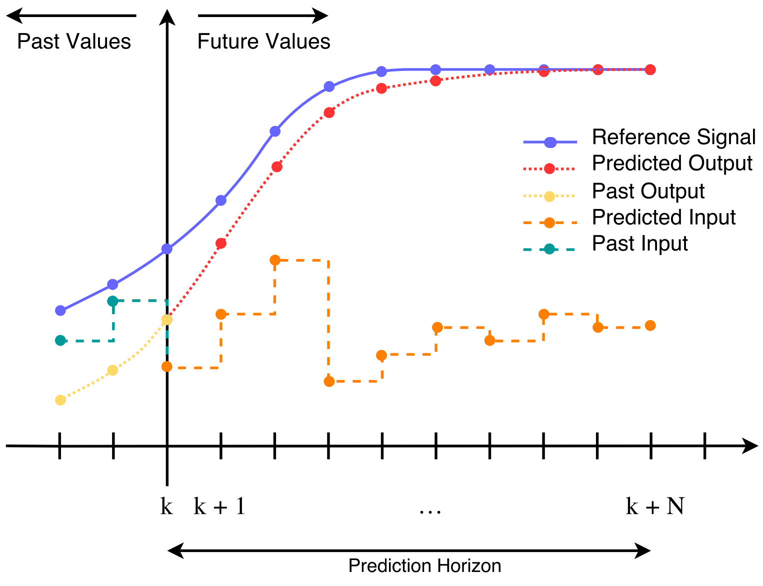

Figure 5.1: MPC application scheme (adapted from [57]).

5.1 Minimum Variance Control

In this section the design of Minimum Variance Control is described. Recalling equation (5.1), MVC adopts a similar cost function, apart from $\rho(j) = \rho = 0$.

The control target includes system noise, observation noise, and other unmodeled dynamics that cannot be represented in mathematical expressions. For that reason, the Controlled AutoRegressive Moving Average (CARMA) model represents the process dynamics

$$A(q^{-1})y(k) = B(q^{-1})u(k - d_0) + C(q^{-1})e(k), \tag{5.2}$$

where polynomials $A(q^{-1})$, $B(q^{-1})$, and $C(q^{-1})$ are defined as in equation (3.1), and d_0 is the dead time of the system. To implement the controller some assumptions are to be made: the process is required to have a single-input single-output, to have a minimum phase property ($B(q^{-1})$ has all zeros inside the unit circle). Furthermore, no common factors can appear in $(A(q^{-1}), B(q^{-1}))$, all the zeros of $C(q^{-1})$) have to be inside the unit circle, and $\{e(k)\}$ is a sequence of independent random variables (white noise) with zero mean and variance σ_e^2.

If both sides of equation (5.2) are multiplied by $F(q^{-1})$, a monic polynomial of degree $d_0 - 1$, such that

$$F(q^{-1}) = 1 + f_1 q^{-1} + \ldots + f_{d_0-1} q^{-d_0+1}, \tag{5.3}$$

then, according to the following Diophantine equation

$$C(q^{-1}) = F(q^{-1})A(q^{-1}) + q^{-d_0}G(q^{-1}), \tag{5.4}$$

where $G(q^{-1})$ is given by

$$G(q^{-1}) = 1 + g_1 q^{-1} + \ldots + g_{n_a-1} q^{-n_a+1}, \tag{5.5}$$

equation (5.2) becomes

$$\begin{aligned} A(q^{-1})F(q^{-1})y(k+d_0) &= B(q^{-1})F(q^{-1})u(k) + C(q^{-1})F(q^{-1})e(k+d_0) \Leftrightarrow \\ \Leftrightarrow \left\{C(q^{-1}) - q^{-d_0}G(q^{-1})\right\} y(k+d_0) &= B(q^{-1})F(q^{-1})u(k) + C(q^{-1})F(q^{-1})e(k+d_0). \end{aligned} \tag{5.6}$$

The polynomials $F(q^{-1})$ and $G(q^{-1})$ can be determined by polynomial division. An explicit formula for the coefficients of these polynomials is detailed in Appendix A. Furthermore equation (5.6) can be rearranged as follows

$$y(k+d_0) = \hat{y}(k+d_0|k) + \tilde{y}(k+d_0|k), \tag{5.7}$$

where

$$\hat{y}(k+d_0|k) = \frac{G(q^{-1})y(k) + B(q^{-1})F(q^{-1})u(k)}{C(q^{-1})}, \tag{5.8}$$

is the d_0-step ahead predictor based on data up to time k, and

$$\tilde{y}(k+d_0|k) = F(q^{-1})e(k+d_0), \tag{5.9}$$

is the d_0-step prediction error.

The control law is determined in such a way that the cost function

$$\begin{aligned} J &= E\left\{[y(k+d_0) - y_{ref}(k+d_0)]^2\right\} \\ &= E\left\{[\hat{y}(k+d_0|k) + \tilde{y}(k+d_0|k) - y_{ref}(k+d_0)]^2\right\} \\ &= E\left\{\hat{y}^2(k+d_0|k)\right\} + E\left\{\tilde{y}^2(k+d_0|k)\right\} + 2E\left\{\hat{y}(k+d_0|k) \times F(q^{-1})e(k+d_0)\right\} + \\ &+ E\left\{y_{ref}^2(k+d_0)\right\} - 2E\left\{\hat{y}(k+d_0|k)y_{ref}(k+d_0)\right\} - 2E\left\{y_{ref}(k+d_0) \times F(q^{-1})e(k+d_0)\right\}, \end{aligned} \tag{5.10}$$

is as small as possible. The assumption that $e(k)$ is an i.i.d. sequence is of real importance as it makes the cross terms of $F(q^{-1})e(k+d_0)$ vanish

$$\begin{aligned} J &= E\left\{\hat{y}^2(k+d_0|k)\right\} + E\left\{\tilde{y}^2(k+d_0|k)\right\} + E\left\{y_{ref}^2(k+d_0)\right\} - 2E\left\{\hat{y}(k+d_0|k)y_{ref}(k+d_0)\right\} \\ &= E\left\{\hat{y}^2(k+d_0|k)\right\} + \sigma_e^2(1 + f_1^2 + \ldots + f_{d_0-1}^2) + E\left\{y_{ref}^2(k+d_0)\right\} - 2E\left\{\hat{y}(k+d_0|k)y_{ref}(k+d_0)\right\}. \end{aligned} \tag{5.11}$$

Since the second and third terms are independent of $u(k)$, minimizing J amounts to choose the input $u(k)$ so that

$$\frac{\partial}{\partial u}\left(\hat{y}^2(k+d_0|k)\right) - 2\frac{\partial}{\partial u}\left(\hat{y}(k+d_0|k)y_{ref}(k+d_0)\right) = 0. \tag{5.12}$$

Finally, the MVC law is

$$u(k) = \frac{C(q^{-1})y_{ref}(k+d_0) - G(q^{-1})y(k)}{B(q^{-1})F(q^{-1})}. \tag{5.13}$$

Recall that if no reference signal y_{ref} is considered, the MVC law is

$$u(k) = -\frac{G(q^{-1})}{B(q^{-1})F(q^{-1})}y(k). \tag{5.14}$$

This controller cancels the zeros of the plant transfer function. As $C(q^{-1})$ is assumed to be stable, the requirement for stability is that $B(q^{-1})$ cannot have zeros outside the unit circle. The closed loop characteristic equation is $q^{d_0-1}B(q^{-1})C(q^{-1}) = 0$, which implies $d_0 - 1$ poles at the origin, n_a poles at the zeros of $C(q^{-1})$, and $n_a - d_0$ poles at the zeros of $B(q^{-1})$. If $B(q^{-1})$ is minimum phase, then these poles are inside the unit circle. Hence the importance of the assumption made earlier.

If $B(q^{-1})$ is not stable, then $y(k)$ will still be bounded and have minimum variance, but $u(k)$ will generally be unstable, growing with no bound.

To deal with non minimum phase systems, the cost in equation (5.10) can be modified to weigh the loss due to the control action

$$J = E\left\{[y(k+d_0) - y_{ref}(k+d_0)]^2 + \rho u^2(k)\right\}, \tag{5.15}$$

so that ρ is positive, and balances the importance assigned to the control variable. However, this setup, known as detuned MVC, does not solve the problem of instability when the system is non minimum phase and open loop unstable.

5.2 Generalized Predictive Control

GPC is one of the simplest and most popular MPC techniques. Proposed by Clarke *et al.* [58], the main idea of GPC is to calculate a sequence of future control signals such that it minimizes a multi step quadratic cost defined over a prediction horizon as in equation (5.1). It incorporates a control horizon with weighted control increments, and it is capable of controlling an ample variety of processes, including non minimum phase.

The processes are described by a CARIMA model

$$A(q^{-1})y(k) = B(q^{-1})u(k-d_0) + C(q^{-1})\frac{e(k)}{\Delta}, \tag{5.16}$$

where $\Delta = 1 - q^{-1}$, d_0 is the system dead time, and $A(q^{-1})$, $B(q^{-1})$, and $C(q^{-1})$ are polynomials in the backward shift operator q^{-1}. Again and for simplicity, $C(q^{-1})$ is considered to be equal to 1.The controller automatically has an integrator, necessary to compensate for the drifting noise term.

The objective of the controller is to compute the future control sequence such that the future output is driven close to the reference. Defining defining the horizons N_1 and $N_2 = N_u = N$, a set of predicted outputs $\hat{y}(k+j|k)$ for $N_1 \leq j \leq N$ are computed from past values of inputs and outputs, as well as future control signals. These predictors are then, according to the model, used in the cost function to obtain an expression whose minimization leads to the desired control sequence.

Consider the following Diophantine equation

$$1 = F_j(q^{-1})\Delta A(q^{-1}) + q^{-j}G_j(q^{-1}), \tag{5.17}$$

where $F_j(q^{-1})$ and $G_j(q^{-1})$ of degrees $j-1$ and n_a, respectively, are uniquely defined polynomials. The subscript j denotes the polynomial associated with the prediction of the output at time $k+j$. An explicit formula for computing the coefficients of these polynomials is detailed in Appendix A.

The output at j steps ahead is

$$y(k+j) = F_j(q^{-1})B(q^{-1})\Delta u(k+j-d_0) + G_j(q^{-1})y(k) + F_j(q^{-1})e(k+j), \tag{5.18}$$

and the predictor at j steps ahead, orthogonal to the output, is

$$\hat{y}(k+j|k) = F_j(q^{-1})B(q^{-1})\Delta u(k+j-d_0) + G_j(q^{-1})y(k). \tag{5.19}$$

Furthermore, upon defining the following equality

$$F_j(q^{-1})B(q^{-1}) = W_j(q^{-1}) + q^{-(j-d_0+1)}\bar{W}_j(q^{-1}), \tag{5.20}$$

the predictor can be equivalently written as

$$\hat{y}(k+j|k) = \underbrace{W_j(q^{-1})\Delta u(k+j-d_0)}_{\text{free response}} + \underbrace{\bar{W}_j(q^{-1})\Delta u(t-1) + G_j(q^{-1})y(k)}_{\text{forced response}}. \tag{5.21}$$

The first term is called free response, as it represents the prediction of the output $y(k+j)$ when no future control action is applied to the system. The second term is the forced response, since it relates to the output prediction taking the future control actions $u(k+j-d_0)$, for $j > 1$.

Let the forced response be expressed as $\bar{y}_j$, and W be a matrix formed with the terms w_i^j of polynomials $W_j(q^{-1})$ (in fact $w_i^j = w_i$). Accordingly,

$$\mathbf{y} = \begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix}, \ \Delta\mathbf{u} = \begin{bmatrix} \Delta u(k) \\ \vdots \\ \Delta u(k+N-d_0) \end{bmatrix}, \ \mathbf{e} = \begin{bmatrix} F_1(q^{-1})e(k+1) \\ \vdots \\ F_N(q^{-1})e(k+N) \end{bmatrix},$$
$$\mathbf{y}_{ref} = \begin{bmatrix} y_{ref}(k+1) \\ \vdots \\ y_{ref}(k+N) \end{bmatrix}, \ \mathbf{W} = \begin{bmatrix} w_1 & 0 & 0 & \dots & 0 \\ w_2 & w_1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_N & w_{N-1} & w_{N-2} & \dots & w_1 \end{bmatrix}, \ \bar{\mathbf{y}} = \begin{bmatrix} \bar{y}_1(k) \\ \vdots \\ \bar{y}_N(k) \end{bmatrix}. \tag{5.22}$$

The cost function can be recast as

$$\begin{aligned} J(1,N,N) &= E\left\{(\mathbf{y}-\mathbf{y}_{ref})^T(\mathbf{y}-\mathbf{y}_{ref}) + \rho\Delta\mathbf{u}^T\Delta\mathbf{u}\right\} \\ &= (\mathbf{W}\Delta\mathbf{u} + \bar{\mathbf{y}} - \mathbf{y}_{ref})^T(\mathbf{W}\Delta\mathbf{u} + \bar{\mathbf{y}} - \mathbf{y}_{ref}) + \rho\Delta\mathbf{u}^T\mathbf{u}. \end{aligned} \tag{5.23}$$

The minimum of J is found by equating its gradient to zero, which leads to

$$\Delta\mathbf{u} = \left(\mathbf{W}^T\mathbf{W} + \rho\mathbf{I}\right)^{-1}\mathbf{W}^T(\mathbf{y}_{ref} - \bar{\mathbf{y}}). \tag{5.24}$$

If no reference signal $\mathbf{y}_{ref}$ is considered, the GPC law is just

$$\Delta\mathbf{u} = -\left(\mathbf{W}^T\mathbf{W} + \rho\mathbf{I}\right)^{-1}\mathbf{W}^T\bar{\mathbf{y}}. \tag{5.25}$$

Notice that this action involves inverting an $N \times N$ matrix, which for a large value of N requires a lot of computation time, prohibitive for a real time application. This shortcoming is addressed in [58] by reducing the control horizon to $N_u < N$ and assuming that the control increments are constant and equal to zero after N_u. In this case, matrix W becomes

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ w_2 & w_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & w_1 \\ \vdots & \vdots & \vdots & \vdots \\ w_N & w_{N-1} & \dots & w_{N-N_u+1} \end{bmatrix} \tag{5.26}$$

Thereby, the computations are reduced since the matrix being inverted is only of size $N_u \times N_u$.

Equation (5.24) yields the future control sequence for $k+j-d_0$ for $j=1,\ldots,N$, and according to the receding horizon strategy, only the first element of the sequence is applied to the process. As is, the control law amounts to

$$\Delta u(k) = \begin{bmatrix} 1 & 0 & \ldots & 0 \end{bmatrix} \left(\mathbf{W}^T\mathbf{W} + \rho\mathbf{I}\right)^{-1} \mathbf{W}^T \left(\mathbf{y}_{ref} - \bar{\mathbf{y}}\right). \tag{5.27}$$

Since $\Delta u(k) = u(k) - u(k-1)$ the control input is then

$$u(k) = u(k-1) + \Delta u(k). \tag{5.28}$$

In summary, at each time-instant the working principle of the adaptive controller consists of computing the steps described in Algorithm 4.

Algorithm 4 GPC method

```
1: set time instant $k \leftarrow 1$
2: obtain input and output measurements
3: estimate the parameters of the model from the measurements using algorithm 1, 2, 3, or similar
4: compute the controller parameters using the design control using equations (5.17)-(5.22)
5: obtain an optimal control sequence by minimizing the GPC cost function over a defined prediction
   horizon using equation (5.24)
6: apply the first element of the optimal control sequence using equations (5.27) and (5.28)
7: observe and analyze the controlled operation of the process
8: do $k \leftarrow k+1$ and go to 2: until the simulation is finished
```

Additionally, an alternative to the classic GPC setup is derived in Appendix C. That approach, as demonstrated in [59], has better results when the parameters of the system are constantly varying.

5.3 Target following as a control problem

The problem of target following can be formulated as a control problem. The target is described by an ARMA model.

$$y(k+d) = \frac{C(q^{-1})}{A(q^{-1})} e(k+d), \tag{5.29}$$

where $d \geq 1$. The aim is to estimate and predict, at each time instant, the observations of the target $y(k)$. Let the prediction of $y(k+d)$ be $\hat{y}(k+d|k)$, then the prediction error is

$$\tilde{y}(k+d|k) = y(k+d) - \hat{y}(k+d|k). \tag{5.30}$$

Define a cost function as to minimize the power of $\tilde{y}(k+1|k)$

$$J = E\left\{\tilde{y}^2(k+d|k)|\mathcal{O}^k\right\}. \tag{5.31}$$

Here, $\mathcal{O}^k$ denotes the observations up to time instant k, and $E\{\cdot\}$ is the expected value.

Let, by definition,

$$u(k) \triangleq \hat{y}(k+d|k), \tag{5.32}$$

be the control sequence. A control system can now be developed around this problem. Fig. 5.2 represents the block diagram used in this setup.

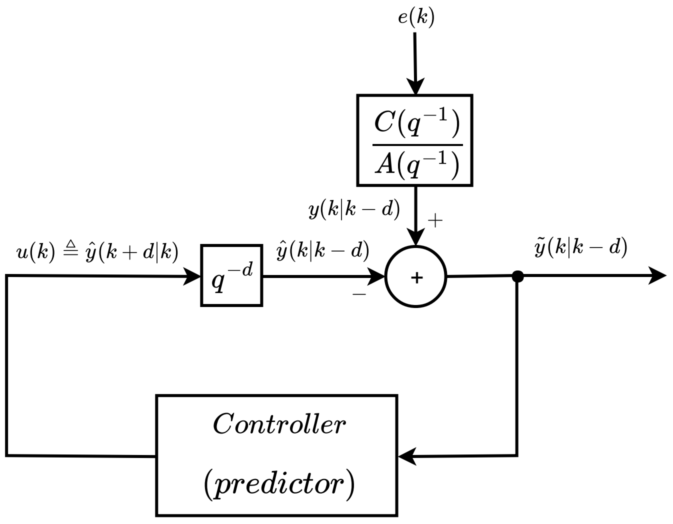

Figure 5.2: Block diagram of the target following problem.

If MVC is applied to the controller then, the Diophantine equation can be used to manipulate the polynomials

$$\begin{aligned} \frac{C(q^{-1})}{A(q^{-1})} = F_d(q^{-1}) + q^{-d}\frac{G_d(q^{-1})}{A(q^{-1})} &\Leftrightarrow C(q^{-1}) = F_d(q^{-1})A(q^{-1}) + q^{-d}G_d(q^{-1}) \\ &\Leftrightarrow F_d(q^{-1})A(q^{-1}) = C(q^{-1}) - q^{-d}G_d(q^{-1}) \\ &\Leftrightarrow G_d(q^{-1}) = q^{d}\left[C(q^{-1}) - F_d(q^{-1})A(q^{-1})\right]. \end{aligned} \tag{5.33}$$

Again the polynomial $F_d(q^{-1})$ of order $d-1$ is monic

$$F_m(q^{-1}) = 1 + f_1 q^{-1} + \ldots + f_{d-1}q^{-d+1}. \tag{5.34}$$

The difference between the observed value and the predicted value can be expanded

$$\begin{aligned} \tilde{y}(k+d) &= y(k+d) - \hat{y}(k+d) \\ &= \frac{C(q^{-1})}{A(q^{-1})}e(k+d) - \hat{y}(k+d|k) \\ &= F_d(q^{-1})e(k+d) + \frac{G_d(q^{-1})}{A(q^{-1})}e(k) - \hat{y}(k+d|k) \\ &= F_d(q^{-1})e(k+d) + \frac{G_d(q^{-1})}{A(q^{-1})}\frac{A(q^{-1})}{C(q^{-1})}y(k) - \hat{y}(k+d|k) \\ &= \underbrace{F_d(q^{-1})e(k+d) + \frac{G_d(q^{-1})}{C(q^{-1})}y(k)}_{y(k+d)} - \hat{y}(k+d|k). \end{aligned} \tag{5.35}$$

Now $y(k+d)$ is a sum of two distinct terms where the first one correspond to future values of $e(k)$, and the second one depends only on past and present observations. Applying the new expression for $\tilde{y}(k+d)$ to equation (5.31) and assuming the process $e(k)$ to be a sequence of uncorrelated samples of white noise, with zero mean and variance σ_e^2, gives

$$J = E\left\{\left[\frac{G(q^{-1})}{C(q^{-1})}y(k) - \hat{y}(k+d|k)\right]^2\right\} + E\left\{\left[F_d(q^{-1})e(k+d)\right]^2\right\}, \tag{5.36}$$

which means that the minimum value of J can be attained when

$$\hat{y}(k+d|k) = \frac{G(q^{-1})}{C(q^{-1})} y(k). \tag{5.37}$$

The minimum value of J is then

$$J_{min} = E\left\{\left[F_d(q^{-1})e(k+d)\right]^2\right\} = \left(1 + f_1^2 + \ldots + f_{d-1}^2\right)\sigma_e^2, \tag{5.38}$$

which is the variance of the prediction error, considering the optimal predictor.

5.4 Results

All the estimated models of this section admit the same order, equal to 5, even though the true systems to be estimated are of lower orders. The choice of fifth order is made since all the systems under simulation exhibit lower orders and, at the same, the number of plots regarding the parameters is not too large. Consequently the estimated vectors will always have 11 parameters, and part of them should be automatically estimated as zero. The models are then of the type

$$y(k) = \frac{b_0 + b_1 q^{-1} + b_2 q^{-2} + b_3 q^{-3} + b_4 q^{-4} + b_5 q^{-5}}{1 + a_1 q^{-1} + a_2 q^{-2} + a_3 q^{-3} + a_4 q^{-4} + a_5 q^{-5}} u(k - d_0) + e(k). \tag{5.39}$$

5.4.1 MVC results

To illustrate the functioning of the MVC based Self-Tuning controller, the following model is considered

$$y(k) = \frac{1 + 0.5q^{-1}}{1 - 1.099q^{-1} + 0.1778q^{-2}} u(k-1) + e(k). \tag{5.40}$$

For the first simulation, a fixed forgetting factor is used and the generated Gaussian white noise has variance equal to 0.01. The system was simulated during 1000 samples, the first 500 samples being open loop. After that the self-tuning adaptive controller was switched on. Figure 5.3 shows the output signal and the control signal, using the MVC law. Notice how, after the first 500 samples, the

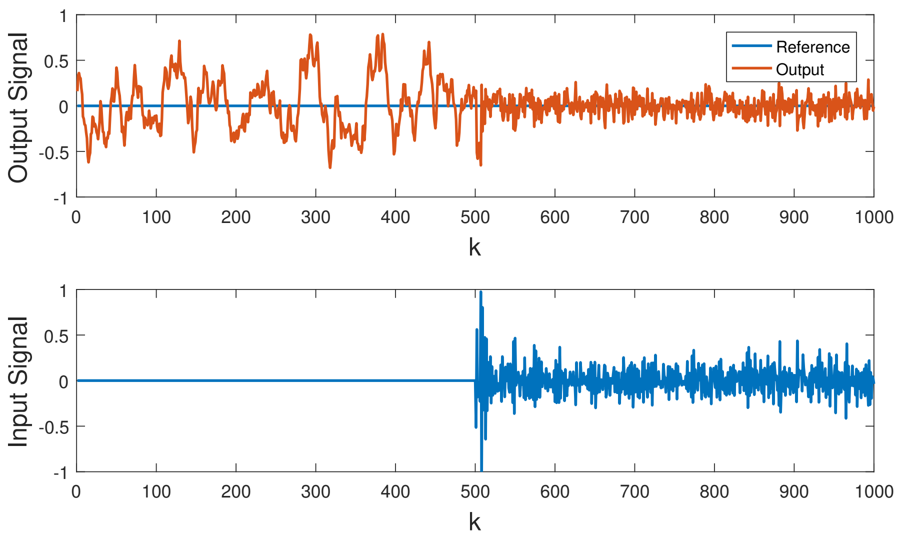

Figure 5.3: Plant output signal $y(k)$ and reference signal (upper plot), and control signal $u(k)$ (lower plot).

controller reduces the variance of the output. For $k < 500$, the estimated variance of the output is equal to 0.1054. The dead time of the system is $d_0 = 1$, and thus $F(q^{-1}) = 1$. Therefore the minimum variance achievable for the output signal y is $\sigma_y^2 = \sigma_e^2 = 0.01$. Calculating the variance over the last 500 output samples gives a value of $\sigma_y^2 = 0.0119$, which amounts to 11.3% of the value before the controller had been switched on.

All the estimated parameters of the model are displayed in Figure 5.4. Notice that only a_1, a_2, ,b_0, and b_1 should have a value different from zero. In open loop, with $u = 0$, the sparse estimator estimate all the parameters b_i as zero, and not the correct values. After switching on the controller their values start approaching the true ones.

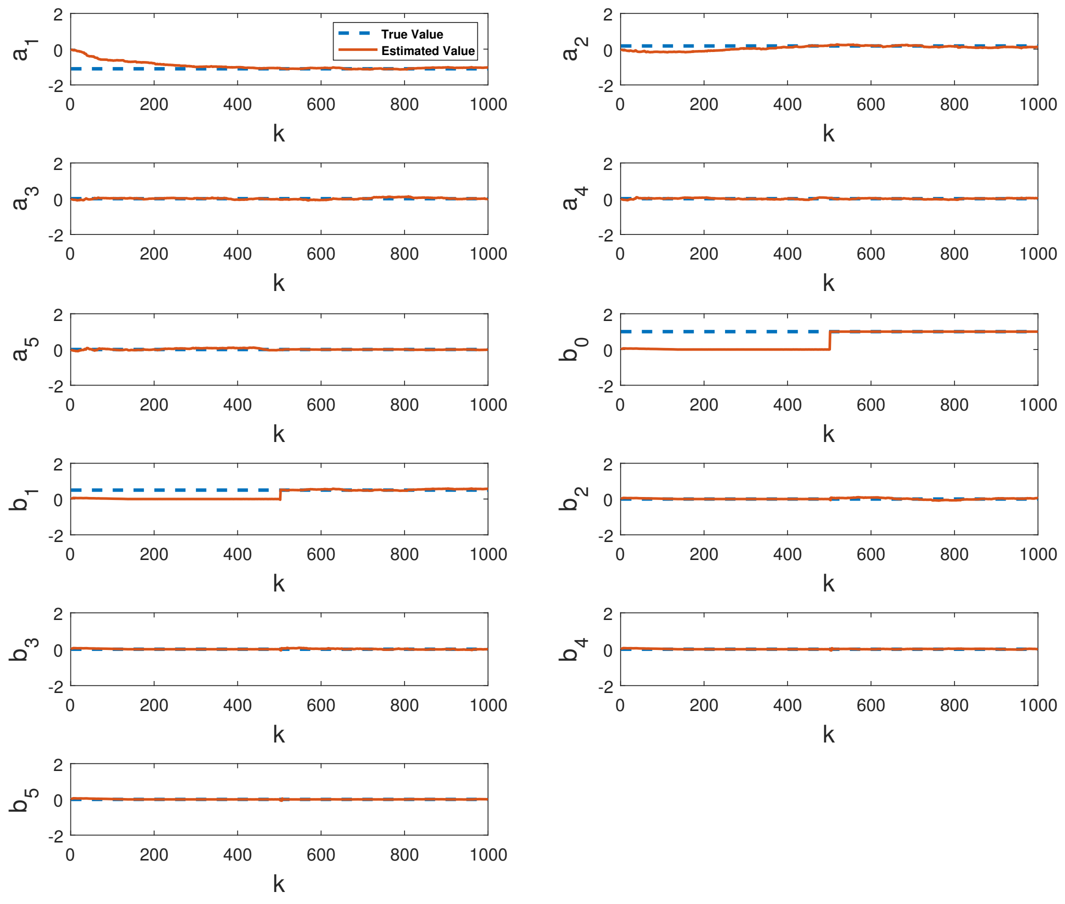

Figure 5.4: Estimated model parameters.

Consider now the following non minimum phase process

$$y(k) = \frac{0.25 + 0.3q^{-1}}{1 - 1.58q^{-1} + 0.67q^{-2}} u(k-1) + e(k). \tag{5.41}$$

This system has one zero at -1.2 (outside the unit circle) that originates an unstable mode. If MVC is used to control the system, a dangerous behavior is expected due to this unstable mode. Figure 5.5 depicts two different scenarios of the plant output and the control input for 100 samples, with Gaussian white noise with variance equal to 0.01.

The far left plots of Figure 5.5 confirm the statements made in section 5.1 about the input signal. Even though the output seems unaffected, the unstable mode makes the control signal grow without

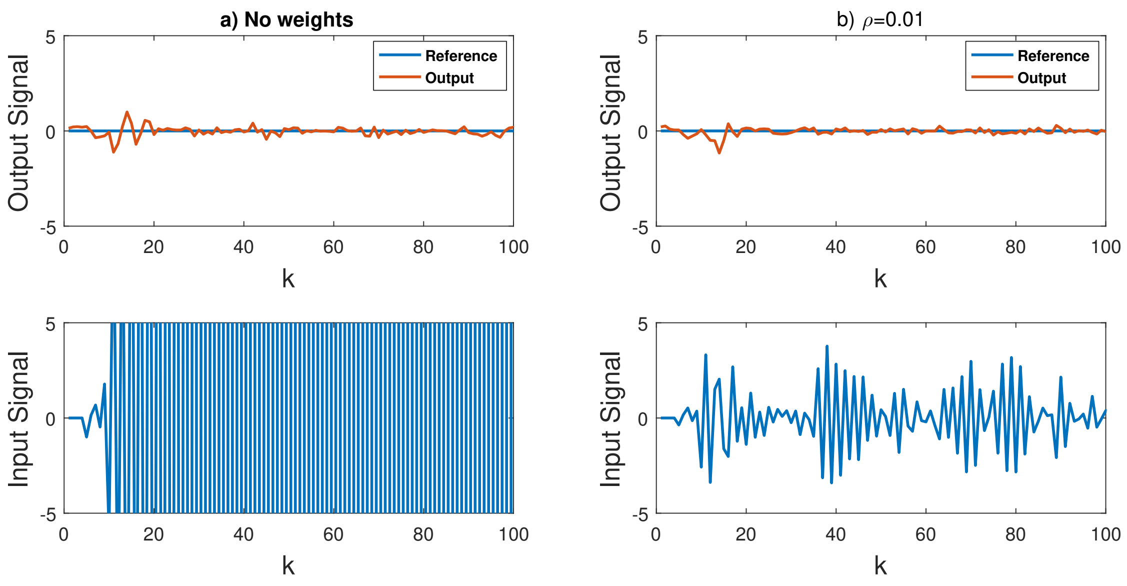

Figure 5.5: Plant output signal $y(k)$ and reference signal (upper plots), and control signal $u(k)$ (lower plots) when ρ is not used and when $\rho = 0.01$, a) and b) respectively.

bound. After a few samples the value of the manipulated variable is already extremely large, which in a real application it is either impossible to attain, or it may damage the equipment if no mechanisms such as saturation of the control signal are accounted for.

As stated in section 5.1, adding a weighting parameter ρ to the manipulated variable in the cost function avoids this shortcoming. The extra penalty introduced by the detuned MVC allows the manipulation of the closed-loop response obtained, where the value of ρ can be seen as a tuning knob. The far right plots of Figure 5.5 show, under the same conditions, a distinct output and input of the same plant. The addition of $\rho = 0.01$ to the control law leads to a bounded control signal and the system is controlled without further problems. Again, this control law is still unable to control open loop unstable and non minimum phase systems.

In certain cases, the use of a sparsity-aware estimator can be greatly advantageous. Consider the following process

$$y(k) = \frac{-0.65 + 0.64q^{-2}}{1 - 1.2q^{-1} + 0.5q^{-2}} u(k-1) + e(k). \tag{5.42}$$

Recovering now the MVC law version without ρ, a problem may arise when using it together with RLS to control this process. The polynomial $B(q^{-1})$ includes a parameter $b_1 = 0$, and as seen before, neither of the estimates of RLS will be equal to zero. The model resulting from the estimation procedure can therefore be non minimum phase, that is, it may lead to the problem of the unbound control signal. The $B(q^{-1})$ polynomial from equation (5.42) has zeros at $z_1, z_2 = \pm 0.9923$. A value of b_1 not too large can easily move one of this zeros outside the unit circle. If sparsity-aware algorithms are used instead, then the coefficient b_1 is reduced to zero, and thus the resulting model is minimum phase. Figure 5.6 a) depicts the the output and control signals considering RLS as estimator and Figure 5.6 b) depicts the output and control signals considering the sparsity-aware algorithm as estimator. The first case, although the output is following the reference trajectory, shows that the control variable grows in absolute value, which does not happen in the second case.

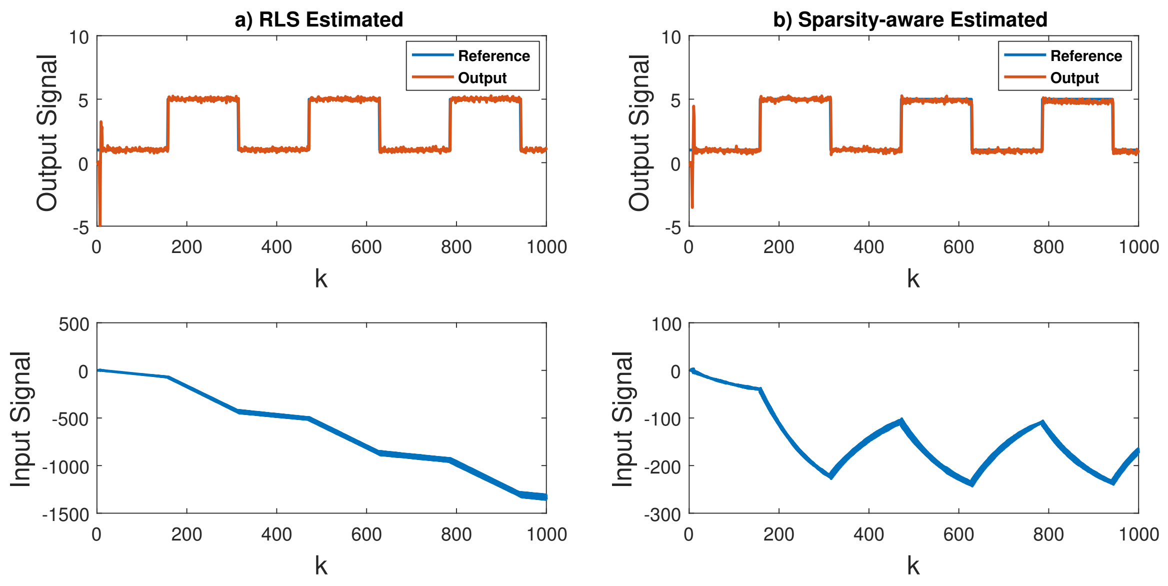

Figure 5.6: Plant output signal $y(k)$ and reference signal (upper plots), and control signal $u(k)$ (lower plots) using RLS and a sparsity-aware algorithm, a) and b) respectively.

5.4.2 GPC results

The self-tuning adaptive controller based on GPC is now analyzed considering the following model

$$y(k) = \frac{0.37}{1 - 0.5q^{-1} - 0.1q^{-2}} u(k-1) + \frac{e(k)}{\Delta}. \tag{5.43}$$

The system was first tested in noise absence conditions. Figure 5.7 shows the output signal and the control signal using a prediction horizon of 10 steps ahead ($N_2 = N_u = 10$).

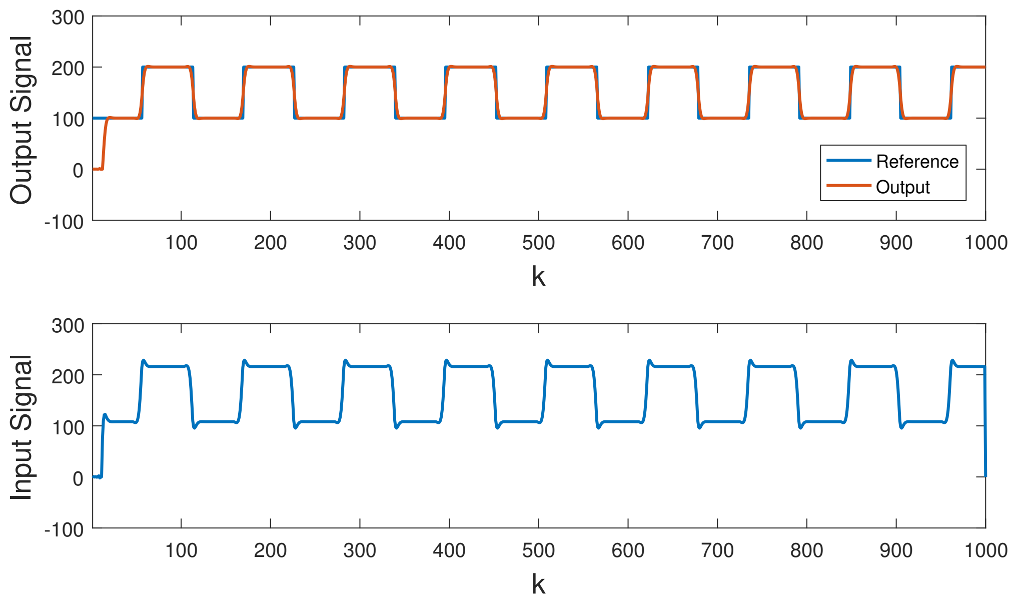

Figure 5.7: Plant output and reference in noise absence conditions, using a prediction horizon equal to 10.

To check how the prediction horizon affects the control law, the system was simulated a second time with a prediction horizon of 1 step ahead only.

This simulation is illustrated in Figure 5.8, where the output signal overshooting is visibly more accentuated, as the predictors do not take into account further reference values. Here, the controller acts similarly to the MVC based self-tuner since, at each time instant, the control law only considers the prediction of the output at one step ahead. Comparing the two cases, it is conclusive that a large prediction horizon allows for better tracking of reference trajectories. A small prediction horizon leads to a later reaction to reference changes causing the overshooting in the output.

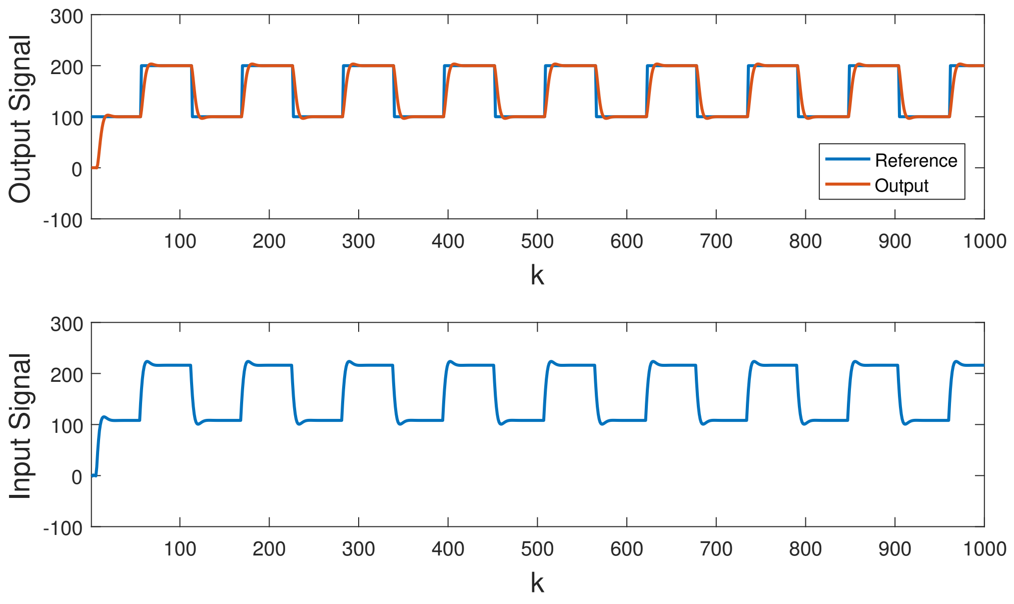

Figure 5.8: Plant output and reference, and control signal in noise absence conditions, using a prediction horizon equal to 1, as in MVC.

More relevant than the noiseless case is to study the functioning of the GPC based self-tuning adaptive controller under noisy conditions. In that regard, Figure 5.9 shows the plant output and input when Gaussian white noise with variance equal to 1 is added to the observations. The prediction horizon is set equal to 10 steps ($N_2 = N_u = 10$).

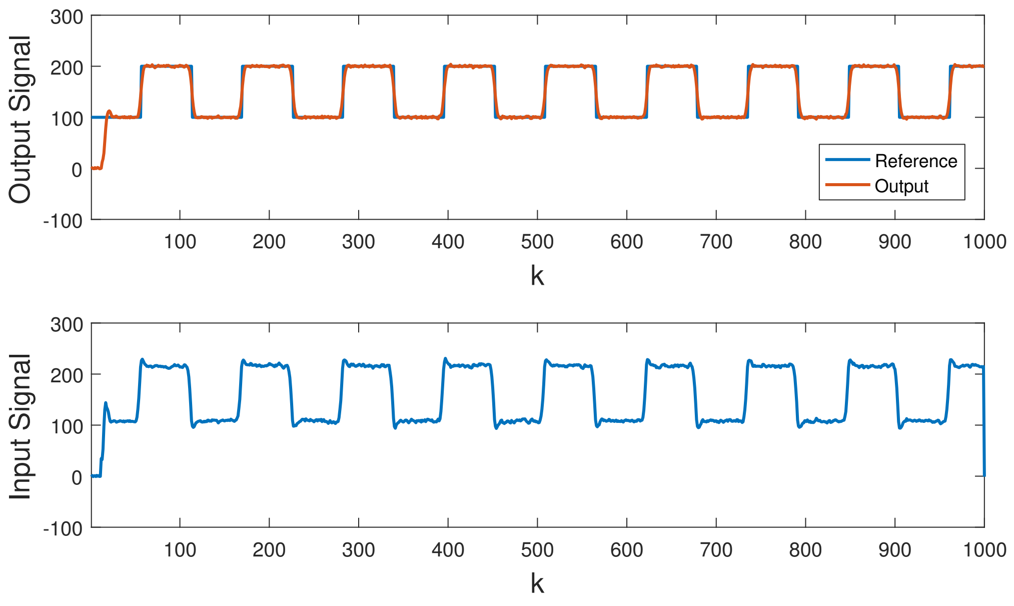

Figure 5.9: Plant output and reference affected by Gaussian white noise with variance equal to 1, using a prediction horizon equal to 10.

In addition, Figure 5.10 shows the plant output and input when not only Gaussian white noise with variance equal to 1 is added to the observations, but also a step disturbance of 10% affects the output. These random steps occurring at random times represent unmeasured disturbances, for instance, changes in the quality of the material. The CARIMA model was designed to account for these disturbances therefore the results do not differ much from one case to the other. These

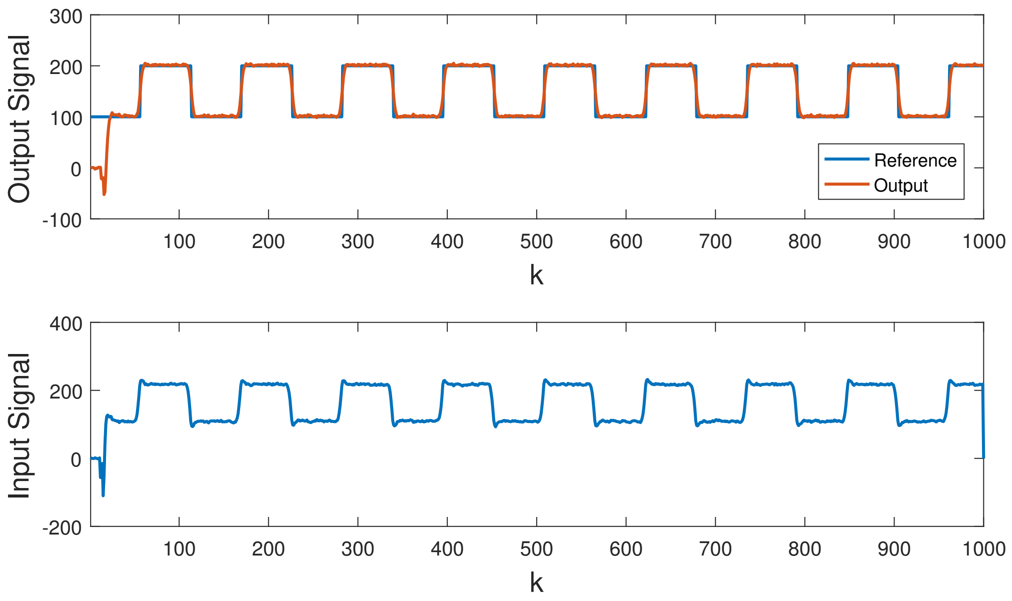

Figure 5.10: Plant output and reference affected by a step disturbance and Gaussian white noise with variance equal to 1, using a prediction horizon equal to 10.

simulations prove that, under significant variations of the plant output, the controller keeps its good performance, either in terms of robustness to disturbances and tracking.

To show the ability of GPC to control non minimum phase systems, the following model is considered

$$y(k) = \frac{-0.007803 - 0.01846q^{-1} - 0.01333q^{-2}}{1 - 2.729q^{-1} + 2.515q^{-2} - 0.7834q^{-3}} u(k-1) + e(k). \tag{5.44}$$

This model is used in [60] to simulate the operation of a small hydro power plant. The objective of the hydro power plant control is to track a reference. An advantage of MPC schemes is that if the future evolution of the reference is known beforehand, the system can react before the change has effectively taken place therefore avoiding the effects of the process dead time. In many applications (robotics for example) the future values of the reference are available at each time instant. In this case, the difference between having access to the reference signal up the present instant and having access also to the future reference trajectory is demonstrated in Figure 5.11 and Figure 5.12, respectively. In both simulations, $\rho = 0.98$ and the horizons are set to $N_1 = 1$, $N_2 = 7$, and $N_u = 3$.

If the controller is aware of the future reference trajectory, at a given time instant k, and a change happens in the prediction horizon $k + N_2$, the control signal applied at k already accounts for that change. This behavior can be observed in Figure 5.12, for example by the time in which the reference moves from 500 to -500, the output is already much lower than 500. The longer the prediction horizon, the sooner the controller actuates to compensate the change.

In contrast, Figure 5.11 reveals that if the reference is considered constant (and equal to the current set point reference) along the prediction interval, then the controller actuates to compensate

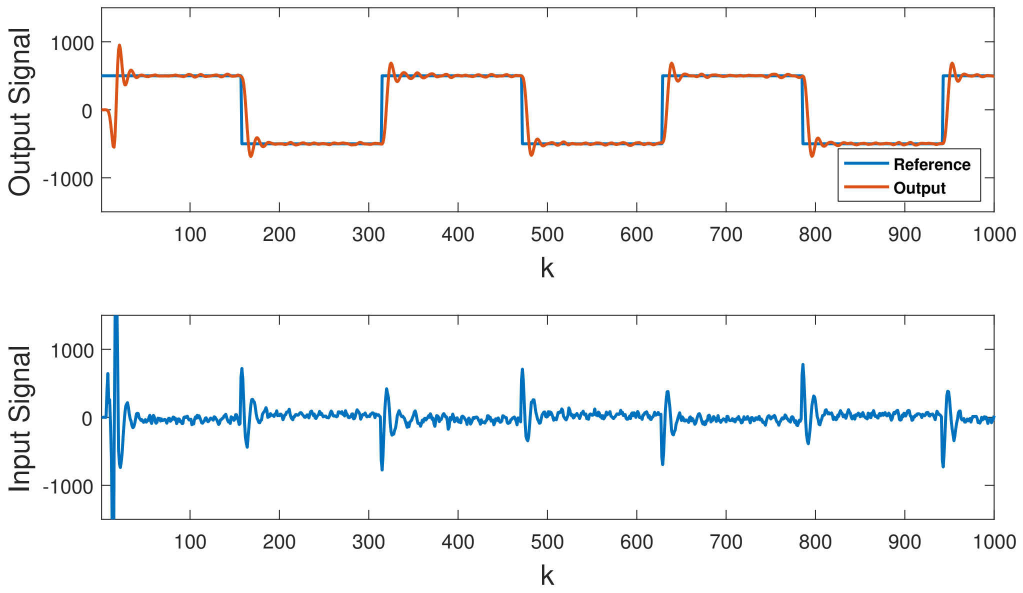

Figure 5.11: Plant output affected by Gaussian white noise with variance equal to 1, using $N_1 = 1$, $N_2 = 7$, $N_u = 3$, and assuming no knowledge of future reference values.

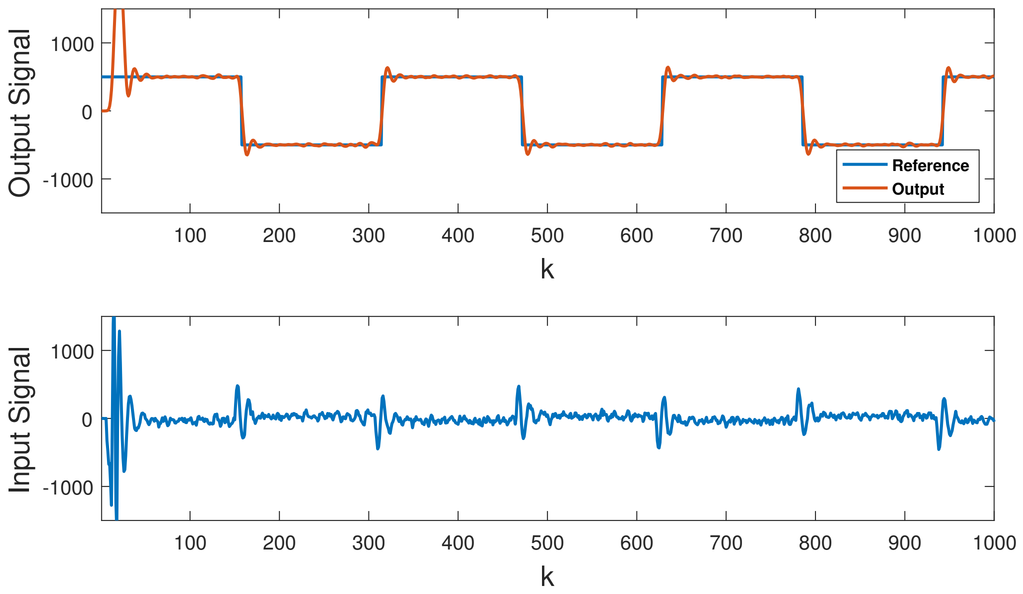

Figure 5.12: Plant output affected by Gaussian white noise with variance equal to 1, using $N_1 = 1$, $N_2 = 7$, $N_u = 3$, and knowing the future reference values.

the variation only when the change takes place.

A simulation is performed to assess the behavior of the controller when sudden changes happen. For that two different models are considered. The first one given by

$$y(k) = \frac{0.37 - 0.23q^{-1}}{1 - 0.95q^{-1} + 0.45q^{-2}} u(k-1) + e(k). \tag{5.45}$$

The first 200 samples of the simulation correspond to the control of the previous model. After that the $A(q^{-1})$ polynomial suffers a change such that the model becomes

$$y(k) = \frac{0.37 - 0.23q^{-1}}{1 + 0.8q^{-1} - 0.5q^{-2}} u(k-1) + e(k). \tag{5.46}$$

Figure 5.13 depicts the output and the input of the plant during the simulation. The output is affected by Gaussian white noise with variance equal to 0.1, $\rho = 0.98$, and the horizons are $N_1 = 1$, $N_2 = 17$, and $N_u = 7$. The change happens at $k = 200$ and the controller takes a couple of

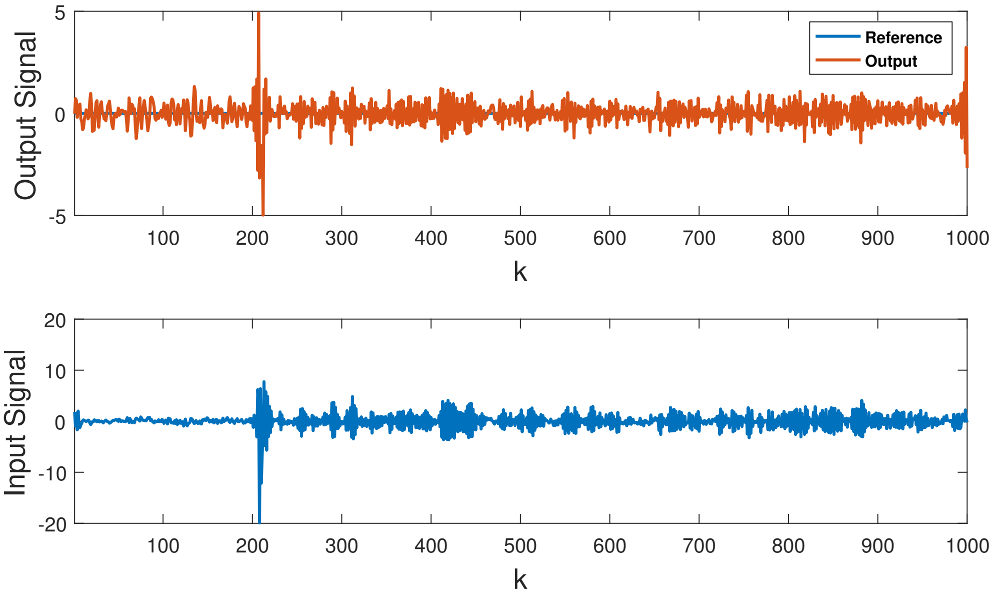

Figure 5.13: Plant output affected by Gaussian white noise with variance equal to 0.1, using $N_1 = 1$, $N_2 = 17$, $N_u = 7$.

samples to readjust the model parameters to the new values. The estimated parameters are shown in Figure 5.14. Even when the estimates do not exactly converge to the true values, the controller maintains good performance regarding variance reduction and reference tracking. After the change takes place, the convergence to the new values could be accelerated by setting a smaller forgetting factor. However, as seen before, the forgetting factor value also influences the sparsity enforced by the estimator. It is then necessary to find a good compromise between convergence and sparsity.

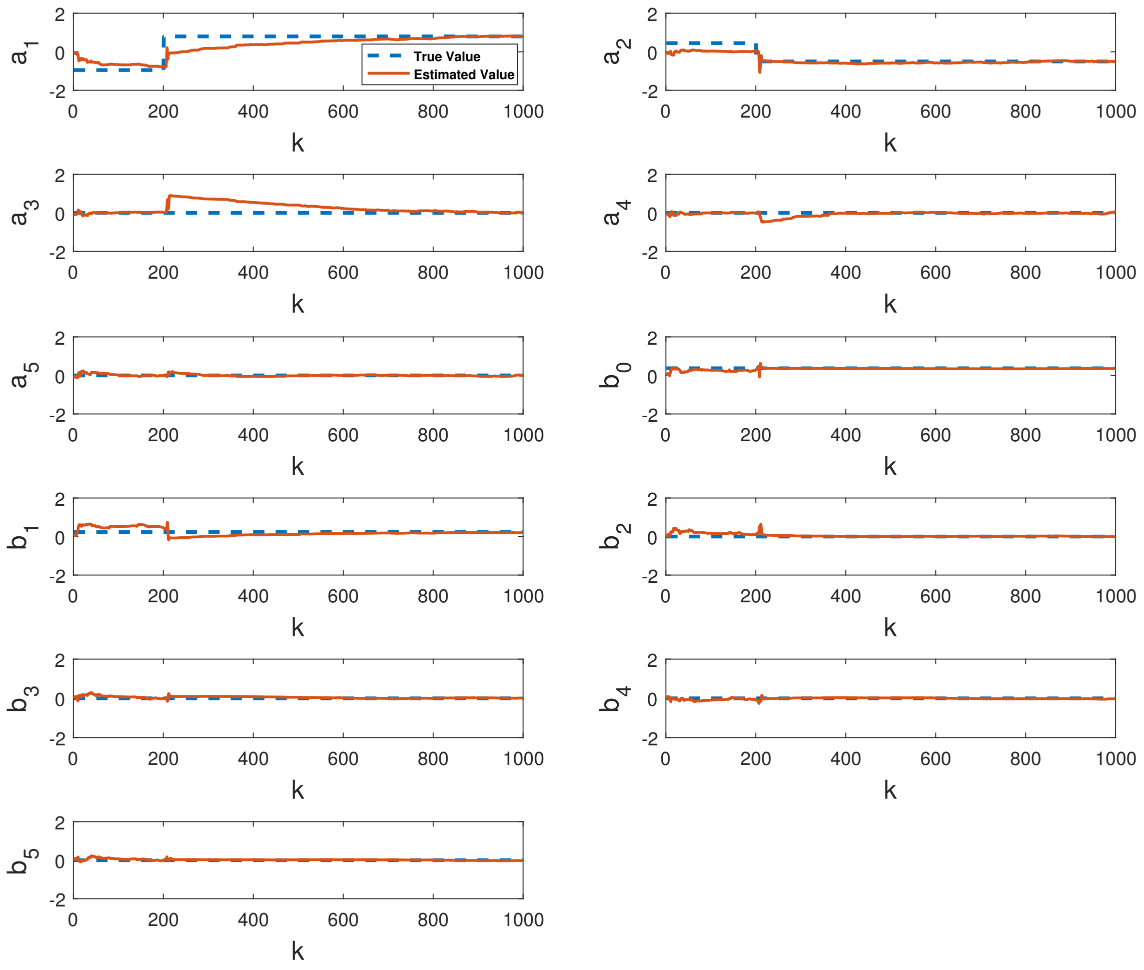

Figure 5.14: Estimated model parameters.

5.4.3 Prediction of electrical energy consumption

Examples of the problem of section 5.3 are the prediction of water and electrical energy consumption of a system, or set of systems in a certain location. Given a sequence of observations of a time series, the goal is to predict the terms associated to future observations of that time series.

The electrical energy consumption $E(k)$ can be modeled by an ARMA model of the form $\frac{C(q^{-1})}{A(q^{-1})}$, a filter that reflects the seasonality, and a constant $\bar{E}$ that reflects the mean value of consumption. The model is as follows

$$E(k) = \frac{C(q^{-1})}{A(q^{-1})} \frac{1}{1-q^{-T}} e(k) + \bar{E}. \tag{5.47}$$

Over the course of the day, the electrical energy consumption goes through some peak hours, and this behavior is similar every single day. Subsequently, these consumptions are almost periodic. This periodicity of 24 hours is the reason for the presence of a daily seasonal filter.

In a real scenario, such as the prediction of the electrical energy consumption at IST, data are collected at time intervals. The available data correspond to electrical energy consumption measurements from Pavilhão de Civil, a building of IST, from January 2014 to June 2015. Also, the data is provided in A h.

The measurements are a combination of HVAC (Heating, Ventilation and Air Conditioning) and other electric systems, such as illumination and electronic systems from classrooms. It would be advantageous to store the consumption measurement values by each of the systems so that the prediction of the HVAC system consumption alone would be possible. However, the available data correspond to the systems consumption in the building altogether.

The measurements are collected at a sampling interval of a quarter of an hour (15 min). Thereby, in each hour there are 4 data acquisitions, and consequently in each day there are 96 different data acquisitions.

The transfer function of the daily seasonal filter is then

$$\frac{1}{1-q^{-96}}, \tag{5.48}$$

which requires 96 initial conditions. Recall that, depending on the application, multiple seasonality filters may be added and applied to account for weekends, holidays, or each station of the year.

For simplification, the dataset was narrowed to the time interval from October 1, 2014 to November 30, 2014.

Figure 5.15 illustrates the original electrical energy consumption from October 1, 2014 to November 30, 2014 while Figure 5.16 corresponds to the week from October 27, 2014 to November 2, 2014.

The peaks and valleys corresponding to day time and night time are shown in Figure 5.16, and also a respectable drop in consumption on Saturday and Sunday.

Furthermore, the measurements corresponding to weekends (Saturdays and Sundays) were removed. As there are no classes in these days, both the number of persons at IST during these periods and the number of HVAC systems turned on, are far less compared to weekdays, which reflects an

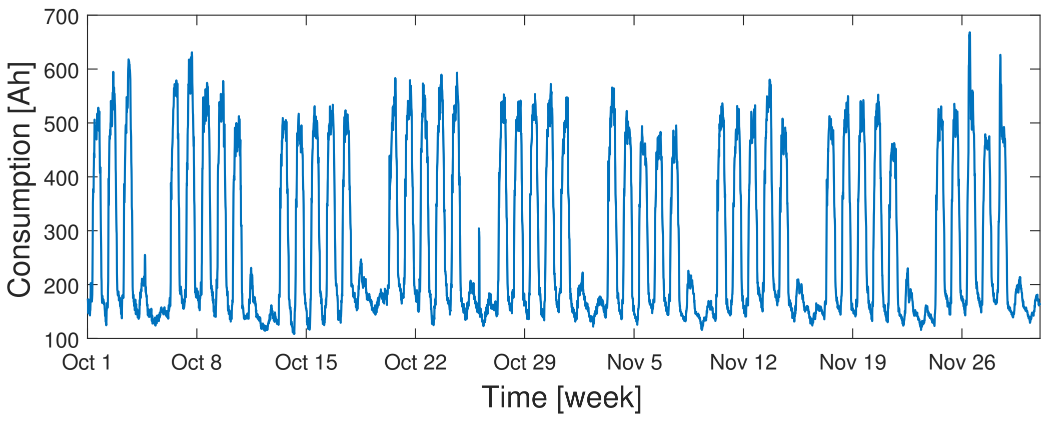

Figure 5.15: Electrical energy consumption data from October 1, 2014 to November 30, 2014.

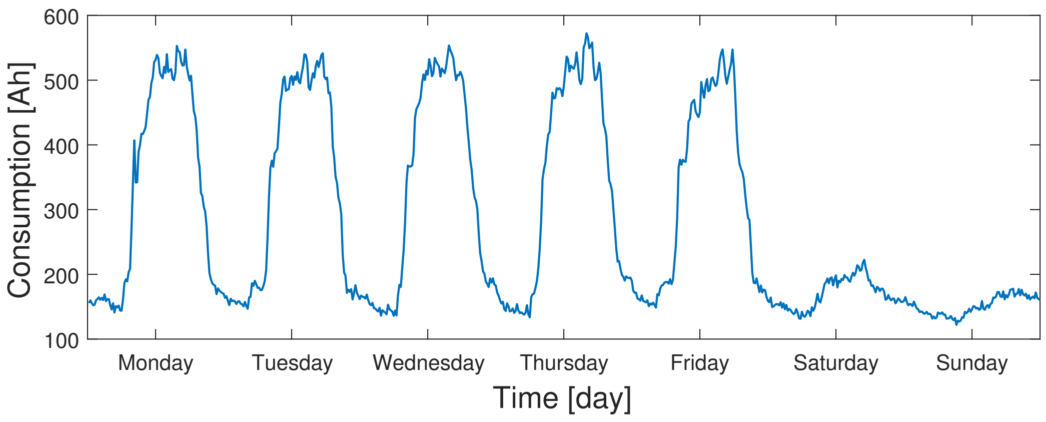

Figure 5.16: Electrical energy consumption data from October 27, 2014 to November 2, 2014.

electrical energy consumption also inferior. Therefore, the end of a Friday is succeeded by the beginning of the following week Monday. These dataset segmentations avoid the application of further seasonal filters.

Figure 5.18 depicts the same time period of Figure 5.15 after removing the weekends.

A random weekday is illustrated in Figure 5.17. The whole consumption of Pavilhão de Civil throughout the day is visible, with the peak starting at around 7 a.m. and ending at 9 p.m., when there is almost no one at IST, and thus a lot less electrical energy is consumed.

According to the section 5.3, the aim is to estimate and predict the process

$$y(k) = \frac{C(q^{-1})}{A(q^{-1})} e(k), \tag{5.49}$$

which cannot be accessed directly, as only $E(k)$ is available. Therefore, to predict its future values it is first necessary to remove the seasonality and the mean consumption value, such that

$$y(k) = \left(1 - q^{-96}\right)\left(E(k) - \bar{E}\right). \tag{5.50}$$

The inverse of the seasonal filter previously mentioned is applied. It has zeros placed on the unit circle, which cancel the poles responsible for the periodicity nature of the electrical energy consumption time series. The mean of $e(k)$ is equal to zero and so it is the mean of $y(k)$. The trend component is removed calculating the expected value of the electrical energy consumption,

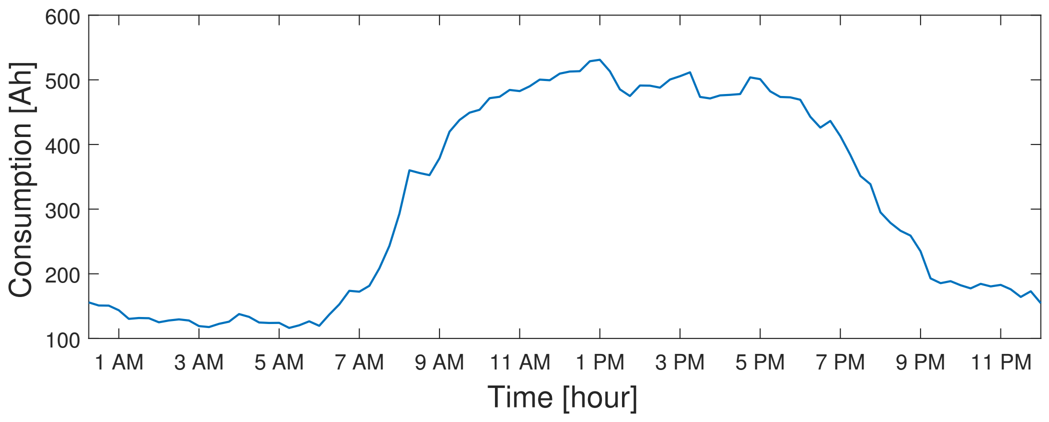

Figure 5.17: Electrical energy consumption data over one weekday.

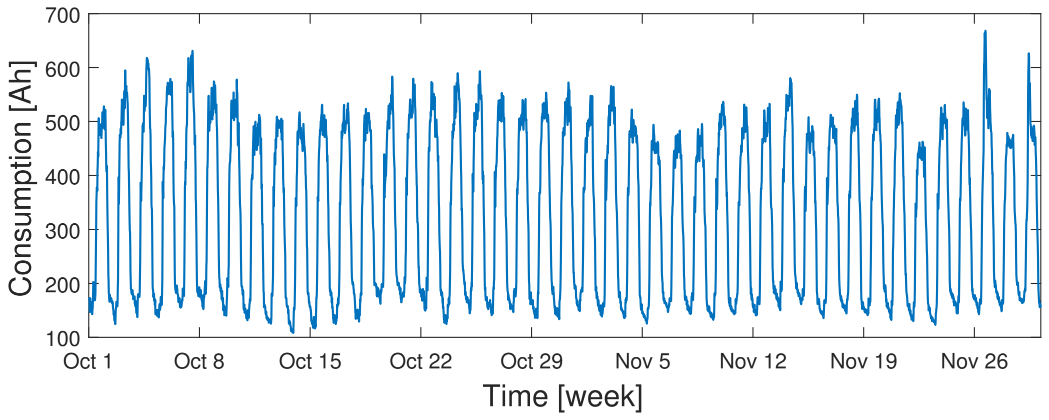

Figure 5.18: Electrical energy consumption data from October 27, 2014 to November 2, 2014, after removing the weekends consumption.

$E\left\{E(k)\right\} = \left\{y(k) + \bar{E}\right\} = \bar{E}$. The value of $\bar{E}$ can therefore be calculated using the estimator

$$\hat{\bar{E}}(N) = \frac{1}{N}\sum_{i=1}^{N} E(i), \tag{5.51}$$

where N is the number of observations. MATLAB also has functions such as *detrend* to remove the mean value or linear trend from vectors.

Figure 5.19 corresponds to the remaining process $y(k)$, after the filter has been applied and the trend component removed. In addition, when using acquired data, the datasets have to be analyzed as they may have outliers, that is, unusual observations caused by some anomalous conditions, which may need to be removed. Since no outliers were registered during the period of time from October 1, 2014 to November 30, 2014, the measurements did not need further processing.

The remaining process $y(k)$ corresponds to an ARMA model, whose parameters can be identified using the aforementioned algorithms. The estimated parameters are then used to predict future values as demonstrated in section 5.3.

Figure 5.20 shows the comparison between 200 samples of the original process $y(k)$ and the one step-ahead predictions $\hat{y}(k+1)$ after re-adding the seasonal and the mean components.

The value of the variance of $e(k)$ can be estimated from the observations using $\dfrac{A(q^{-1})}{C(q^{-1}}$, the inverse

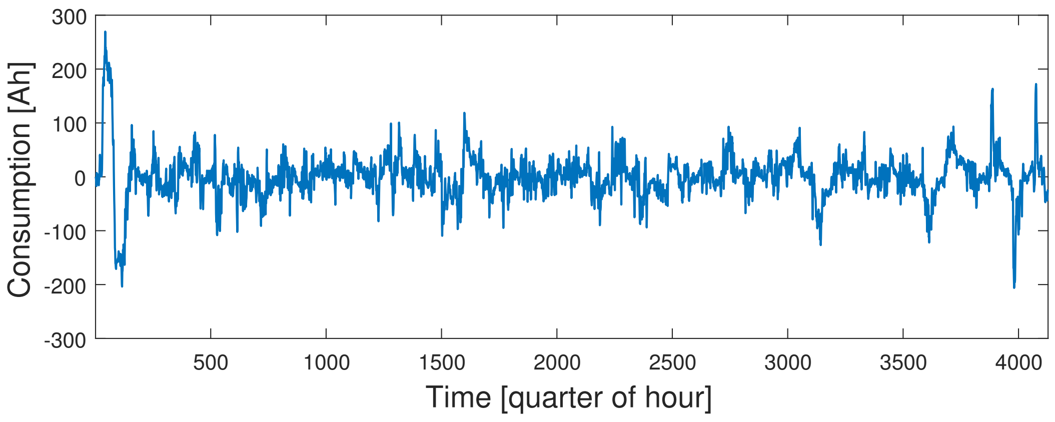

Figure 5.19: Electrical energy consumption data without seasonal part.

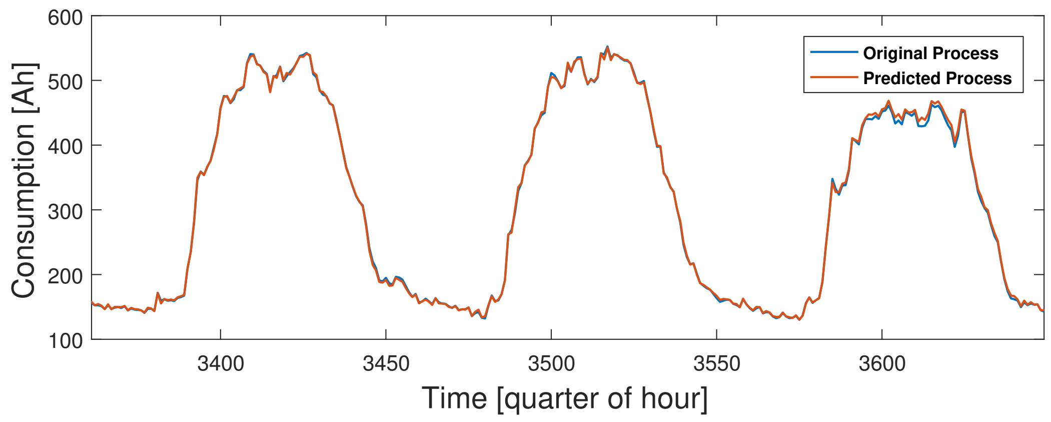

Figure 5.20: Comparison between 200 samples of the original and one step-ahead predicted electrical energy consumption processes.

of the filter $\frac{C(q^{-1})}{A(q^{-1}}$. As is, the estimates $\varepsilon(k)$ of $e(k)$ are obtained and the variance is computed using a variance estimator of the form

$$\hat{\sigma}_e^2 = \frac{1}{N}\sum_{i=1}^{N}\varepsilon^2(i), \tag{5.52}$$

where N is the number of samples of the process. If the prediction is well performed, then the prediction error, $\tilde{y}(k) = y(k) - \hat{y}(k)$, has a variance given by

$$\sigma_{\tilde{y}}^2 = E\left\{\tilde{y}(k+d)|\mathcal{O}^k\right\} = \left(1 + f_1^2 + \ldots + f_{d-1}^2\right)\sigma_e^2. \tag{5.53}$$

When $d = 1$, the variance of the prediction error is expected to be similar to that of the process $e(k)$. The variance of the prediction error is $\sigma_{\tilde{y}}^2 = 52.4624$. Comparing this value to $\sigma_e^2 = 52.4502$, a low deviance is achieved (0.12%). Furthermore, since $e(k)$ is Gaussian white noise with variance σ_e^2, then $\tilde{y}(k)$ should also have the same properties. In this respect, one possibility is to look at the power spectral density of $\tilde{y}$, the Fourier transform of its autocorrelation $R(m) = E\left\{\tilde{y}(k)\right\}E\left\{\tilde{y}(k-m)\right\}$. The Fourier transform of a constant is known to be a Dirac delta function δ. For example, the power spectral density of Gaussian white noise with variance σ^2 is flat over the whole range of frequencies and equal to σ^2, and its autocorrelation is given by

$$R(m) = \sigma^2\delta(m) = \begin{cases} \sigma^2, \text{ if } m = 0 \\ \\ 0, \text{ otherwise} \end{cases} . \tag{5.54}$$

Fig. 5.21 depicts the calculated autocorrelation of $\tilde{y}(k)$. As expected it approaches a Dirac delta function with amplitude equal to the variance of the process.

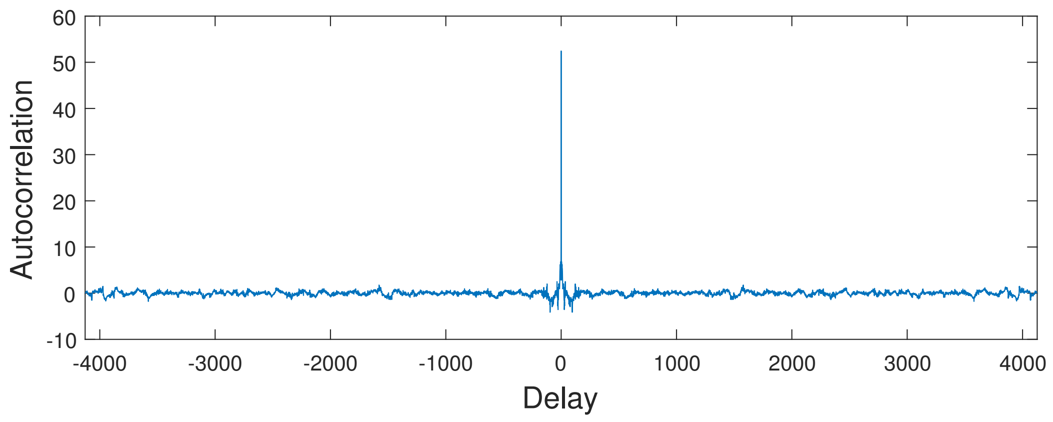

Figure 5.21: Autocorrelation of $\tilde{y}(k)$.

To analyze the effect of the prediction horizon, larger prediction horizons are considered. Figure 5.22 depicts the original electrical energy consumption process and the predicted one for $d = 1, \dots, 100$. The estimated process in the figure corresponds to the model response for 100 samples beyond the already measured 200 samples. As the prediction horizon d increases, the predicted values tend to zero. At the same time, the variance of the prediction error also increases. It can be seen that after a value of d near 40, increasing the prediction horizon becomes absolutely meaningless.

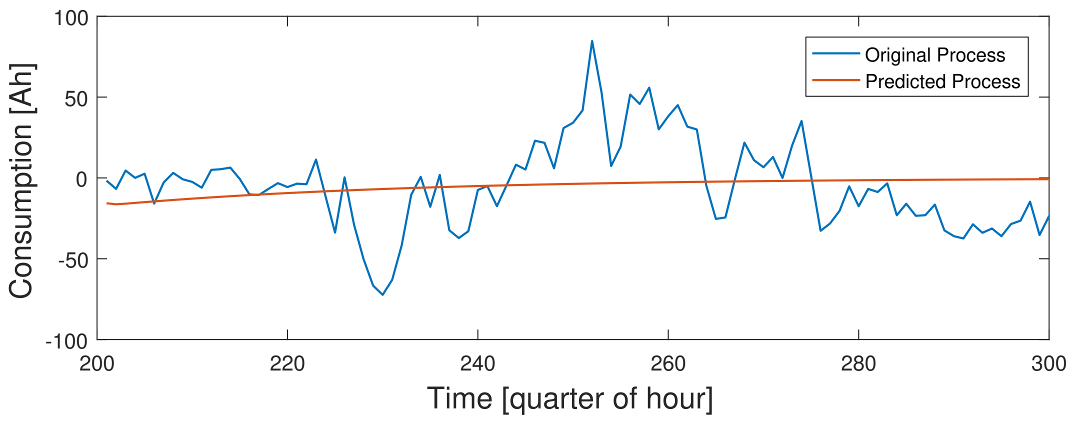

Figure 5.22: Comparison between the original process and the predicted process with horizon $m = 1, \dots, 100$.

6 Conclusions and Future Work

The aim of this thesis is to implement adaptive control strategies in order to estimate sparse models. One of the problems of many adaptive control schemes is that they require exact knowledge of the order of the model. An ARX model could be easily estimated, for example using the MATLAB *arx* command, if the true order of the system is known. However, not knowing in advance the true order of the system will hinder the results of *arx*. If wrong orders of the polynomials are initially used, the estimated parameters will not correspond to those of the true system.

On the other hand, LASSO reveals the true order information of the system without prior knowledge. This allows guessing an arbitrary system order without compromising the parameter estimation and tracking performance.

This thesis demonstrated how LASSO can be explored in order to perform recursive identification of sparse models. The identification of these systems amounts to estimate the coefficients of the system transfer function.

Chapter 2 and Chapter 3 conferred the necessary knowledge and basic ideas on sparse measures and system identification, respectively.

In Chapter 4, the algorithms adapted for the sparse model identification problem were addressed. Regularization terms containing the ℓ_1 norm were added to the cost function of well-known algorithms such as LMS and RLS. Given the non differentiable characteristic of the ℓ_1 norm, subgradients and proximity operators were applied to derive the sparsity-aware solutions. Since the ℓ_1 norm penalizes uniformly the vector of estimated parameters, weighted versions of the algorithms were also introduced.

The RZA-NLMS algorithm has a constant gain μ and its estimates approach values close to the correct ones relatively quickly, but they do not converge, as its gain does not decrease. This behavior was already expected since the disadvantages of LMS compared to RLS are well known in the literature. At least RZA-NLMS improves uppon LMS performance when estimating sparse systems.

The ℓ_1-RLS algorithm is very similar to the RLS in complexity. Numerical simulations demonstrate that this algorithm has better convergence and performance than its regular counterpart when the system to be identified is sparse.

The RW-LASSO is also based on RLS but it is relatively more complex than the ℓ_1-RLS.

MVC and GPC control laws were then coupled with a sparsity-aware algorithm to demonstrate how self-tuning adaptive controllers are capable of estimating sparse models and controlling the respective plants. One step ahead and multi step ahead predictors were used to keep tracking reference trajectories.

One of the main problems is tracking time-varying parameters and the enforced tradeoff between sparsity and tracking capability, which requires fine tuning of parameters. In this regard, variable forgetting factors provide major help, especially in abrupt changes scenarios, but at the same time the sparsity of the solution is affected.

Finally, in the future and since this work were restricted to synthetic data and generated systems, it would be far more impressive to apply these sparse identification methods to existing sparse systems, using acquired data measurements.

It would also be interesting to aim for efficient ways of extending this work to systems in which noise has a particular structure, for example, when the systems are described by ARMAX models.

One possible future work could be further research on self-adaptive parameters to control the zero-attracting terms. Or at least, categorizing values based on different classes such as SNR, noise variance or ratio between zero and nonzero parameters. This work would avoid manual calibration of said parameters for all distinct simulations.

As this work was narrowed to SISO systems, and since it can be formulated similarly, one possibility could be the extension to multi-variable (MIMO) systems.

In these past years the growing area of Artificial Intelligence led to a vast number of recent methods. An interesting extension to this work would be the use of Neural Networks to aid the identification of processes (Neural Network System Identification). For instance, the ability of GPC to make predictions depends largely on the quality of the plant model and a nonlinear plant is more difficult to model. Whether linearizing around a set of operation points, or developing a nonlinear model dependent on assumptions about the plant dynamics, incorrect assumptions will affect the accuracy of the model. As a result of their natural ability to learn and to approximate nonlinear functions, models developed using neural networks capture nonlinear dynamics. Subsequently the use of neural networks to learn plants dynamics can enhance the ability of GPC to make accurate predictions.

Bibliography

[1] M. A. Lelić and M. B. Zarrop, “Generalized pole-placement self-tuning controller part 1, basic algorithm,” *International Journal of Control*, vol. 46, no. 2, pp. 547–568, 1987. [Online]. Available: http://dx.doi.org/10.1080/00207178708933916

[2] T. Mendonca, J. M. Lemos, H. Magalhaes, P. Rocha, and S. Esteves, “Drug delivery for neuromuscular blockade with supervised multimodel adaptive control,” *IEEE Transactions on Control Systems Technology*, vol. 17, no. 6, pp. 1237–1244, November 2009.

[3] P. Gregory, *Proceedings of the Self Adaptive Flight Control Systems Symposium*. Wright Air Development Center, January 1959.

[4] K. J. Åström and B. Wittenmark, “On self tuning regulators,” *Automatica*, vol. 9, no. 2, pp. 185–199, March 1973.

[5] K. J. Åström, “Self-tuning regulators - design principles and applications,” in *International workshop on applications of adaptive control*. Academic Press, 1980.

[6] R. E. Bellman, *Adaptive Control Processes: A Guided Tour*, R. E. Bellman, Ed. MIT Press, 1961.

[7] L. Ljung, “Recursive identification algorithms,” *Circuits, Systems and Signal Processing*, vol. 21, no. 1, pp. 57–68, January 2002.

[8] T. Hastie and R. Tibshirani, *Statistical Learning With Sparsity The Lasso And Generalizations*. CRC Press, 2015.

[9] I. Rish and G. Grabarnik, *Sparse Modeling: Theory, Algorithms, and Applications*, 1st ed. CRC Press, 2014.

[10] B. Sanandaji, T. Vincent, M. Wakin, R. Tóth, and K. Poolla, “Compressive system identification of LTI and LTV ARX models,” in *50th IEEE Conference on Decision and Control and European Control Conference*, December 2011, pp. 791–798.

[11] D. Donoho, “Compressed sensing,” *IEEE Transactions on Information Theory*, vol. 52, pp. 1289–1306, April 2006.

[12] E. Candès, “Compressive sampling,” in *Proceedings of the International Congress of Mathematicians*, vol. 3, 2006, pp. 1433–1452.

[13] M. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, December 2007.

[14] L. Ljung and T. Söderström, *Theory and practice of recursive identification*. MIT press Cambridge, MA, 1983.

[15] H. Ohlsson and L. Ljung, "Identification of switched linear regression models using sum-of-norms regularization," *Automatica*, vol. 49, no. 4, pp. 1045–1050, April 2013. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2013.01.031

[16] L. Bako, "Identification of switched linear systems via sparse optimization," *Automatica*, vol. 47, no. 4, pp. 668–677, April 2011. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2011.01.036

[17] H. Ohlsson, L. Ljung, and S. Boyd, "Segmentation of ARX-models using sum-of-norms regularization," *Automatica*, vol. 46, no. 6, pp. 1107–1111, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109810001330

[18] S. Chen and D. Donoho, "Basis pursuit," in *Proceedings of the 28th Asilomar Conference on Signals, Systems and Computers*, vol. 1, Oct 1994, pp. 41–44.

[19] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1995.

[20] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.

[21] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1996.

[22] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: Algorithms and applications," *IEEE Access*, vol. 3, pp. 490–530, May 2015.

[23] Y. Gu, J. Jin, and S. Mei, "ℓ_0 norm constraint LMS algorithm for sparse system identification," *IEEE Signal Processing Letters*, vol. 16, no. 9, pp. 774–777, September 2009.

[24] Y. Chen, Y. Gu, and A. O. Hero, "Sparse LMS for system identification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 3125–3128.

[25] D. Angelosante, J. A. Bazerque, and G. B. Giannakis, "Online adaptive estimation of sparse signals: Where RLS meets the ℓ_1-norm," *IEEE Transactions on Signal Processing*, vol. 58, no. 7, pp. 3436–3447, July 2010.

[26] D. Angelosante and G. B. Giannakis, "RLS-weighted lasso for adaptive estimation of sparse signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 3245–3248.

[27] E. M. Eksioglu and A. K. Tanc, “RLS algorithm with convex regularization,” *IEEE Signal Processing Letters*, vol. 18, no. 8, pp. 470–473, August 2011.

[28] B. Babadi, N. Kalouptsidis, and V. Tarokh, “SpaRLS: The sparse RLS algorithm,” *IEEE Transactions on Signal Processing*, vol. 58, no. 8, pp. 4013–4025, August 2010.

[29] N. Kalouptsidis, G. Mileounis, B. Babadi, and V. Tarokh, “Adaptive algorithms for sparse system identification,” *Signal Processing*, vol. 91, no. 8, pp. 1910–1919, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168411000697

[30] Y. Kopsinis, K. Slavakis, and S. Theodoridis, “Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls,” *IEEE Transactions on Signal Processing*, vol. 59, no. 3, pp. 936–952, March 2011.

[31] M. S. Asif and J. Romberg, “Dynamic updating for sparse time varying signals,” in *2009 43rd Annual Conference on Information Sciences and Systems*, March 2009, pp. 3–8.

[32] M. Mishali and Y. C. Eldar, “Blind multiband signal reconstruction: Compressed sensing for analog signals,” *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 993–1009, March 2009.

[33] W. F. Schreiber, “Advanced television systems for terrestrial broadcasting: Some problems and some proposed solutions,” *Proceedings of the IEEE*, vol. 83, no. 6, pp. 958–981, June 1995.

[34] Y. C. Eldar, P. Kuppinger, and H. Bölcskei, “Block-sparse signals: Uncertainty relations and efficient recovery,” *Trans. Sig. Proc.*, vol. 58, no. 6, pp. 3042–3054, June 2010.

[35] Y. Chen and A. O. Hero, “Recursive $\ell_{1,\infty}$ group lasso,” *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 3978–3987, August 2012.

[36] M. Gallieri and J. M. Maciejowski, “Stabilising terminal cost and terminal controller for ℓasso-MPC: enhanced optimality and region of attraction,” in *2013 European Control Conference (ECC)*, July 2013, pp. 524–529.

[37] S. K. Pakazad, H. Ohlsson, and L. Ljung, “Sparse control using sum-of-norms regularized model predictive control,” in *52nd IEEE Conference on Decision and Control*, December 2013, pp. 5758–5763.

[38] M. Athans, “Minimum-fuel feedback control systems: Second-order case,” *IEEE Transactions on Applications and Industry*, vol. 82, no. 65, pp. 8–17, March 1963.

[39] C. C. Chan, “The state of the art of electric, hybrid, and fuel cell vehicles,” *Proceedings of the IEEE*, vol. 95, no. 4, pp. 704–718, April 2007.

[40] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[41] E. Candès and T. Tao, “Near-optimal signal recovery from random projections: Universal encoding strategies?” *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, December 2006.

[42] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[43] B. Natarajan, “Sparse approximate solutions to linear systems,” *SIAM Journal on Computing*, vol. 24, pp. 227–234, April 1995.

[44] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[45] H. Zou, “The adaptive lasso and its oracle properties,” *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, 2006.

[46] J. Fan and R. Li, “Variable selection via nonconcave penalized likelihood and its oracle properties,” pp. 1348–1360, 2001.

[47] Y.-L. Yu, “The proximity operator,” http://www.cs.cmu.edu/~suvrit/teach/, Pittsburgh, PA, 15213, USA, March 2014, ”Accessed 12/08/17”.

[48] N. Parikh and S. Boyd, “Proximal algorithms,” *Found. Trends Optim.*, vol. 1, no. 3, pp. 127–239, January 2014. [Online]. Available: http://dx.doi.org/10.1561/2400000003

[49] J. Picado, “On semicontinuity of real functions: an algebraic description,” http://www.mat.uc.pt/~picado/publicat/, 2008, ”Accessed 23/09/17”.

[50] G. Wanka, “Convex analysis,” https://www.tu-chemnitz.de/mathematik/fsrmathe/studium/skripte/, 2003, ”Accessed 23/09/17”.

[51] T. R. Fortescue, L. S. Kershenbaum, and B. E. Ydstie, “Brief paper: Implementation of self-tuning regulators with variable forgetting factors,” *Automatica*, vol. 17, no. 6, pp. 831–835, November 1981. [Online]. Available: http://dx.doi.org/10.1016/0005-1098(81)90070-4

[52] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1985.

[53] E. J. Candès, M. B. Wakin, and S. P. Boyd, “Enhancing sparsity by reweighted ℓ_1 minimization,” *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 877–905, 2008. [Online]. Available: http://dx.doi.org/10.1007/s00041-008-9045-x

[54] B. D. Rao and K. Kreutz-Delgado, “An affine scaling methodology for best basis selection,” *IEEE Transactions on Signal Processing*, vol. 47, no. 1, pp. 187–200, January 1999.

[55] P. Zhao and B. Yu, “On model selection consistency of lasso,” *Journal of Machine Learning Research*, vol. 7, pp. 2541–2563, December 2006.

[56] R. Goebel, R. G. Sanfelice, and A. R. Teel, “Hybrid dynamical systems,” *IEEE Control Systems*, vol. 29, no. 2, pp. 28–93, April 2009.

[57] Wikipedia. https://en.wikipedia.org/wiki/Model_predictive_control. Accessed: 2017-05-20.

[58] D. W. Clarke, C. Mohtadi, and P. S. Tuffs, “Generalized predictive control: Part i. the basic algorithm,” *Automatica*, vol. 23, no. 2, pp. 137–148, March 1987. [Online]. Available: http://dx.doi.org/10.1016/0005-1098(87)90087-2

[59] O. P. Palsson, H. Madsen, and H. T. Søgaard, “Generalized predictive control for non-stationary systems,” *Automatica*, vol. 30, no. 12, pp. 1991–1997, December 1994. [Online]. Available: http://dx.doi.org/10.1016/0005-1098(94)90061-2

[60] Z. Zidane, M. A. Lafkih, and M. Ramzi, “Simulation studies of adaptive predictive control for small hydro power plant,” *Journal of Mechanical Engineering and Automation*, vol. 2, no. 6, pp. 2163–2413, 2012. [Online]. Available: http://article.sapub.org/10.5923.j.jmea.20120206.07.html

A

Matrix Inversion Lemma

This appendix provides the proof of the Matrix Inversion Lemma.

Lemma: Let A, D and $\left[D^{-1}+CA^{-1}B\right]$ be invertible matrices. Then $A+BDC$ is invertible and

$$(A+BDC)^{-1} = A^{-1} - A^{-1}B(D^{-1}+CA^{-1}B)^{-1}CA^{-1} \tag{A.1}$$

Proof. If the equality holds, multiplying each side of the equation by $A+BDC$ must result in the identity matrix I.

As $A+BDC$ is invertible the result of the left-hand side is

$$(A+BDC)(A+BDC)^{-1} = (A+BDC)^{-1}(A+BDC) = I$$

Multiplying the right-hand side by $(A+BDC)$ on the left yields

$$\begin{aligned}
(A+BDC)(A^{-1} - A^{-1}B(D^{-1}+CA^{-1}B)^{-1}CA^{-1}) &= \\
AA^{-1} + BDCA^{-1} - AA^{-1}B(D^{-1}+CA^{-1}B)^{-1}CA^{-1} - BDCA^{-1}B(D^{-1}+CA^{-1}B)CA^{-1} &= \\
I + BDCA^{-1} - BD(D^{-1}+CA^{-1}B)(D^{-1}+CA^{-1}B)^{-1}CA^{-1} &= \\
I + BDCA^{-1} - BDCA^{-1} &= \\
I &
\end{aligned}$$

Multiplying the right-hand side by $(A+BDC)$ on the right yields

$$\begin{aligned}
(A^{-1} - A^{-1}B(D^{-1}+CA^{-1}B)^{-1}CA^{-1})(A+BDC) &= \\
A^{-1}A - A^{-1}B\left[DA^{-1}B+C^{-1}\right]DA^{-1}A + A^{-1}BCD - A^{-1}B\left[DA^{-1}B+C^{-1}\right]^{-1}DA^{-1}BCD &= \\
I - A^{-1}B\left[DA^{-1}B+C^{-1}\right]^{-1}\{D - \left[DA^{-1}B+C^{-1}\right]CD + DA^{-1}BCD\} &= \\
I - A^{-1}B\left[DA^{-1}B+C^{-1}\right]^{-1}\left[D - DA^{-1}BCD - D + DA^{-1}BCD\right] &= \\
I &
\end{aligned}$$

which proves (A.1). □

B

Diophantine Equation Solution

The purpose of this appendix is to provide a quick review of the solution of linear Diophantine equations. Diophantus of Alexandria studied in the third century A.D. the problem of finding integers (x, y) solving the equation $ax + by = c$ with a, b, and c given integers.

Consider the polynomial Diophantine equation

$$C(q^{-1}) = F(q^{-1})A(q^{-1}) + q^{-d}G(q^{-1}) \tag{B.1}$$

and the following polynomials

$$\begin{aligned}
A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_{n_a} q^{-n_a}, \\
C(q^{-1}) &= 1 + c_1 q^{-1} + \ldots + c_{n_c} q^{-n_c}, \\
F(q^{-1}) &= 1 + f_1 q^{-1} + \ldots + f_{d-1} q^{-d+1}, \\
G(q^{-1}) &= g_0 + g_1 q^{-1} + \ldots + g_{n_g} q^{-n_g},\ n_g = \max\{n_a - 1, n_c - 1\}.
\end{aligned}$$

If $A(q^{-1})$ and $C(q^{-1})$ are arbitrary polynomials in q^{-1}, then there exists two unique polynomials $F(q^{-1})$ and $G(q^{-1})$ which satisfy equation (B.1). The polynomial $F(q^{-1})$ is the quotient when dividing $C(q^{-1})$ by $A(q^{-1})$, while $q^{-d}G(q^{-1})$ is the remainder. These two polynomials can also be determined by equating the coefficients of different powers of q^{-1}. Thus

$$\begin{aligned}
c_1 &= a_1 + f_1, \\
c_2 &= a_2 + a_1 f_1 + f_2, \\
&\vdots \\
c_{d-1} &= a_{d-1} + a_{d-2} f_1 + \ldots + a_1 f_{d-2} + f_{d-1} \\
c_d &= a_d + a_{d-1} f_1 + \ldots + a_1 f_{d-1} + g_0 \\
c_{d+1} &= a_{d+1} + a_d f_1 + \ldots + a_2 f_{d-1} + g_1 \\
&\vdots \\
c_{n_g} &= a_{n_g} + a_{n_g-1} f_1 + \ldots + a_{n_g-d+1} f_{d-1} + g_{n_g-d} \\
0 &= a_{n_g} f_1 + a_{n_g-1} f_2 + \ldots + a_2 f_{n_g-1} + g_{n_g-d+1} \\
&\vdots \\
0 &= a_{n_g+1} f_{d-1} + g_{n_g}
\end{aligned}$$

C

GPC for time-varying systems with no Diophantine equations

The GPC is based on the assumption that the output predictions can be expressed as a linear combination of present and future controls and most of the times it is performed by solving Diophantine equations. However, in scenarios with constantly varying parameters, the Diophantine equation is not the best option [59]. Additionally, an alternative method using predictors may be used.

$$h_i(k) = \begin{cases} b_i(k), & \text{if } i = 1 \\ b_i(k) - a_{i,k}h_{i-1}(k-1), & \text{if } i = 2 \\ -a_i(k)h_{i-1}(k-1), & \text{if } i \geq 3 \end{cases}, \tag{C.1}$$

and

$$v_i(k) = \begin{cases} -a_1(k+i)y(k) + b_2(k+i)u(k-1), & \text{if } i = 1 \\ -a_1(k+i)v_{i-1}(k), & \text{if } i \geq 2 \end{cases}, \tag{C.2}$$

$$\mathbf{v}(k) = \begin{bmatrix} v_1(k) \\ \vdots \\ v_N(k) \end{bmatrix}. \tag{C.3}$$

The matrix $\mathbf{W}(k)$ is filled with values as follows

$$\mathbf{W}(k) = \begin{bmatrix} h_1(k+1) & 0 & \dots & 0 & 0 \\ h_2(k+2) & h_2(k+2) & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{N-1}(k+N-1) & h_{N-2}(k+N-1) & \dots & h_1(k+N-1) & 0 \\ h_N(k+N) & h_{N-1}(k+N) & \dots & h_2(k+N) & h_1(k+N) \end{bmatrix} \tag{C.4}$$

$\mathbf{y}(k) = \mathbf{W}(k)\Delta\mathbf{u}(k) + \mathbf{v}(k)$

This setup maintains the features and advantages of the classical approach detailed previously and allows for better tracking of time varying parameters.