SVILUPPO DI STRATEGIE DI MONITORAGGIO ED IDENTIFICAZIONE DI CONTROLLORI PREDITTIVI

ANDREA MICCHI

Relatori:

ALESSANDRO BRAMBILLA GABRIELE PANNOCCHIA

Dottorato di ricerca in Ingegneria Chimica e dei Materiali

Dipartimento di Ingegneria Chimica, Chimica Industriale e Scienza dei Materiali
Facoltà di Ingegneria
Università di Pisa

Anno 2008

IDENTIFICATION AND MONITORING OF MULTIVARIABLE PREDICTIVE CONTROLLERS

ANDREA MICCHI

Advisors:

ALESSANDRO BRAMBILLA GABRIELE PANNOCCHIA

Phd Thesis in Chemical and Material Engineering

Department of Chemical Engineering, Industrial Chemistry
and Material Science (DICCISM)
Engineering faculty
University of Pisa

Year 2008

Dedicated to my family

RINGRAZIAMENTI

Desidero ringraziare innanzitutto il Professor Brambilla e l'Ingegner Pannocchia per il supporto e i consigli durante questi tre bellissimi anni vissuti come dottorando al CPCLab.
Un ringraziamento particolare va anche al professor Joe Qin della University of Southern California, per l'ospitalità e l'aiuto che mi ha fornito nella ricerca.
E, soprattutto, grazie alla mia famiglia, papà, mamma e Gianluca, che mi ha sempre sostenuto in tutto e specialmente nel periodo trascorso negli USA, dove non ha fatto mai mancare il suo appoggio pur con le difficoltà che la distanza può creare.

Da ultimo, non posso fare a meno di salutare tutti coloro che in questi tre anni ho conosciuto a Pisa e in California. Sono persone speciali, e da ognuno di loro ho imparato qualcosa. Mi sia concesso di ricordare, senza che gli altri se ne abbiano a male, i dottorandi (in rigoroso ordine sparso) Aurora, Martina, Gabriele, Alessandro, e Matteo, compagni di tante avventure e di tanti caffè, e i CPCLabbisti "storici" Alessandro, Federico, Martina, Claudio, Silvia e Mirco. Infine un saluto particolare va a Carlos e Edoardo, fondamentali per la mia sopravvivenza sotto il sole della California.

ACKNOWLEDGMENTS

First, I would like to thank Prof. Alessandro Brambilla and Ing. Gabriele Pannocchia for their support and their advice during these fantastic years as a PhD in the CPCLab.
Moreover, many thanks to Prof. Joe Qin of Southern California University, for his hospitality and his help in my researches.
Last but not the least, thanks to my family, father, mother and brother, for their support and patience, especially during the period I spent in the USA, in which the distance between us made everything they did for me more precious.

In the end, greetings to everyone I met in Pisa and in California. They are special, and I learnt something important from everyone of them. In particular, I like to mention Aurora, Martina, Gabriele, Alessandro and Matteo, PhD students at the University of Pisa, and Alessandro, Federico, Martina, Claudio, Silvia, Mirco, thesis students at the CPCLab. In the end, thanks to Carlos and Edoardo for their help during the months I spent in California.

CONTENTS

1 PROCESS CONTROL AND IDENTIFICATION: AN INTRODUCTION 5
- 1.1 Process control: a basilar operation 5
- 1.2 Process representations 6
  - 1.2.1 TF models 7
  - 1.2.2 ARX models 9
  - 1.2.3 OE models 9
  - 1.2.4 State space models 10
- 1.3 Identification: obtaining models from data 13
  - 1.3.1 Experiment design 13
- 1.4 Identification techniques 16
  - 1.4.1 Introduction to PEM techniques 17
  - 1.4.2 Subspace identification methods 19

2 MPC SCHEME: STANDARD FORMULATION AND ASSUMPTIONS 29
- 2.1 MPC in the plant 30
- 2.2 Internal structure of the MPC 31
  - 2.2.1 The observer 33
  - 2.2.2 The steady-state optimization 34
  - 2.2.3 The dynamic optimization 36
- 2.3 Some final points on MPC 39
  - 2.3.1 Disturbance model choice 39
  - 2.3.2 MPC tuning 41

3 IDENTIFICATION OF MIMO ILL-CONDITIONED SYSTEMS 45
- 3.1 Introduction and previous work 45
- 3.2 Design and analysis of input signals for ill-conditioned processes 48
  - 3.2.1 Design of rotated inputs 48
  - 3.2.2 Issues in model order recovery 51
- 3.3 Case studies 52
  - 3.3.1 Introduction 52
  - 3.3.2 Inputs and outputs plots (Example #1) 53
  - 3.3.3 Model order recovering (Example #1) 56
  - 3.3.4 Quality of the identified models 57
  - 3.3.5 Effect of the identified models on MPC closed-loop behavior 58
  - 3.3.6 Practical issues in implementing rotated inputs 60
- 3.4 Conclusions 60

4 SUBSPACE IDENTIFICATION OF UNSTABLE SYSTEMS 63
- 4.1 Introduction 63
- 4.2 Basic definitions and assumptions 65
- 4.3 A subspace method for unstable systems 65

4.3.1 Issues in calculating B in unstable systems 66
4.3.2 Reorganizing the state space model of the process 67
4.3.3 Defining a stable predictor for the model 68
4.3.4 Computation of $\mathcal{A}(q)$ and $\mathcal{B}(q)$ 70
4.4 Case study 74
4.4.1 CSTR reactor and simulation generalities 75
4.4.2 Model identification results 77
4.4.3 Comparison between identified model based MPC 80
4.5 Conclusions 81

5 MPC MONITORING USING PREDICTION ERROR SEQUENCE 85
5.1 Introduction 85
5.2 Generalities and notation 87
5.2.1 Real system 87
5.2.2 MPC process model 87
5.2.3 Prediction error definition 88
5.3 Prediction Error Analysis 89
5.3.1 Autocorrelation in a time series 89
5.3.2 Definition of θ index 92
5.3.3 Finding a generating process for the PE sequence 94
5.3.4 Definition of the PE generating process 95
5.3.5 Analysis of the observability of the PE generating system 97
5.4 Obtaining the observability matrix order from data 99
5.4.1 Summary of the proposed monitoring method 99
5.5 Simulation example 100
5.5.1 Generalities on simulations 101
5.5.2 Results in case of no mismatches 104
5.5.3 Mismatch in the process model 107
5.6 Conclusions 110

6 CONCLUSIONS 113

LIST OF FIGURES

Figure 1 Feedback control scheme 6
Figure 2 White noise sequence 7
Figure 3 Position of poles and related effect on system stability in the complex plane for discrete-time processes. 8
Figure 4 ARX model representation in a block scheme 9
Figure 5 OE model representation using a block scheme 10
Figure 6 Description of the identification process 14
Figure 7 Step signal 16
Figure 8 GBN signal 17
Figure 9 Data fitting example 19
Figure 10 Oblique projection: graphical interpretation 20
Figure 11 Orthogonal projection: graphical interpretation 21
Figure 12 Orthogonal projection: numeric example 21
Figure 13 Hierarchy in a plant control system 31
Figure 14 MPC position in the scheme of a plant 31
Figure 15 MPC internal structure 32
Figure 16 Graphical representation of the observer module 33
Figure 17 Graphical representation of the steady state module 34
Figure 18 Graphical representation of the dynamic module 37
Figure 19 Input disturbance model 39
Figure 20 Output disturbance model 40
Figure 21 MPC tuning: heavy weights on input variations, inputs (top) and outputs (bottom) 43
Figure 22 MPC tuning: heavy weights on set-points, inputs (top) and outputs (bottom) 44
Figure 23 Example #1: standard GBN input (top), rotated GBN input (middle) and closed-loop input (bottom) signals. 55
Figure 24 Example #1: alignment of inputs (left) and outputs (right) in three cases: OL data collection with random inputs, OL data collection with rotated inputs, CL data collection with random setpoints. 56
Figure 25 Example #1: density function of $\bar{\epsilon}$ obtained from a Monte-Carlo study. 57
Figure 26 Model error parameter $\epsilon(\omega)$ vs. frequency for three identified models: Example #1 (top) and Example #2 (bottom). 59
Figure 27 Example #1: closed-loop inputs and outputs during a setpoint change in the direction $[1,\ 1]^T$. 60

Figure 28 Example #2: closed-loop inputs and three outputs during a setpoint change on the product compositions. 61

Figure 29 CSTR reactor 75

Figure 30 CSTR heat-material balance 76

Figure 31 Poles values for system in eq. 4.46 for the proposed method and projection method (top) and for N4SID identification method (bottom) 78

Figure 32 Poles convergence for system in eq. 4.46 for increasing number of data. #3 equilibrium point 79

Figure 33 Step response for MPC based on identified models. 82

Figure 34 Generation of a colored noise sequence 91

Figure 35 Plots of a white noise sequence (left) and of a colored noise sequence (right) 91

Figure 36 Autocorrelation plots for a white noise sequence (left) and for a colored noise sequence (right) 91

Figure 37 Graphical representation of the parameters for the calculation of θ 93

Figure 38 Values assumed by θ parameter when parameter n^{oc}_{max} varies in case of no model nor noise covariance mismatch. $oc^{max}_{lim} = 1.5\varepsilon_{95}$ 102

Figure 39 Values assumed by θ parameter when parameter oc^{max}_{lim} varies in case of no model nor noise covariance mismatch. $n^{oc}_{max} = 7$ 103

Figure 40 Inputs(top) and outputs(bottom) for the case of no model nor noise covariance mismatch 105

Figure 41 Prediction error sequence (top) and relative $\mathbf{r}(\tau)$ parameter (bottom) for the case of no model nor noise covariance mismatch 106

Figure 42 Inputs(top) and outputs(bottom) for the case of model mismatch 107

Figure 43 Prediction error sequence(top) and relative $\mathbf{r}(\tau)$ parameter (bottom) for the case of model mismatch 108

Figure 44 Inputs(top) and outputs(bottom) for the case of covariance matrices mismatch 109

Figure 45 Prediction error sequence(top) and relative $\mathbf{r}(\tau)$ parameter (bottom) for the case of covariance matrices mismatch 110

LIST OF TABLES

Table 1 CVs & MVs for the scheme in figure 14 32
Table 2 LP and QP expression for the steady state module 35
Table 3 QP expression for the dynamic module 37
Table 4 Tuning parameters for example in figures 21 and 22 41
Table 5 Inputs and outputs for Example #2. 54
Table 6 Values assumed by $\hat{A}^j$ with A from the (4.3) as j increases 67
Table 7 CVs, MVs and outputs for CSTR process of figure 29 75
Table 8 Equilibrium points values for CSTR heat-material balance of figure 30 75
Table 9 $\beta(\omega)$ values for proposed method(left), projection method(center), and N4SID(right) 80
Table 10 MPC tuning matrices used to collect data for identification 80
Table 11 Values of Φ parameter for the MPC systems of figure 33 81
Table 12 θ value for autocorrelation sequences in figure 36 93
Table 13 Number of autocorrelated systems in the Monte-Carlo simulation varying n^{oc}_{max} value. $oc^{max}_{lim} = 1.5\varepsilon_{95}$ 102
Table 14 Number of autocorrelated systems in the Monte-Carlo simulation varying oc^{max}_{lim} value. $n^{oc}_{max} = 7$ 103
Table 15 Number of autocorrelated systems in two Monte-Carlo simulations with mismatches. $oc^{max}_{lim} = 1.5\varepsilon_{95}$, $n^{oc}_{max} = 7$ 104

INTRODUCTION

Process optimization represents an important task in the management of industrial plants. It consists in finding the most appropriate working conditions for plant variables in order to maximize the profits. In other words, an optimization scheme steers the plant inputs and outputs considering the investment for energy and raw materials and the income from plant products, and respecting the constraints which can be present in the plant. Given these premises, it is clear that in industrial practice there is a high interest in the implementation of efficient and reliable optimization methods for plants.

At the moment, one of the most successful optimization scheme is represented by MPC. This is an acronym standing for "Model Predictive Controller". MPC optimizes the process to which it is applied "predicting" the state of the system over a future time window, using a model of the same process as a part of the internal structure of the controller. MPC has been used since the last two decades, and at the moment it represents a proven optimization scheme, with thousands of applications in chemical and petrochemical industry.

Once the MPC scheme for a system is defined, it is important to check regularly for its performances in order to guarantee optimal operation in spite of unknown disturbances and/or changes in the process dynamics. The operation of checking performances is named "Performance Monitoring" or simply "Monitoring", and it represents a basilar task for the reasons previously introduced. Performance monitoring, in despite of its importance, was not widely treated in the past, and only in the last years this operation has received the attention it needs. A monitoring technique should be able to discern the cases in which the optimization scheme is working in optimal or sub-optimal conditions, and, in this last case, it should recognize the causes of performance degradation.

In the literature, the causes of performance degradation are usually two, i. e. inadequate estimation of unknown disturbances and a mismatch between the internal model and the real process. Both of them will be analyzed in the following chapters and in particularly in chapter 5. In the first case, the operations that are needed to correct the mistake consist in a better definition of the noise level of the system and in the calculation of a new estimator using the correct disturbance information. In the second case, the only way to improve the performances of the system is the definition of a new model. As it will be shown in chapter 1, this is a complex task, which takes a long time and requires a particular attention. The definition of a model for the system is called "Identification", and several different methods for performing it can be found in the literature. An identification technique uses

input and output data sets coming from the system to define a model of the process. This model can assume different formulations, as it will be explained in chapter 1, but in this work in particular steady space models will be considered for the description of processes. These models are particularly interesting, because they have a compact structure which can be easily used inside an MPC.

In this thesis, techniques for systems which can present difficulties in identification will be introduced, i. e. unstable systems and ill-conditioned systems, and a monitoring technique for optimization schemes, tailored on MPC structure, will be discussed.

Unstable and ill-conditioned systems present problems in identification in a large number of cases. The first ones cannot be identified with a class of identification schemes that perform a particular regression on data, as it will be discussed in chapter 4, because numerical problems arise for the presence of high powers of unstable matrices. This work presents an extension of their structure which permits to handle data coming from unstable system.

On the other hand, ill-conditioned processes give problems because data coming from this kind of processes are aligned in a particular direction, called "strong direction". For this reason, a high level of information is present in the data set for that direction, but a low information level is present for the others, resulting in models which cannot describe the system adequately in all directions.

Finally, the problem of MPC monitoring in this work is addressed analyzing the difference between the value of the real outputs coming from the system and the value of outputs predicted by the internal MPC model, which is usually indicated as "Prediction error".

This analysis takes into account the statistical properties of the previously mentioned prediction error, in order to define if the optimization scheme works in sub-optimal conditions. Then, if this analysis shows the presence of some issues, the cause that generates these issues is determined checking the rank of a particular matrix obtained from data, that is the observability matrix. This matrix and its meaning will be described extensively in chapter 1.

This thesis is organized as follows.

- In chapter 1 a general introduction to process control and identification is reported, with the aim of presenting the theoretical basis of this work.
- In chapter 2 a standard formulation of an MPC algorithm is presented, describing the internal structure of the controller and the most common choices for its tuning.
- In chapter 3 an identification method for ill-conditioned systems is reported, focused on the analysis of the behavior of models obtained via the use of different kinds of signals.

- In chapter 4 a first result on identification is introduced, that is a method for the identification of unstable systems.
- In chapter 5 an MPC monitoring technique based on the analysis of information coming from the mismatch between predicted outputs and real ones is presented.
- In chapter 6 conclusions for this thesis are reported.

1 PROCESS CONTROL AND IDENTIFICATION: AN INTRODUCTION

Contents

1.1 Process control: a basilar operation 5
1.2 Process representations 6
1.2.1 TF models 7
1.2.2 ARX models 9
1.2.3 OE models 9
1.2.4 State space models 10
1.3 Identification: obtaining models from data 13
1.3.1 Experiment design 13
1.4 Identification techniques 16
1.4.1 Introduction to PEM techniques 17
1.4.2 Subspace identification methods 19

Process control and identification have a strong theoretical characterization. They need a large knowledge in mathematics, especially in linear algebra. In addition, as in every field, there is a wide use of technical terms. In this chapter the most important concepts are reported and analyzed. It is suggested, if further information are needed, to refer to Ljung [20], who presented a comprehensive work on these arguments.

This chapter is organized as follows.

- In section 1.1 a basic introduction to process control is presented, with the aim and meaning of this operation.
- In section 1.2 some kind of process model and input types are analyzed and their characteristics are reported.
- In section 1.3 basis for identification are reported, including identification schemes and input types.
- In the end, in section 1.4, the two major families of identification methods are introduced, that is Prediction Error Methods (PEM) and Subspace Identification Methods (SIM): for the latter, a deeper analysis is presented.

1.1 PROCESS CONTROL: A BASILAR OPERATION

Process control is an important task to be performed for plant management. It consist in checking the value of some process variables and in defining the operations needed to drive these variables to the desired values.

It is obvious that this is an important operation also in the sense of plant safety. Actually, if process parameters are at the right value, no problems of instabilities can arise. Similarly, maintaining process parameters to certain desired values guarantees also that quality limits are met.
In practice, it is common to distinguish between "base" and "advanced" controllers: the first ones are the classical PID controllers, which are usually SISO (Single Input Single Output), that is they control one output moving a single input. On the other hand, the second ones are usually MIMO (Multiple Inputs Multiple Outputs), that is they can take into account the interactions between different inputs and outputs so they can move several inputs at the same time to control several outputs.
In this work, discrete-time feedback controllers are considered. This definition can be explained with the following two points.

DISCRETE-TIME : control action and input/output values collection are not performed continuously but at given times. Time interval between two different samplings is usually fixed, and it is called "sampling time".

FEEDBACK : the present value of output variables is used for the definition of the control action needed to reach the set-point.

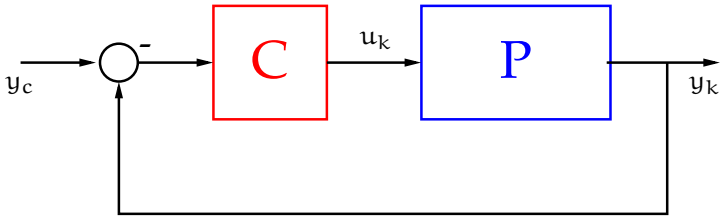

Figure 1: Feedback control scheme

The block scheme of a classical feedback controller is reported in figure 1. In this figure, y is an output variable, that is the variable to be controlled, while u is an input variable, that is the variable moved to modify the value of y; y_c is the desired value for y, usually known as set-point. Finally, subscript k represents the considered sampling time.
P is the function that represents the behavior of y when u is changed: it is possible to refer to it with different terms, such as plant, system, process, etc. .
In the end, C is the actual controller for the system, that is the block which designs an appropriate u for y to be equal to y_c.

1.2 PROCESS REPRESENTATIONS

To describe the process P in figure 1, various kinds of models were introduced during years, with different characteristic and

usability. In this chapter only the ones which will be considered in this work are introduced and analyzed extensively. The formulation of everyone of the model types which will not be presented in this chapter (such as ARMAX, Box-Jenkins, etc.) were analyzed in a large number of works (see Ljung [20] for example), so it is suggested to refer to these work if more information is needed.
So, in this chapter the attention will be focused only on four types of models, namely TF, ARX, OE and state space models.
In all of them, white noises are present in order to describe the stochastic variations of parameters. White noise is a random noise for which occurrences at time k are independent from the occurrences at previous times. In other words, supposing v represents a white noise sequence, v_k is independent by $v_0, \ldots, v_{k-1}$. A white noise sequence is plotted in figure 2: more information on its characteristics will be reported later in chapter 5, where in addiction colored noise characteristics will be described. This last kind of noise will be analyzed later, and at this point it is sufficient to say that a colored noise is a white noise filtrated by a certain dynamic block.

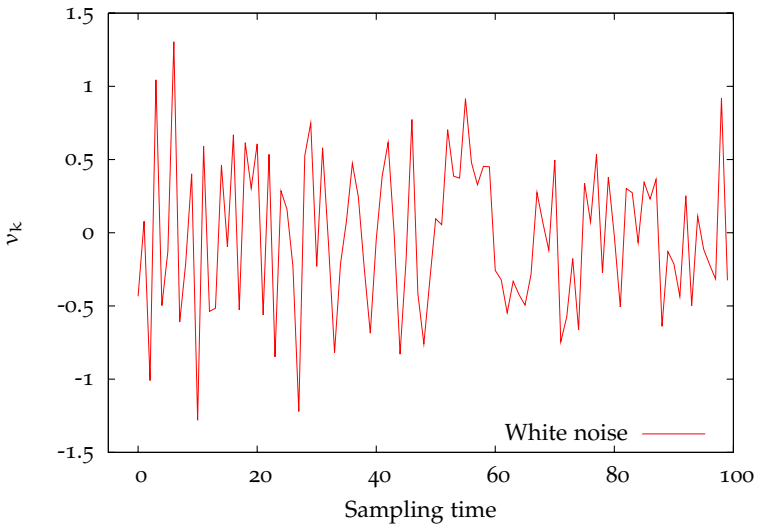

Figure 2: White noise sequence

1.2.1 TF models

TF stands for Transfer Function: indeed, it represents the function that "sends, transfers" the value of inputs to the value of the outputs. Dealing with a SISO system for simplicity, transfer

functions can be expressed using different forms, as it is possible to see in the (1.1):

$$\begin{aligned} y_k &= \frac{\prod_{j=1}^{l}(q^{-1}+b_j)}{\prod_{j=1}^{n}(q^{-1}+a_j)}u_k + v_k = \\ &= \frac{\sum_{j=0}^{l}\beta_j q^{-j}}{\sum_{j=0}^{n}\alpha_j q^{-j}}u_k + v_k = G(q)u_k + v_k \end{aligned} \tag{1.1}$$

the values of poles and zeros of the system in this form are respectively $-1/a_j$ and $-1/b_j$. q is the forward shift operator, that is the operator that performs a forward time shift of variables, as showed in eq. 1.2.

$$qu_k = u_{k+1} \tag{1.2}$$

Consequently, q^{-1} represents the backward shift operator. Dealing with discrete processes, for the system to be stable all the poles must be included in the unitary circle in the complex plane. In figure 3 a stable and an unstable pole are showed. Usually,

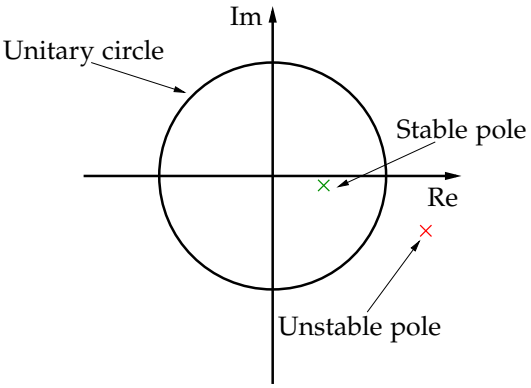

Figure 3: Position of poles and related effect on system stability in the complex plane for discrete-time processes.

processes are strictly causal. For this reason, it happens that β_0 is equal to 0 in almost the totality of the cases. Furthermore, as a conventional assumption, it is common to consider $\alpha_0 = 1$, even if this has not a particular physical meaning. It can be easily demonstrated, indeed, that every process can be described in such a way that $\alpha_0 = 1$ (it is sufficient to divide all the terms for α_0 if it differs from 1). In case the system is represented by a MIMO model, the transfer function is composed by a $p \times m$ matrix of transfer function, where p represents the number of outputs and m the number of inputs respectively. This means that, for a MIMO system,

$$G(q) = \begin{bmatrix} G_{11} & \dots & G_{1m} \\ \vdots & \dots & \vdots \\ G_{p1} & \dots & G_{pm} \end{bmatrix} \tag{1.3}$$

where G_{ij} represents the transfer function from the j-th input to the i-th output.

1.2.2 ARX models

ARX models are represented in a graphic way in figure 4 and their analytical expression is written in eq. 1.4:

$$\mathcal{A}(q)y_k = \mathcal{B}(q)u_k + v_k \tag{1.4}$$

ARX acronym stands for AutoRegressive with eXogenous inputs. The autoregressive part is the one given by the term $\mathcal{A}(q)y_k$, which performs a regression on y_k data; the exogenous part is the one that refers to $\mathcal{B}(q)u_k$, which introduces the external inputs in the system.

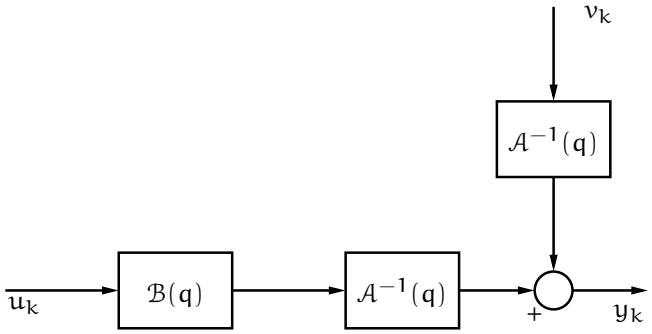

Figure 4: ARX model representation in a block scheme

Both of these terms can be expanded as in the (1.5) and in the (1.6)

$$\mathcal{A}(q) = \sum_{j=0}^{n_a} \mathcal{A}_j q^{-j} = I + \sum_{j=1}^{n_a} \mathcal{A}_j q^{-j} \tag{1.5}$$

$$\mathcal{B}(q) = \sum_{j=0}^{n_b} \mathcal{B}_j q^{-j} = 0 + \sum_{j=1}^{n_b} \mathcal{B}_j q^{-j} \tag{1.6}$$

where $\mathcal{A}_j \in \mathbb{R}^{p\times p}$ and $\mathcal{B}_j \in \mathbb{R}^{p\times m}$ being m and p the number of inputs and outputs respectively, and n_a and n_b the orders of the ARX model.

As seen for the TF model, $\mathcal{A}_0 = I$ is a conventional assumption, while $\mathcal{B}_0 = 0$ is necessary for the system to be strictly causal.

1.2.3 OE models

Output Error (OE) model block scheme is reported in figure 5. The difference between OE model and ARX model is that in the OE model the error v_k is not filtered by any specific dynamic block. For this reason, the stochastic part of the equation is considered to have the characteristics of a white noise. The formulation of OE is the following

$$y_k = \mathcal{F}^{-1}(q)\mathcal{B}(q)u_k + v_k \tag{1.7}$$

where

$$\begin{aligned} F(q) &= I + \mathcal{F}_1 q^{-1} + \mathcal{F}_2 q^{-2} + \cdots + \mathcal{F}_n q^{-n} \\ B(q) &= 0 + \mathcal{B}_1 q^{-1} + \mathcal{B}_2 q^{-2} + \cdots + \mathcal{B}_l q^{-l} \end{aligned} \tag{1.8}$$

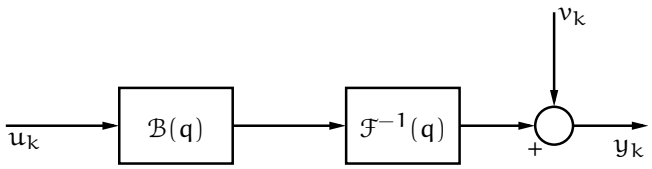

Figure 5: OE model representation using a block scheme

$\mathcal{F}_i$ and $\mathcal{B}_i$ can assume matrix or scalar values, while I and 0 represent respectively an identity matrix and a zero matrix of suitable dimensions. As seen for the transfer function model, $\mathcal{B}_0 = 0$ is required in order the system to be strictly causal.

1.2.4 State space models

State space models are the favorite kind of models for usage in the internal structure of MPC, especially in academic research. This is due to several reasons, mainly related to simplification in the identification of system models and to a major usability of those.
In other words it is possible to say that:

- state space models are particularly suitable for being used in an algorithm. Their structure, which involves only linear operations and uses no polynomial, can be easily described using different programming languages, such as MATLAB for example.
- identification of state space models is faster and simpler. This is due to the fact that a widely used class of identification methods is usually designed on the characteristics of this kind of models, that is subspace identification algorithms.

The standard expression of a SS model is reported in eq. 1.9

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + Du_k + v_k \end{aligned} \tag{1.9}$$

In the (1.9) A, B, C, D represent the process matrices, while $x_k \in \mathbb{R}^n, y_k \in \mathbb{R}^p, u_k \in \mathbb{R}^m$ are respectively the states of the plant, the outputs and the inputs to the system. In the end, w_k and v_k are respectively process noise and measurement noise.
States vector represents a series of variables, which can have in some cases a physical meaning, used to describe the state of the plant at sampling time k.
D matrix assumes the value of 0 for process systems, because it is needed for the system to be strictly causal. In case it were $D \neq 0$, the inputs would have a zero sampling time delay effect on the outputs, that is an unrealistic situation in plants. From now on, in the description of state space models, it will be considered $D = 0$.

It is possible to demonstrate that system poles in (1.1) are the eigenvalues of A matrix in (1.9). When a discrete system is stable, all the eigenvalues of A matrix are inside the unitary circle in the complex plane, as previously said. In this case, that A matrix is also known as Hurwitz in a discrete sense.

In MPC, for reasons that will be showed in the following chapters, it is common to add p fictitious integral disturbances to the model of the process, obtaining the expression in eq. 1.10

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_d d_k + w_k^x \\ d_{k+1} &= d_k + w_k^d \\ y_k &= Cx_k + C_d d_k + v_k \end{aligned} \tag{1.10}$$

In (1.10) $d_k \in \mathbb{R}^p$, and the pair B_d, C_d is known as disturbance model. There are standard representations for it that can be used, but, as it was showed by Odelson et al.[25] everyone of them is equivalent.

Model properties

SS model in (1.9) can enjoy some important properties in dependence of the value of its inner matrices, which are controllability (or stabilizability) and observability (or detectability). In this section, a brief description for all of them is reported:

Controllability and stabilizability: controllability measures the capacity of reaching a desired state starting from another casual one. This means that, if the system is controllable, it is possible to steer the state from the initial value to a desired one in a certain time window.
Suppose to have the state equation of (1.11):

$$x_{k+1} = Ax_k + Bu_k \tag{1.11}$$

Starting from time zero (and supposing x_0 is equal to zero for simplicity) it is possible to make some substitutions obtaining

$$\begin{aligned} &x_1 = Bu_0; \\ &x_2 = Ax_1 + Bu_1 = ABu_0 + Bu_1; \\ &\vdots \\ &x_k = A^{k-1}Bu_0 + \ldots + Bu_{k-1} = \begin{bmatrix} B & \ldots & A^{k-1}B \end{bmatrix} \begin{bmatrix} u_{k-1} \\ \vdots \\ u_0 \end{bmatrix}; \end{aligned}$$

and consequently

$$x_k = \Delta \begin{bmatrix} u_{k-1} \\ \vdots \\ u_0 \end{bmatrix} \tag{1.12}$$

$\Delta = \begin{bmatrix} B & \dots & A^{k-1}B \end{bmatrix}$ matrix in equation 1.12 takes the name of "controllability matrix": if (1.13) holds

$$\operatorname{rank} \Delta = n \qquad \text{for } k \geqslant n \tag{1.13}$$

where n is the system order, it is possible to say that the system is controllable.

Stabilizability is a condition weaker than controllability. A system is stabilizable if the states (or more precisely the modes) that are not controllable are stable, and this means that the system remains stable during normal operations even if some of its states are not controllable.

Observability and detectability: observability measures the capacity of recovering the states sequence of the system starting from its external output data, that is how much information about the system can be recovered from outputs at maximum.
Considering the system in equation 1.14

$$\begin{aligned} x_{k+1} &= Ax_k \\ y_k &= Cx_k \end{aligned} \tag{1.14}$$

Starting from time 0 as previously done for controllability, it is possible to obtain

$$\begin{aligned} y_0 &= Cx_0 \\ y_1 &= CAx_0 \\ &\vdots \\ y_{k-1} &= CA^{k-1}x_0 \end{aligned}$$

that is the result in eq. 1.15 can be reached:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{k-1} \end{bmatrix} = \Gamma x_0 \tag{1.15}$$

where $\Gamma = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{k-1} \end{bmatrix}$ is known as observability matrix. If this matrix respect the condition in (1.16)

$$\operatorname{rank} \Gamma = n \qquad \text{for } k \geqslant n \tag{1.16}$$

the system is said to be observable. This is because, if at least n rows of Γ are independent, the n component of x_0 can be calculated through linear combinations of the output variables.

Such as in the case of stabilizability w.r.t controllability, detectability is a condition weaker than observability. A system is said to be detectable when, even if it is not completely observable, the unobservable states are asymptotically stable.

1.3 IDENTIFICATION: OBTAINING MODELS FROM DATA

Identification is one of the steps in the definition of a model-based control scheme such as MPC. It consist in finding a mathematical model of the process, starting from plant data. Clearly, the aim is to find a model which describes in the best possible way the behavior of plant outputs in response to an input sequence entering in the system.
Identification is not a simple task: it is composed by several steps, which are graphically represented using a block scheme in figure 6 (that can be found in Ljung[20]). The conceptual flow of this figure can be described with the following few points, where the name of the block that performs the mentioned action is reported in bracket.

- Some prior knowledge on the system gives the possibility of define a valid signal for data collection (Experiment Design), to select a model family that can approximate in the appropriate way the system (Choose model set) and to consider an adequate fitting criterion (Choose criterion of fit).
- Data are collected (Data).
- Data, model set and fitting criterion are used to calculate an appropriate set of parameters (Calculate model).
- The model that is obtained needs to be validated (Validate model) in order to check its performances. If this model satisfies the requisites it is adopted as process model, otherwise it is necessary to check the previous steps and/or to make better starting choices.

In this section, experiment design step in particular is reported. Topics regarding identification step will be reported in the next section.

1.3.1 Experiment design

Experiment design is a basilar step in process identification, even only few works were presented on this topic (see chapter 4 for more information). This step consists in defining an input signal which can excite the process in the right way to obtain as more knowledge as possible about the system.
In this sense, an important point is that data used in identification should be "informative enough" in order to obtain an optimal model. This quality refers to the information content of data, and it means that using "informative enough" data it is possible to discriminate between two different models of the same models set. Furthermore, it is strictly correlated to another concept, that is the "persistently exciting" signal, because it can

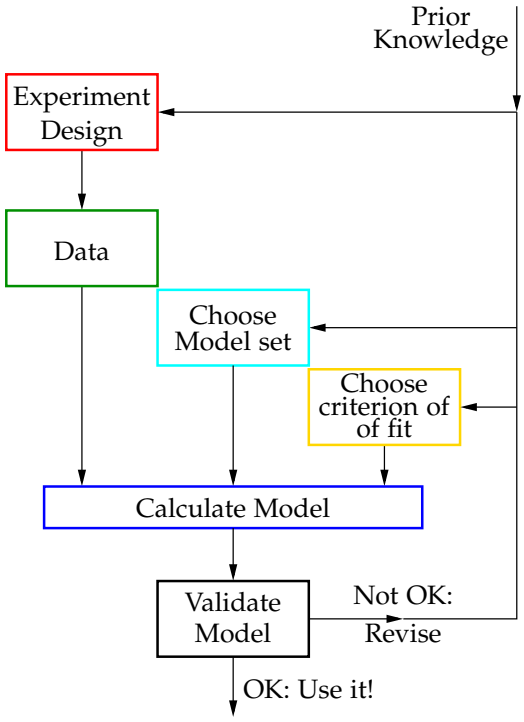

Figure 6: Description of the identification process

be showed that a persistently exciting input signal generates informative enough data for identification. For this particular kind of signal, the power spectrum Φ respects the following:

$$\Phi(\omega) > 0 \quad \text{for almost all } \omega \tag{1.17}$$

Power spectrum Φ, roughly speaking, is the function that describes the way the energy is distributed with frequency inside a signal. In this work, it is not important to analyze extensively these definitions and their analytical background, but the interested reader should refer to Ljung [20] for a deeper treatment.

From the definition of "persistently exciting signal", it results clear that the power content of the input signal at various frequencies plays a major role in experiment design. In particular, a low signal in some bands of frequency leads to outputs with poor information for the process in those bands. Then, a model obtained from this kind of data set could not work properly when subjected to signals in the same frequency regions.

In general, there are two approaches to experiment design, that is open loop (OL) inputs and closed loop (CL) inputs, which have different characteristics:

OL INPUTS : an input sequence is designed, and it is sent directly to the process. Such a structure is simpler and does not need any controller (such as the CL does), but needs the plant to stop normal operations because outputs usually do not respect quality limits;

CL INPUTS : in this case, a control scheme is present. Set point variations are sent to the system, and the controller provides to calculate the appropriate inputs to reach the new set-points. This kind of experiment is more complicated of an OL experiment for the necessity of a controller, but has the advantage of maintaining outputs at desired specifications, so it can be performed during normal plant operations.

So, it results better to use a CL data collection scheme when available. The principal reasons are two:

- As seen, CL data collection does not influence the production of the plant, because the controller maintains outputs within a given range around desired values. On the other hand, in case of an OL data collection, usually the products need to be reprocessed, wasting time and resources.
- Models obtained from CL data are often more appropriate to be used inside an MPC scheme, because the power spectrum of data is higher in the most important frequency regions for control.

These properties are consequence of the previous considerations that were made about frequency content of signals and data collection schemes.

Once the scheme of data collection has been selected, there are several kind of signals among which it is possible to choose the most useful for every single identification. The most common identification signals in industrial life are steps, but also binary signals such as GBN signals can be used.
In general, these are not the only available choices, but several different signals can be used, such as PRBS, sinusoidal, etc. Everyone of them has different characteristics in terms of frequency content, which could be useful in some applications. In this chapter, however, these latter are not described because they are not used through this work. It is recommended to refer to Ljung [20] in order to find more information about these types of inputs.
Step test and GBN signal are described below in order to define their characteristics:

STEP TEST : plotted in Figure 7. This is the simplest one and the most used in industrial life as previously said. It consist in changing the value of a single variable for a certain amount of time that permits to the system to go to steady state conditions, then in changing the variable back to the previous value. Often, after the first step, another one is made, with the same magnitude but opposite sign with respect to the initial condition, before driving the system to starting conditions again;

GBN : Generalized Binary Noise, a completely random binary signal, that is a signal that varies between two fixed values

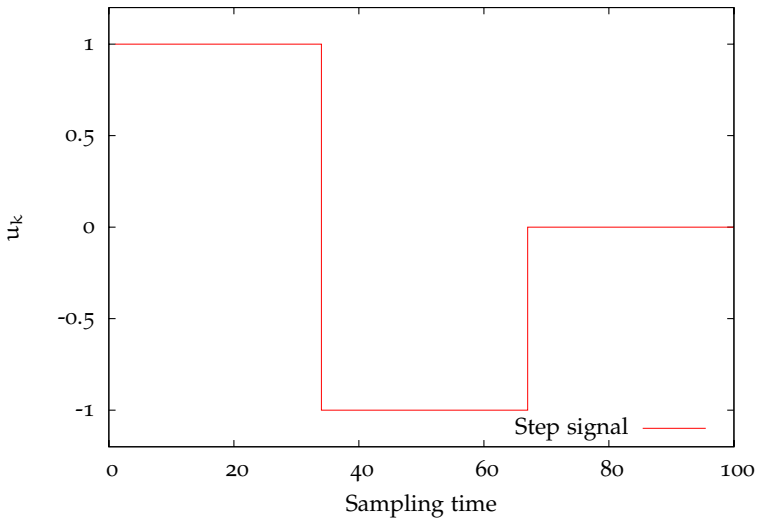

Figure 7: Step signal

(usually $\pm t$ with t as user-selected parameter) following a casual pattern. It is particularly suitable because it excites the system in a large band of frequencies, so the information coming from data is more complete. To build it, simply define a probability parameter $\delta \in \mathbb{N}$, and then respect the following rules

- for the first sampling time, generate a random number $0 < x \leqslant 1$, then

$$u_0 = \begin{cases} t & \text{if } 0 < x \leqslant 0.5 \\ -t & \text{if } 0.5 < x \leqslant 1 \end{cases} \tag{1.18}$$

- for the next sampling times, select a value for δ, and then

$$u_k = \begin{cases} -u_{k-1} & \text{if } 0 < x \leqslant \frac{1}{\delta} \\ u_{k-1} & \text{if } \frac{1}{\delta} < x \leqslant 1 \end{cases} \tag{1.19}$$

  As it is easy to understand, δ represent the mean switching time for u_k. Roughly speaking, δ sampling times are needed for the GBN signal to switch on average. So, the higher δ, the fewer number of switch for u_k. This is important because a low number of switches means a higher power at low frequencies and vice versa.

A standard GBN sequence is plotted in figure 8.

1.4 IDENTIFICATION TECHNIQUES

The block "Calculate the model" in Figure 6 has been not considered till now in this description, even if it is the central operation

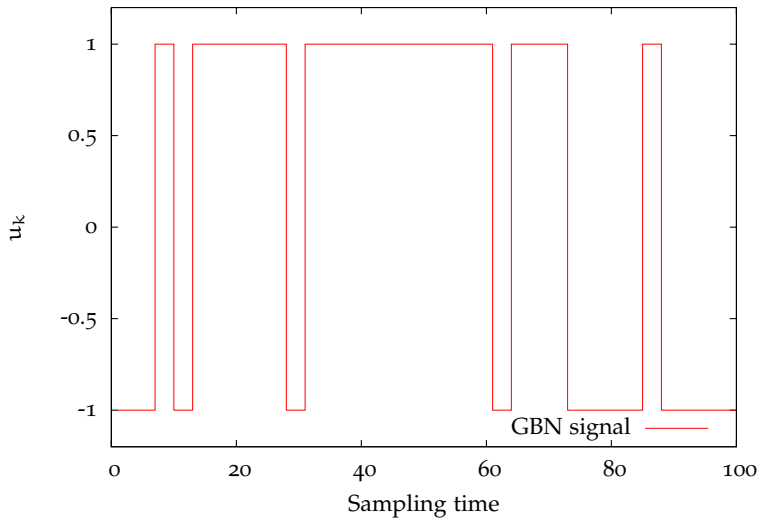

Figure 8: GBN signal

in identification task. In general, it is performed using two different approaches, namely PEM and SIM. In paragraphs 1.4.1 and 1.4.2, a brief description of the two is reported, showing the principal characteristics of both of them.

1.4.1 Introduction to PEM techniques

PEM techniques were the first methods to be developed for identification, because their underlying idea is very simple and it represents the most direct answer to the problem of finding the best possible model inside a set. Before describing this method, it is important to introduce the meaning of model set. A model set represents all the models with the same structure but different parameters:

$$y_k + \alpha_1 y_{k-1} = \beta_1 u_{k-1} + v_k \tag{1.20}$$

for example the (1.20) represents the family of all the ARX models with $n_a = 2$ and $n_b = 1$.
Actually, PEM methods find the best model as the one that, inside a family, gives outputs with the closest value to the real ones. To obtain a consistent estimation for models using PEM, two requirements are needed:

- a set of models in which the best one has to be chosen;
- a data fitting method .

Starting from the first point, it is possible to say that every model is function of several parameters: suppose to call the set of these parameters $\bar{\beta}$. So the best model will be the one associated to the $\bar{\beta}^*$ that minimizes the error in the sense of a certain criterion.

Analytically, given a minimizing function f, the operation performed is the following:

$$\overline{\beta}^* = \arg\min_{\overline{\beta}} f(\overline{\beta})$$

This action can be performed constructing a "predictor", that is an expression that permits to predict the values of y at time k using the previous occurrences of inputs and outputs. The predictor formulation depends on the adopted family of models, as it can be easily explained with the following example.
Suppose to consider an ARX model for a SISO system, with $n_a = 2$ and $n_b = 1$, as the one in equation 1.20. Rewrite this model in the following form:

$$y_k = -\alpha_1 y_{k-1} + \beta_1 u_{k-1} + \nu_k \tag{1.21}$$

It is straightforward that the best prediction of y_k which can be obtained with the deterministic values at time $k-1$ is given by

$$\hat{y}_{k,\mu} = -\alpha_1 y_{k-1} + \beta_1 u_{k-1} = \begin{bmatrix} y_{k-1} & u_{k-1} \end{bmatrix} \mu \tag{1.22}$$

where $\mu = \begin{bmatrix} -\alpha_1 \\ \beta_1 \end{bmatrix}$.
ν_k represents the stochastic part of the process, that is the one that cannot be described with deterministic equations. Now, suppose to minimize the difference between the real and the predicted value in order to obtain an estimation of μ, namely $\hat{\mu}$. A LS minimization method is used for data fitting, that is

$$\hat{\mu} = \arg\min_{\mu} (\mathbf{y} - \hat{\mathbf{y}}_\mu)'(\mathbf{y} - \hat{\mathbf{y}}_\mu) \tag{1.23}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \hat{\mathbf{y}}_\mu = \begin{bmatrix} \hat{y}_{1,\mu} \\ \hat{y}_{2,\mu} \\ \vdots \\ \hat{y}_{N,\mu} \end{bmatrix} \tag{1.24}$$

The solution of this problem is

$$\hat{\mu} = \hat{\mathbf{y}}_\mu^+ \mathbf{y} \tag{1.25}$$

where $\hat{\mathbf{y}}_\mu^+$ represents the pseudo-inverse of the matrix $\hat{\mathbf{y}}_\mu$. This operation gives an estimation of μ parameters and permits to define the model included in $\bar{\beta}$ family which approximates the data in the best way. It is clear even from this simple example that, if a model set that does not describe properly the system is selected, the result could be poor.

In figure 9, in order to show visually the meaning of minimizing the prediction error, two different models are compared with real data. As it is possible to see, model 1 is better in the sense of predicting the model behavior than model 2, because it is closer to the real data. The formulation of PEM is particularly suitable to build an iterative identification method. This gives better results in model recovering but, on the other hand, it has an heavy computational load, so the time required for the identification of a model increases.

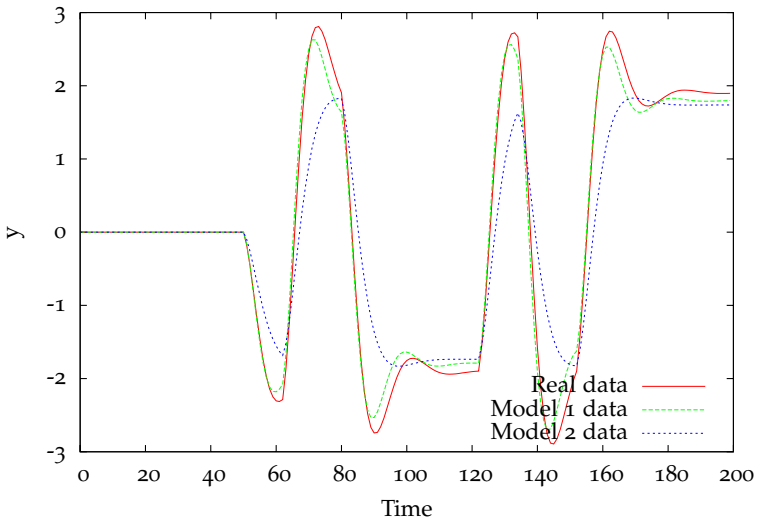

Figure 9: Data fitting example

1.4.2 Subspace identification methods

Subspace identification methods (SIM) are the second major family of identification methods. Their approach to the problem is completely different from the PEM one, and it is based mostly on a geometrical analysis of data.
The analysis performed by SIM can be called geometrical because they work using some projections of data matrices. The basic idea is that if a model describes a system in an appropriate way, the data it gives should lie on the same hyperplane on which real data lie. In particular, eliminating the contribution of noise from real data (which is an action that can be performed projecting data in a particular plane), all the deterministic information on the plant can be recovered.
In this chapter the meaning of projection is reported and the identification method used through this work is described. As it will be possible to see, it derives from the combination of two existent methods, that were selected for their satisfactory performances.

The meaning of projection

SID methods use a projection of data matrices in order to recover the system model, as suggested by their name. The concept of matrix projection derives directly from the geometrical meaning of this word, as it will be showed in this section.

First, a definition of a projection matrix is given:

Assumption 1. *A projection matrix* P *is a matrix that performs a projection of an entity (matrix, vector) belonging to a generic hyperspace* $W \in \mathbb{R}^{g \times Q}$ *onto a subspace* $U \in \mathbb{R}^{h \times Q}$ *of* W. U *is called range w.r.t* P, *while* Z *for which* $W = U \bigoplus Z$ *is called null space w.r.t.* P.

Projection matrix P must be idempotent, that is $P = P^2$. As a consequence of this, its eigenvalues can assume only the values of 0 and 1.
Direct sum $U \bigoplus Z$ means that every vector $w \in W$ can be decomposed uniquely as $w = u + z$ with $u \in U$ and $z \in Z$, that is the entire space W can be spanned using U and Z.

There are two different kind of projections, that is the oblique projection and the orthogonal projection. In figure 10 the first one is represented. In a plane $X = \{x_1, x_2\}$, vector $\vec{v}$ is multiplied by a matrix P to project it onto the subspace represented by the x_1 axis. This is done moving along the direction of x_2, and consequently vector $P \cdot \vec{v}$ is obtained.
In an analytic sense, given a vector $\vec{v} = \begin{bmatrix} v_1 & v_2 \end{bmatrix}'$ with $v_1 \in x_1$ and $v_2 \in x_2$, and considering $U = x_1$ and $Z = x_2$, matrix P that projects the column space of $\vec{v}$ onto its range U along the direction of Z is the one for which

$$P \cdot \vec{v} = \begin{bmatrix} v_1 \\ 0 \end{bmatrix}. \tag{1.26}$$

The projection of the row space is performed using a P matrix that post-multiplies the entity to be projected. Supposing $\vec{v} = \begin{bmatrix} v_1 & v_2 \end{bmatrix}$, it results that the matrix P which projects the row space of $\vec{v}$ onto its range U along Z is the one for which

$$\vec{v} \cdot P = \begin{bmatrix} v_1 & 0 \end{bmatrix}. \tag{1.27}$$

Orthogonal projection is a particular projection for which range

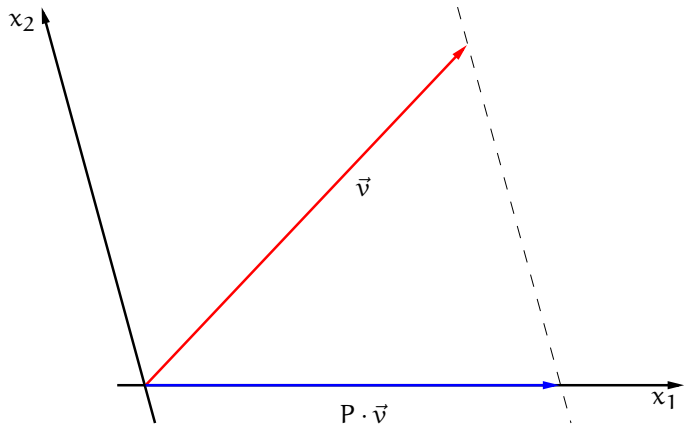

Figure 10: Oblique projection: graphical interpretation

U and null space V are orthogonal, such as in figure 11, where x_1 and x_2 form a right angle.

Analytically, it is possible to demonstrate that, given a matrix $X \in \mathbb{R}^{q \times Q}$ with $q \leqslant Q$, the matrix W_x that orthogonally projects every entity belonging to the space $\mathbb{R}^{Q \times Q}$ in the row space spanned by the rows of matrix X is given by

$$W_x = X'(X \cdot X')^{-1} X \tag{1.28}$$

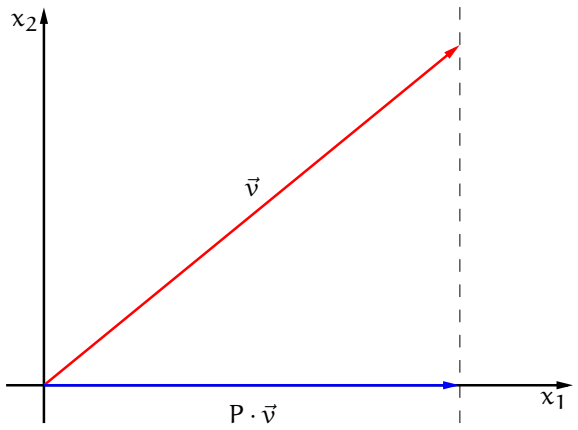

Figure 11: Orthogonal projection: graphical interpretation

A small example is reported: looking to figure 12, suppose to have the matrix $X = \begin{bmatrix} x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$: its row space spans, clearly, the space $\mathbb{X}$. Consider now the vector $x_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ in the same plot: it is independent from plane $\mathbb{X}$, so its projection in this plane is expected to be 0. Constructing W_x as in the (1.28), it is possible to obtain

$$\begin{aligned} W_x &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}' \left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}' \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{1.29}$$

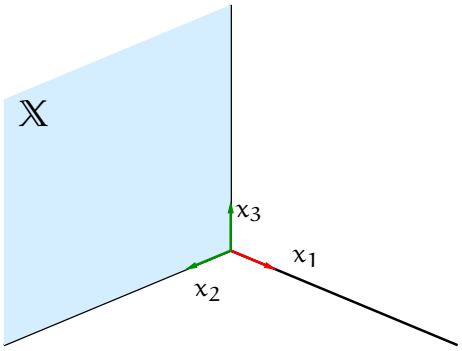

Figure 12: Orthogonal projection: numeric example

So, the projection of x_1 on $\mathbb{X}$ is

$$x_1/\mathbb{X} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{1.30}$$

as it was expected.

Basic definitions for SID identification methods

SID identification methods have a quite complicated theoretical background, as previously stated: their formulation needs some basic concepts and assumptions, that will be reported in this section. At first, it is important to define the kind of model that will be adopted; linear discrete time-invariant state-space system in the following form will be considered:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Kv_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{1.31}$$

in which $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^p$ is the output, $v \in \mathbb{R}^p$ is stochastic noise, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ are the system matrices and $K \in \mathbb{R}^{n\times p}$ is the noise model matrix. Then, once the model is defined, it is important to introduce some assumptions:

Assumption 2. *The pair* (A, B) *and* (A, K) *are stabilizable, the pair* (A, C) *is observable, the closed-loop matrix* $(A - KC)$ *is strictly Hurwitz (in a discrete-time sense), the noise* v_k *is white, and statistically independent of past outputs and inputs, i.e.*$E\{y_k v_j'\} = 0$ *and* $E\{u_k v_j'\} = 0$ *for all* $j > k$.

Given a positive integer r, assumed to satisfy $r > n$, let the vectors of "future" (with respect to time k) outputs, inputs and noises be defined, respectively, as:

$$\mathbf{y}_k = \begin{bmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_{k+r-1} \end{bmatrix}, \quad \mathbf{u}_k = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+r-1} \end{bmatrix}, \quad \mathbf{v}_k = \begin{bmatrix} v_k \\ v_{k+1} \\ \vdots \\ v_{k+r-1} \end{bmatrix} \tag{1.32}$$

To understand the meaning of indexes that will be used in this analysis, consider the following assumption:

Assumption 3. *Data vectors* (u, y) *are collected for* L *sampling times, namely from sample time* 0 *to sample time* $L - 1$ *(with* $L \gg r$*).*

Starting from the model (1.31), one can obtain from the states equation with few substitutions:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Kv_k \\ x_{k+2} &= A^2 x_k + ABu_k + Bu_{k+1} + AKv_k + Kv_{k+1} \\ &\vdots \\ x_{k+r-1} &= A^{r-1}x_k + \begin{bmatrix} A^{r-2}B & \dots & B \end{bmatrix} \begin{bmatrix} u_k \\ \vdots \\ u_{k+r-2} \end{bmatrix} + \\ &\quad + \begin{bmatrix} A^{r-2}K & \dots & K \end{bmatrix} v_k \begin{bmatrix} v_k \\ \vdots \\ v_{k+r-2} \end{bmatrix} \end{aligned}$$

Stacking the equations, it results

$$\begin{bmatrix} x_k \\ x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+r-1} \end{bmatrix} = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^{r-} \end{bmatrix} x_k + \\ + \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ B & 0 & \cdots & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A^{r-2}B & \cdots & AB & B & 0 \end{bmatrix} \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+r-1} \end{bmatrix} + \\ + \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ K & 0 & \cdots & \cdots & 0 \\ AK & K & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ A^{r-2}K & \cdots & AK & K & 0 \end{bmatrix} \begin{bmatrix} v_k \\ v_{k+1} \\ v_{k+2} \\ \vdots \\ v_{k+r-1} \end{bmatrix} \tag{1.33}$$

in which 0 and I are used to denote the full zero matrix and the identity matrix, respectively, of suitable dimensions. From the outputs equation in (1.31):

$$\begin{aligned} y_k &= Cx_k + v_k \\ y_{k+1} &= Cx_{k+1} + v_{k+1} \\ &\vdots \\ y_{k+r-1} &= Cx_{k+r-1} + v_{k+r-1} \end{aligned}$$

It easy to reach the (1.34) by stacking the equation from y_k to y_{k+r-1}:

$$\mathbf{y}_k = \begin{bmatrix} C \\ C \\ C \\ \vdots \\ C \end{bmatrix} \begin{bmatrix} x_k \\ x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+r-1} \end{bmatrix} + \boldsymbol{\nu}_k \tag{1.34}$$

Now, substituting (1.33) in (1.34)

$$\mathbf{y}_k = \Gamma_r x_k + H_r^u \mathbf{u}_k + H_r^v \boldsymbol{\nu}_k \tag{1.35}$$

in which Γ_r is named extended observability matrix (see section 1.2.4) , while H_r^u and H_r^v are known as lower block-triangular Toeplitz matrices. It is straightforward that, for the Γ_r matrix

$$\Gamma_r = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{r-1} \end{bmatrix} \tag{1.36}$$

while for the H_r^u and the H_r^y ones

$$\begin{aligned} H_r^u &= \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ CB & 0 & \cdots & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ CA^{r-2}B & \cdots & \cdots & CB & 0 \end{bmatrix} \\ H_r^y &= \begin{bmatrix} I & 0 & \cdots & \cdots & 0 \\ CK & I & \cdots & \cdots & 0 \\ CAK & CK & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ CA^{r-2}K & \cdots & \cdots & CK & I \end{bmatrix} \end{aligned} \tag{1.37}$$

Being the data collected for L sampling times as it results from assumption 3, it is possible to write:

$$Y_f = \Gamma_r X + H_r^u U_f + H_r^y V_f \tag{1.38}$$

where the matrices Y_f, X, U_f, V_f are constructed by placing side-by-side the vectors $\mathbf{y}_k$, x_k, $\mathbf{u}_k$ and $\boldsymbol{\nu}_k$ respectively, i. e.

$$\begin{aligned} Y_f &= \begin{bmatrix} \mathbf{y}_r & \mathbf{y}_{r+1} & \cdots & \mathbf{y}_{r+M-1} \end{bmatrix} \\ X &= \begin{bmatrix} x_r & x_{r+1} & \cdots & x_{r+M-1} \end{bmatrix} \\ U_f &= \begin{bmatrix} \mathbf{u}_r & \mathbf{u}_{r+1} & \cdots & \mathbf{u}_{r+M-1} \end{bmatrix} \\ V_f &= \begin{bmatrix} \boldsymbol{\nu}_r & \boldsymbol{\nu}_{r+1} & \cdots & \boldsymbol{\nu}_{r+M-1} \end{bmatrix} \end{aligned} \tag{1.39}$$

In the equations 1.39 the number of columns of the matrices is equal to $M = L - 2r + 1$.
In some subspace identification algorithms, as the one considered in this work, it is common to introduce the matrix $Z_f \in \mathbb{R}^{(pr+mr)\times M}$, which is obtained by stacking Y_f and U_f, that is

$$Z_f = \begin{bmatrix} Y_f \\ U_f \end{bmatrix} \tag{1.40}$$

Similarly, the matrix

$$Z_p = \begin{bmatrix} Y_p \\ U_p \end{bmatrix} \tag{1.41}$$

can be defined using the matrices of "past"(w.r.t. sampling time k) output and input data, i. e.Y_p and U_p

$$\begin{aligned} Y_p &= \begin{bmatrix} \mathbf{y}_0 & \mathbf{y}_1 & \cdots & \mathbf{y}_{M-1} \end{bmatrix} \\ U_p &= \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_{M-1} \end{bmatrix} \end{aligned} \tag{1.42}$$

Projection for obtaining system matrices

Almost the totality of subspace methods perform a projection of output (and input) data onto a subspace orthogonal to noise: in this way, the projected data matrices depend only on deterministic contributions, while all the stochastic information contained in the original data is discarded.

The main difference between the various methods that can be analyzed relies in the matrices used to perform these projections, and some difference can be found also in the way the model matrices are obtained from Γ_r, H_r^u and H_r^y.

In this work, the method of orthogonal projections proposed by Huang et al. [15] is used, introducing some modifications as detailed later in this section.
The adopted algorithm is particularly interesting because it overcomes the main problem of many subspace methods, that is the lack of consistency when used with Closed-Loop (CL) data. This is because the projection performed by most subspace algorithms does not respect assumption 2 in case of a CL system. This is because it is common for subspace algorithms to perform the following operations.

- X matrix is expressed as a linear regression on the past input and output data, that is

$$X \cong \Theta_X Z_p. \tag{1.43}$$

- Substituting the previous equation in the (1.38), it results

$$Y_f = \begin{bmatrix} \Gamma_r \Theta_X & H_r^u \end{bmatrix} \begin{bmatrix} Z_p \\ U_f \end{bmatrix} + H_r^v V_f. \tag{1.44}$$

- a projection in the space row space of $\begin{bmatrix} Z_p \\ U_f \end{bmatrix}$, namely W, is performed

$$Y_f/W = \begin{bmatrix} \Gamma_r \Theta_X & H_r^u \end{bmatrix} \begin{bmatrix} Z_p \\ U_f \end{bmatrix} /W + H_r^v V_f/W. \tag{1.45}$$

If assumption 2 holds and data are collected using an OL scheme, $V_f/W = 0$, because v lies on an orthogonal plane w.r.t. u and y. If these conditions are not respected it is not possible to remove the noise term with this projection, so the estimated model is inconsistent. OL scheme for data collection is necessary, otherwise $V_f/W \neq 0$ because future inputs U_f would depend on V_f values.
For this reason, a method which uses a different approach to the problem is considered in this work. The underlying philosophy of that method is to recover the matrix Γ_r from the (left) orthogonal space of the matrix Z, defined as:

$$Z = Z_f W \tag{1.46}$$

where two choices of the orthogonal projection matrix $W = W' \in \mathbb{R}^{M \times M}$ are considered, depending on the data collection scheme. If data are collected in open loop, the appropriate choice is $W = Z_p^+ Z_p$, where the superscript $^+$ denotes the right pseudo-inverse operator (computed via Singular Value Decomposition, SVD). Huang et al. [15] show that this choice leads to an orthogonal projection of Z_f onto the row space of Z_p. If data

are collected in closed loop, instead, the appropriate choice is $W = Z_{CL}^{+} Z_{CL}$, where $Z_{CL} = \begin{bmatrix} \bar{Y}_f' & Z_p' \end{bmatrix}'$ and $\bar{Y}_f \in \mathbb{R}^{pr \times M}$ is the matrix of future set-points, defined similarly to Y_f [15].
This modification is needed because, in case of a CL scheme, a projection with Y_p leads to the recovering of a matrix that contains both the model of the process and the model of the controller, without possibilities of discerning between them. Introducing this new matrix Z_{CL} this problem can be avoided, because the contribution of the controller model is thrown away: for further information on this points Huang et al. [15, Sec.3.2] is suggested.

Having defined the projected data matrix Z as in the (1.46), the first step is to perform an SVD:

$$Z = U_Z S_Z V_Z' = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \end{bmatrix} V_Z' \tag{1.47}$$

where S_1 and S_2 are diagonal matrices, which contain the significant and the negligible singular values of Z, respectively. The dimension of S_1 can be obtained by using e. g.an Akaike Information Criterion[1] in [38] or a heuristic Principal Component Analysis (PCA) approach as described.
The first mr singular values of Z are considered, and then subsequent $\hat{n}$ significant singular values are selected according to:

$$\frac{\sigma_{mr+\hat{n}}}{\sum_{j=1}^{\hat{n}} \sigma_{mr+j}} > \rho, \qquad \sigma_j \in \operatorname{diag} S_Z, \qquad 1 \leqslant \hat{n} \leqslant r \tag{1.48}$$

in which ρ is a positive scalar close to 0 (typical values for ρ are between 0.01 and 0.05). Thus, the diagonal matrix S_1, containing the significant singular values, has dimension $mr + \tilde{n}$, where $\tilde{n}$ is considered equal to the largest value of $\hat{n}$ for which (1.48) holds. Notice that since the left singular vectors matrix of Z, i.e. U_Z, has dimension $mr + pr$, the matrix U_2 in (1.47) has $pr - \tilde{n}$ columns. Huang et al. [15] show that:

$$\Gamma_r^{\perp} \begin{bmatrix} I & -H_r^u \end{bmatrix} = T U_2' \tag{1.49}$$

where $\Gamma_r^{\perp}$ is a basis matrix for the left null space of Γ_r, i.e. a full rank matrix such that $(\Gamma_r^{\perp})\Gamma_r = 0$ and T is any nonsingular transformation matrix of suitable dimensions (often chosen as the identity matrix). Equation (1.49) and a suitable partitioning:

$$T U_2' = \begin{bmatrix} P_1' & P_2' \end{bmatrix} \tag{1.50}$$

1 Akaike Information Criterion (AIC) is based on the identification of several models with increasing order, and on the comparison of their capacity of predicting the future values of y with the number of parameters they have.
The reason for this is that AIC tries to prevent the estimation of a too high order due to the presence in data of fictitious poles given by noise. So, for AIC, the best model is the one for which a suitable function $f(\epsilon, n_{par})$ (where $\epsilon = y - \hat{y}$ represents the difference between real values and prediction for a model and n_{par} represents the number of parameters of the same model) obtains the minimum value.

in which P_1' has pr columns, allow one to compute Γ_r and H_r^u from the following relations:

$$P_1'\Gamma_r = 0 \tag{1.51a}$$
$$-P_1'H_r^u = P_2' \tag{1.51b}$$

It is clear that Γ_r is computed from (1.51a) as an orthonormal basis matrix of the (right) null space of P_1', whereas (1.51b) is solved for H_r^u in a least-squares sense. Notice that the computation of H_r^u in a least-squares sense can be improved by taking into account the Toeplitz structure of this matrix, reported in (1.37).

Obtaining A and C matrices

The matrices A and C are computed from Γ_r: this is done using the shift invariance property of the observability matrix. Looking to Γ_r, it is possible to notice that

$$\underline{\Gamma}_r = \overline{\Gamma}_r A \tag{1.52}$$

with

$$\overline{\Gamma}_r = \Gamma_r(1:(r-1)p,:) \qquad \underline{\Gamma}_r = \Gamma_r(p+1:rp,:) \tag{1.53}$$

using a MATLAB notation. So, it is easy to obtain C and A

$$\hat{C} = \Gamma_r(1:p,:) \tag{1.54a}$$
$$\overline{\Gamma}_r A = \underline{\Gamma}_r(p+1:rp,:) \tag{1.54b}$$

where (1.54b) is solved for A in a least-squares sense, that is

$$\hat{A} = \overline{\Gamma}_r^{+}\underline{\Gamma}_r. \tag{1.55}$$

Obtaining B matrix

In this work, computation of the matrix B is changed with respect to the original algorithm in Huang et al. [15], because several studies pointed out poor results of this step in a number of cases [27, 31]. In particular, Huang et al. [15, Sec.3.1] recovered the matrix B from H_r^u, referring to [38] for this step. In this approach, H_r^u is obtained from equation 1.51b and then B is recovered using the particular structure of matrix H_r^u reported in equation 1.37.

A different approach to recover B was proposed by Qin et al. [31] and independently by Pannocchia et al. [27], which is very similar to that used in [20]. Specifically, it consists in obtaining that matrix by using a prediction error approach and solving a least-squares problem.
In these works, first, $\hat{y}_k$ is computed as

$$\hat{y}_{k|B} = C\sum_{j=0}^{k-1} A^j B u_j = f_k(\mathrm{Vec}\ B) \tag{1.56}$$

Being (1.56) linear in B, by differentiating w.r.t. B terms and reorganizing matrices in an adequate way, it results

$$\hat{y}_{k|B} = \varphi_k \text{Vec } B \tag{1.57}$$

where Vec B is a vector obtained by stacking each column of B on top of the next one, and $\varphi_k \in \mathbb{R}^{p \times n(m+1)}$ is the Jacobian matrix of f_k, that can be easily computed from A, C and the known sequence of inputs. Stacking the various occurrences of y_k on the top of each other, eq. 1.58 can be obtained

$$\underbrace{\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_L \end{bmatrix}}_{\mathcal{Y}} = \underbrace{\begin{bmatrix} \varphi_0 \\ \varphi_1 \\ \vdots \\ \varphi_L \end{bmatrix}}_{\Phi} \text{Vec } B \tag{1.58}$$

The determination of $\hat{B}$, that is the estimate of B, now is straightforward: using a least squares approximation

$$\hat{B} = \Phi^{+} \mathcal{Y} \tag{1.59}$$

This second approach gives consistent estimations for B, and it is the one used in identification in this work when it is not differently specified.

Finally, if required, the disturbance model matrix K can be obtained in several ways. Huang et al. [15, Sec.3.3] propose a formulation based on Kalman filter states computed from Γ_r and H_r^u, which is applicable to open-loop data only. They also discuss an alternative formulation applicable to closed-loop data, which in turn computes B, C and K in a slightly different way.

2 MPC SCHEME: STANDARD FORMULATION AND ASSUMPTIONS

Contents

2.1 MPC in the plant 30
2.2 Internal structure of the MPC 31
2.2.1 The observer 33
2.2.2 The steady-state optimization 34
2.2.3 The dynamic optimization 36
2.3 Some final points on MPC 39
2.3.1 Disturbance model choice 39
2.3.2 MPC tuning 41

Interest on process optimization has been growth in industrial world in the last years, as previously remarked. Several methods were proposed or investigated in order to improve the results for optimization, but the attention rapidly focused on MPC scheme, which had a very fast development due to its high flexibility and performances.
At the moment, MPC technology can be considered mature, because its basic formulation is widely accepted and it seems that no large modification on the algorithm could be introduced. For this reason, everyone of the different MPC algorithms in commerce follows the same guidelines, even if several differences can be found between the various formulations.

MPC acronym stands for **M**odel **P**redictive **C**ontroller, and these words describe well its principal features:

MODEL : MPC is based on a model of the process. This can be a multivariable model, because MPC scheme can handle a large number of variables at the same time, taking into account the interactions and dependencies that can be found in the whole plant;

PREDICTIVE : MPC uses the previously introduced model of the process to predict the plant response to inputs, in order to find the best working point to minimize costs and to respect specifications.

An appoint can be done on the last word, controller. This definition is not totally right, because MPC is not a controller in the classic meaning of that word, but it would be better to define it "optimizer". Indeed, it does not control directly any process variable, but it uses information coming from the plant just to change the set-point of base controllers (P, PI, PID) of the process in order to optimize the system.

At the moment, there are thousands of plants which use MPC scheme, and their number is rapidly increasing: after a first period, in which MPC was mainly developed in industry, it started to be investigated also in the academic world, which provided the theoretical bases to its structure.

MPC design is simple in theory: one has only to build some suitable cost functions for the plant, to consider which constraints are active in the process, such as physical constraints or quality constraints, and to minimize those functions respecting these constraints, in order to make the costs for plant running as small as possible.

As previously stated, the various formulations of MPC which can be found in commerce have some minor differences between themselves. In general, these can be related mainly to some features in the analytical expressions for cost functions and constraints. In this chapter, MPC structure will be analyzed from a closer point of view, its component will be introduced and the way they operate will be explained.

This chapter is organizes as follows:

- in section 2.1, the hierarchy in a control system is analyzed;
- in section 2.2 the internal structure of an MPC algorithm is presented, and the various modules it is composed of are characterized;
- in section 2.3 some aspects in the definition of an MPC scheme are introduced, such as disturbance model selection and controller tuning.

2.1 MPC IN THE PLANT

In the previous paragraph it was pointed out that MPC stands over the standard controllers such as PI and PID and steer them. Actually, in an optimization scheme, there is a hierarchy between the various parts it is composed of.

In figure 13, indeed, the principal levels of this hierarchy are reported. Climbing the pyramid from the bottom, an higher level represent a more general view of the plant: on the lowest, there are actuators, which actually perform the control action on the plant, then base controllers which are the level that directly drives actuators. After the base controllers there is MPC, which coordinates the action of PID to get the specifications coming from the real-time optimization level; at least, there is a Planning & Scheduling level, that gives the guidelines for the entire plant.

The arrow on the left shows how sampling time of those levels becomes faster going down the pyramid, starting from the approximately weekly execution for the planning & scheduling step to the continuous time of actuators.

In figure 14, a process scheme with an MPC is reported, and in table 1 the correspondent CVs and MVs are indicated. As it

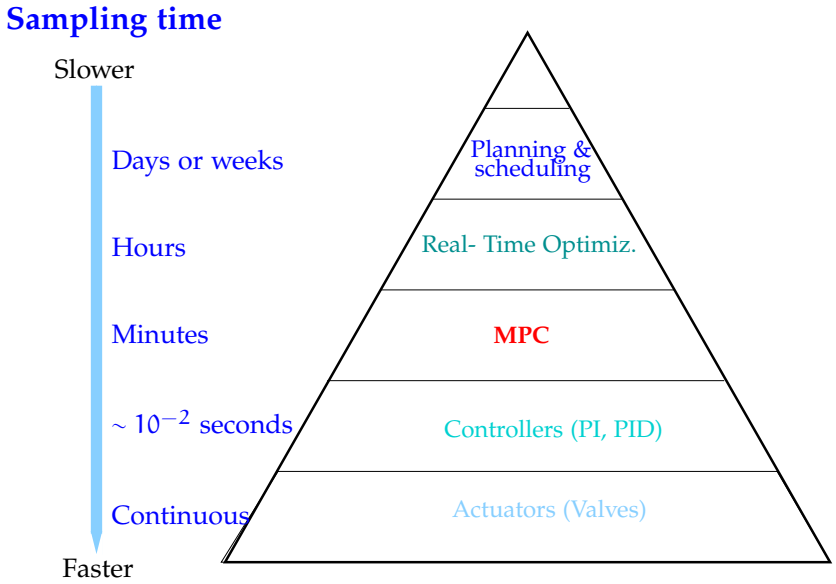

Figure 13: Hierarchy in a plant control system

is possible to see, MPC reads some important values from the plant: x_d and x_b, that is distillate and bottom purity which must be controlled to respect quality specifications, VLV's opening percentage, that needs to be controlled to avoid

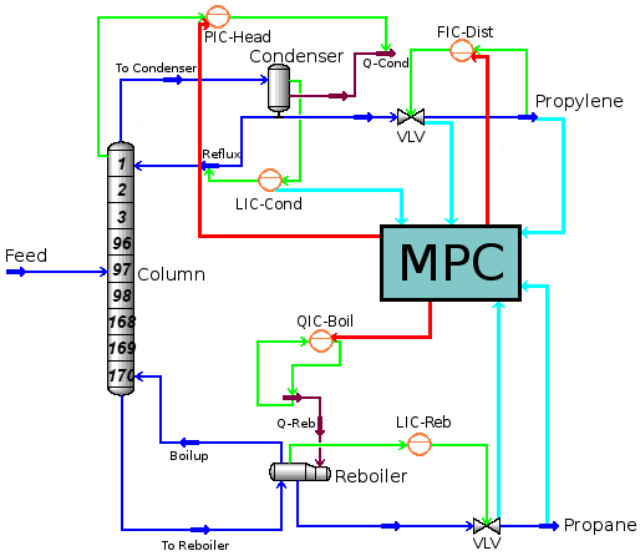

Figure 14: MPC position in the scheme of a plant

2.2 INTERNAL STRUCTURE OF THE MPC

In everyday activities it is common to consider MPC as a single block, but it is composed by three subsections with well-separated tasks:

Table 1: CVs & MVs for the scheme in figure 14

CV	MV
x_d	Distillate rate
x_b	Reboiler duty
VLV-1 opening percentage	Head Pressure
VLV-2 opening percentage	
Condenser liquid level	

- the observer (or filter);
- the steady state module;
- the dynamic module.

Everyone of them has its own objective, and they work in sequence starting from the observer and ending with the dynamic module.

In figure 15 the internal structure of MPC is reported: at every sampling time k, MPC reads the value of the outputs of the process, check these with its predictions at the previous sampling time and correct the values of these predictions. Then it uses those corrected values to calculate an optimal control sequence of several steps and sends the first step of this sequence to the controller.

Prediction is made using a model of the controlled process: various kind of models can be adopted but in academic research the preferred one is the state space model, which was introduced in section 1.2.4.

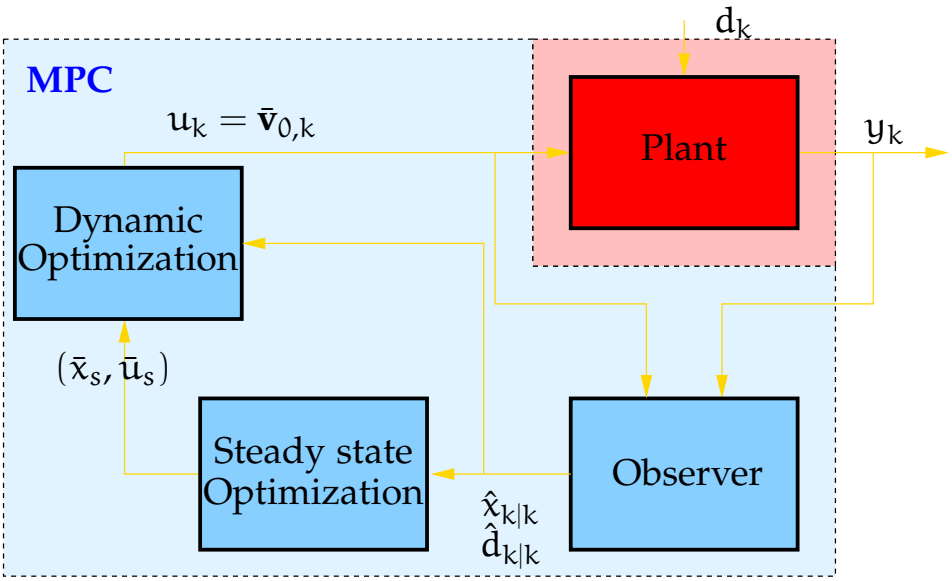

Figure 15: MPC internal structure

2.2.1 The observer

The first module that performs an action in the MPC algorithm is the observer: it has a basilar task, that is the updating of states values predicted at the previous sampling time using the present value of plant outputs.
In figure 16 it is possible to see that it is performed in two steps, the first one named filtering and the second one named prediction, at every sampling time.

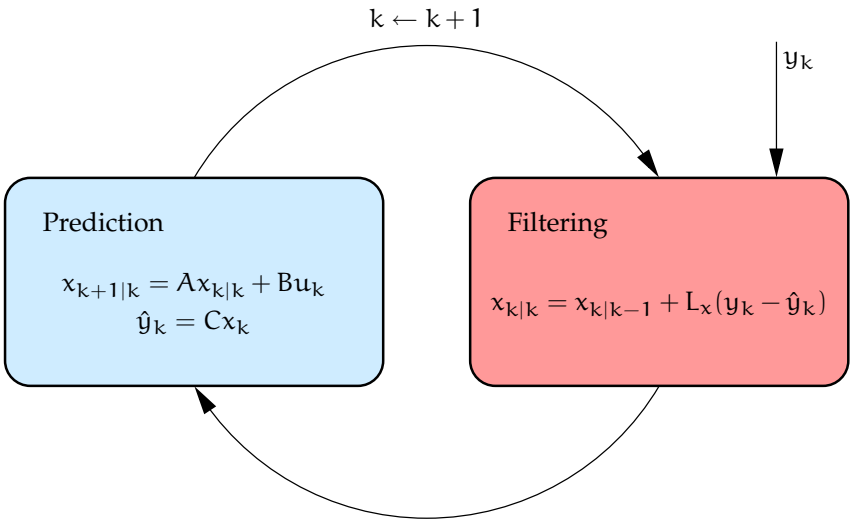

Figure 16: Graphical representation of the observer module

This figure has the following interpretation: at sample time k, the value y_k is taken from the plant. The difference between this value and the $\hat{y_k}$ value is used to correct the prediction of the states vector $x_{k|k-1}$ through the filter L_x. The new value $x_{k|k}$ is then used to predict $x_{k+1|k}$.
Variables are indicated using a double subscript, like $k|k$: the left counter is used to indicate the sampling time at which values are predicted, while the right counter indicates the sampling time at which they are corrected.

Recalling eq.1.10, it is possible to notice that, inside an MPC algorithm, the process model is not only characterized by a state vector, but it comes also with an integral disturbances vector. In that case, the equations of prediction step become (eq.2.1)

$$\begin{aligned} x_{k+1|k} &= Ax_{k|k} + Bu_k + B_d d_{k|k} \\ d_{k+1|k} &= d_{k|k} \\ \hat{y}_k &= Cx_{k|k} + C_d d_{k|k} \end{aligned} \tag{2.1}$$

while the filtering step expression is reported in (2.2)

$$\begin{aligned} x_{k|k} &= x_{k|k-1} + L_x(y_k - \hat{y_k}) \\ d_{k|k} &= d_{k|k-1} + L_d(y_k - \hat{y_k}) \end{aligned} \tag{2.2}$$

In this case, L_x and L_d are the filter of the process, and usually they are considered as part of the disturbance model.

2.2.2 The steady-state optimization

The aim of this module is to calculate the optimal values of input and states required at steady-state. The optimal values are those for which costs are minimized and constraints are satisfied, respecting the quality specifications required for the plant.
In figure 17 a graphical representation of the action performed by the steady state module is reported.

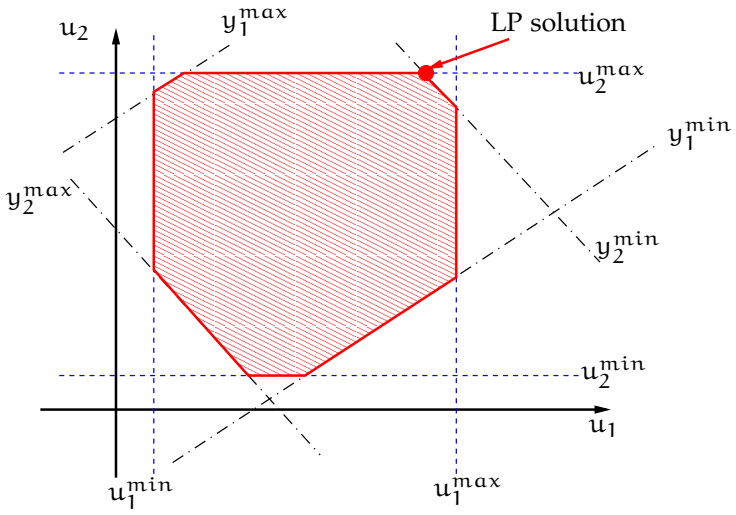

Figure 17: Graphical representation of the steady state module

Figure 17 has the following meaning: suppose to have a 2×2 system (that is a system with 2 inputs and 2 outputs). In the $u_1 - u_2$ plain represent the values of constraints: the first and the second input will be included in a band between their (respective) maximum the minimum admissible value, that is $u_1^{max} - u_1^{min}$ and $u_2^{max} - u_2^{min}$, here reported with the dashed lines.
Then, plot also the values of constraints for outputs y_1 and y_2: since MPC is linear, those values will be straight lines; here,they are represented with dot-dashed lines.
At this point, the whole $u_1 - u_2$ plain, in which the solution of the minimization problem has to be found, is restricted to an admissibility area internal to the constraints, represented by the shaded area in figure. It is possible to demonstrate that, using a Linear Programming (LP) problem, the solution of the minimization problem lies along the perimeter of that area, and in particular it corresponds to one of the vertexes. So, it is located where two of the lines representing constraints meet themselves. In 17 one of these points was arbitrary selected as solution and labeled as "LP solution", even if this plot has only a qualitative meaning.

The SS module action has been analyzed in a qualitative way till now, but it must be translate in an analytical expression to be implemented.
As it was pointed out, the problem to be solved is a minimizing

problem under several constraints: a general expression for this kind of problem is reported in (2.3).

$$(x_s, u_s) = \arg \min_{x_s, u_s, \bar{\epsilon}, \underline{\epsilon}} f(x_s, u_s). \tag{2.3}$$

$f(x_s, u_s)$ has not a unique form, but it can be expressed using a LP problem, as discussed before, or a QP (Quadratic Programming) problem: in table 2 the different expression of the two are reported.

Table 2: LP and QP expression for the steady state module

	LP	QP
Costs	$u_s' r$	$u_s' R u_s$
Penalization	$\bar{\epsilon}' \bar{q} + \underline{\epsilon}' \underline{q}$	$\bar{\epsilon}' \bar{Q} \bar{\epsilon} + \underline{\epsilon}' \underline{Q} \underline{\epsilon}$
$f(u_s, x_s)$	$\bar{\epsilon}' \bar{q} + \underline{\epsilon}' \underline{q} + u_s' r$	$\bar{\epsilon}' \bar{Q} \bar{\epsilon} + \underline{\epsilon}' \underline{Q} \underline{\epsilon} + u_s' R u_s$

This two expressions have some differences: LP formulation is simpler and requires less calculation time to be performed; on the other hand, QP is slower but its control action is usually smoother, preserving the actuators and the process in general from too large input variations. As reported in table 2 both of these functions are composed of two part: the first one takes into account the costs for the plant, considering the costs generated by inputs, while the second one is a penalization part that gives a penalty to the function value in case that soft constraints are not respected. Parameters $r, \bar{q}, \underline{q}$ in LP and $R, \bar{Q}, \underline{Q}$ in QP are tuning parameters, used to modify the priority of variables.

Now, let's introduce the expressions for constraints: for everyone of them, the meaning is shortly reported.

Steady-state constraint: the process is at steady-state, so the predicted states is equal to the state at the present time

$$x_s = A x_s + B u_s + B_d d_{k|k}; \tag{2.4}$$

hard constraints for inputs: the value of u vector can never break these limits

$$u^{min} \leqslant u_s \leqslant u^{max}; \tag{2.5}$$

soft constraints for outputs: outputs can go over these limits, but a strong penalization on minimization function arises:

$$\begin{gathered} y^{min} - \underline{\epsilon} \leqslant C x_s + C_d d_{k|k} \leqslant y^{max} + \bar{\epsilon}; \\ \underline{\epsilon} \geqslant 0; \\ \bar{\epsilon} \geqslant 0; \end{gathered} \tag{2.6}$$

set-points constraint: at steady state, the outputs must be at set-point value y_c. This statement can assume two different expressions:

- the first one calculates directly the value of the set point

$$y_c = H_y C x_s + H_y C_d d_{k|k}; \tag{2.7}$$

- the second one considers the set-point as hard constraints on the CVs

$$y_c \leqslant H_y C x_s + H_y C_d d_{k|k} \leqslant y_c. \tag{2.8}$$

In conclusion, putting everything together, the minimization problem for the steady state module becomes

$$\begin{aligned}
(x_s, u_s) = \arg \min_{x_s, u_s, \bar{\epsilon}, \underline{\epsilon}} & f(u_s, x_s); \\
\text{subject to} & \\
x_s &= A x_s + B u_s + B_d d_{k|k}; \\
y_c &= H_y C x_s + H_y C_d d_{k|k}; \\
u^{min} &\leqslant u_s \leqslant u^{max} \\
y^{min} - \underline{\epsilon} &\leqslant C x_s + C_d d_{k|k} \leqslant y^{max} + \bar{\epsilon}; \\
\underline{\epsilon} &\geqslant 0; \\
\bar{\epsilon} &\geqslant 0;
\end{aligned} \tag{2.9}$$

where the first formulation for set-point constraints was considered.

2.2.3 The dynamic optimization

The last module that composes the MPC structure is the dynamic module. This is the one that calculates the actual control action sent to the plant, and the way it works can be described in a qualitative way using figure 18.
In that figure it is possible to notice that dynamic module predicts the future outputs, from the present sampling time to a prediction horizon H_p, as the consequence of a future inputs sequence. This input sequence has an horizon too, named control horizon H_c: dynamic module calculates optimal control action from k to $k + H_c$, then it considers that inputs assume the steady state value u_s calculated by SS module for sampling times after $k + H_c$.

So, it has been cleared that dynamic module calculates the control action from k to $k + H_c$ for which the output would reach as soon as possible the value of set-point (the y_c line).
Looking at figure 18, it results that this is equivalent to minimizing the shaded area in the upper part of the figure. Indeed, this area represents the gap between real output and set-point, which should be clearly the smallest possible. The part of the output line over the soft constraint in future y sequence represents a soft constraint violation, which introduces for the involved sampling times an extra penalty on minimizing function value.

As previously done for the SS module, for the dynamic module also is possible to write the problem both in a LP representation and in a QP representation, but in general for this kind of

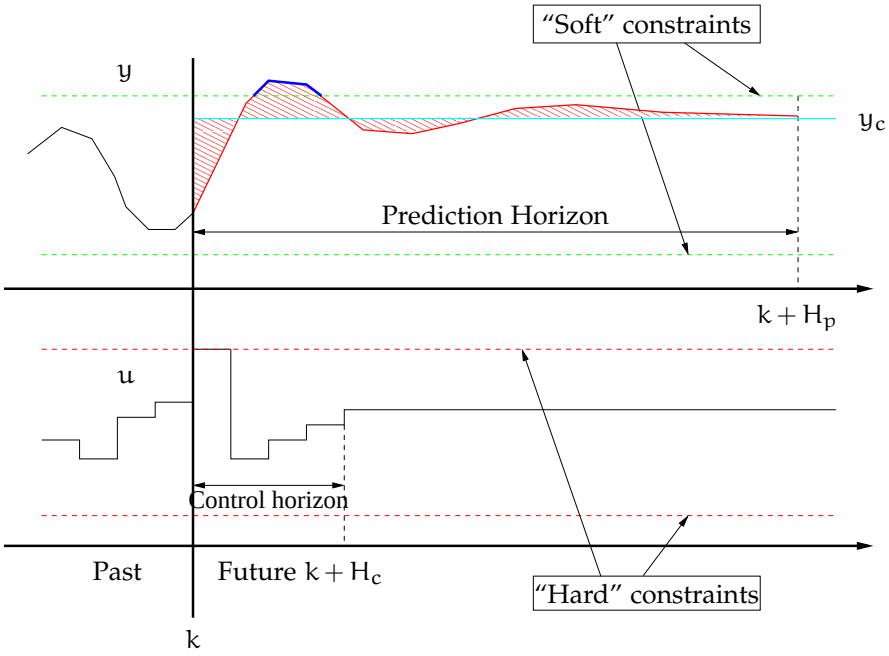

Figure 18: Graphical representation of the dynamic module

problems the second one is preferred, for its feature of smoothness.

An important note is that MPC operates with a "Receding horizon" approach: this means that only the first control action of the optimal sequence of H_c occurrences is implemented in the system, while the following are discarded. So, the optimal sequence is recalculated at every sampling time.

Even in this case, it is important to derive an analytic expression for this module: in table 3 this function is built, reporting the meaning of the single parts it is composed of.

Table 3: QP expression for the dynamic module

	QP
Cost	$(\hat{y}_k - y_s)' Q (\hat{y}_k - y_s)$
Penalization	$\bar{\epsilon}' \bar{Q} \bar{\epsilon} + \underline{\epsilon}' \underline{Q} \underline{\epsilon}$
Smoothness	$\Delta u' S \Delta u$
f	$\bar{\epsilon}' \bar{Q} \bar{\epsilon} + \underline{\epsilon}' \underline{Q} \underline{\epsilon} + \Delta u' S \Delta u + (\hat{y}_k - y_s)' Q (\hat{y}_k - y_s)$

Here the cost part represent an approximation of the integral of the red area in figure 18, where $y_s = C x_s + C_d d_{k|k}$. The penalization is similar to the one in SS module, while a "smoothness" expression is introduced: this measures the movements of inputs, and avoid a too large difference between u_k and u_{k+1} in

order to preserve actuators from large variations, which could lead to damages on the mechanical parts of the plant.

Constraints for this problems are the following:

Prediction constraint: it is used to predict states and outputs values form sampling time k to sampling time $k + H_p$

$$\begin{aligned} x_{j+1} &= Ax_j + Bu_j + B_d d_{k|k} \\ y_j &= Cx_j + C_d d_{k|k} \end{aligned} \tag{2.10}$$

Stationary constraint: values for u_j shall be equal to the steady state after the limit of the control horizon

$$u_j = u_s \quad \longrightarrow \quad j \geqslant k + Hc \tag{2.11}$$

Hard constraints: u_j and Δu_j shall not violate their maximum and minimum admissible value

$$\begin{aligned} u^{min} &\leqslant u_j \leqslant u^{max} \\ \Delta u^{min} &\leqslant \Delta u_j \leqslant \Delta u^{max} \end{aligned} \tag{2.12}$$

Soft constraints: a penalty is introduced in case y_j goes over the soft limits

$$\begin{aligned} y^{min} - \underline{\epsilon} \leqslant \hat{y}_j \leqslant y^{max} + \bar{\epsilon}; \\ \underline{\epsilon} \geqslant 0; \\ \bar{\epsilon} \geqslant 0; \end{aligned} \tag{2.13}$$

In conclusion, minimization problem for the dynamic module becomes:

$$\begin{aligned} u_k^* = \arg \min_{u_j, \bar{\epsilon}, \underline{\epsilon}} & \sum_{k}^{k+H_p} \bar{\epsilon}' \bar{Q} \bar{\epsilon} + \underline{\epsilon}' \underline{Q} \underline{\epsilon} + \Delta u' S \Delta u + (\hat{y}_k - y_s)' Q (\hat{y}_k - y_s) \\ & \text{subject to} \\ & x_{j+1} = Ax_j + Bu_j + B_d d_{k|k}; \\ & y_j = Cx_j + C_d d_{k|k}; \\ & u_j = u_s \quad \longrightarrow \quad j \geqslant k + Hc; \\ & u^{min} \leqslant u_j \leqslant u^{max}; \\ & \Delta u^{min} \leqslant \Delta u_j \leqslant \Delta u^{max}; \\ & y^{min} - \underline{\epsilon} \leqslant \hat{y}_j \leqslant y^{max} + \bar{\epsilon}; \\ & \underline{\epsilon} \geqslant 0; \\ & \bar{\epsilon} \geqslant 0. \end{aligned} \tag{2.14}$$

The real action sent to the process is the first occurrence of the u_k^* matrix, that is $u_k^*(:, 1)$ in a MATLAB notation. The receding horizon approach imposes to discard the other occurrences and to recalculate the whole u_k^* sequence at the next sampling time.

2.3 SOME FINAL POINTS ON MPC

At this point, MPC structure results well defined. The various modules it is composed of were introduced and analyzed in detail, focusing on everyone of them and reporting their main characteristics. Two minor points remain to be investigated in order to obtain a complete description of that controller: disturbance model choice and MPC tuning.
The first was shortly introduced in the previous chapters, but only minor information was given.
The second one is useful to understand the meaning of weighting matrices in MPC modules and their effect on controller behavior.

2.3.1 Disturbance model choice

Disturbance model selection, that is the choice for B_d, C_d, L_x and L_d, is usually done using standard solutions, such as input disturbance model(IDM) and output disturbance model (ODM).

IDM: this disturbance model considers that the disturbance is an unmeasured input entering in the process, so it is filtered by its dynamics. With this disturbance model, matrices assume the following values:

$$B_d = B \quad ; C_d = \mathbf{0}; \quad L_x = \mathbf{0}; \quad L_d = \mathbf{I}. \tag{2.15}$$

B_d is not necessary equal to B, but this choice is quite standardized: the only restriction on B_d value is that it shall be not a zero matrix. Referring to figure 19 the meaning of the input disturbance model results clear.

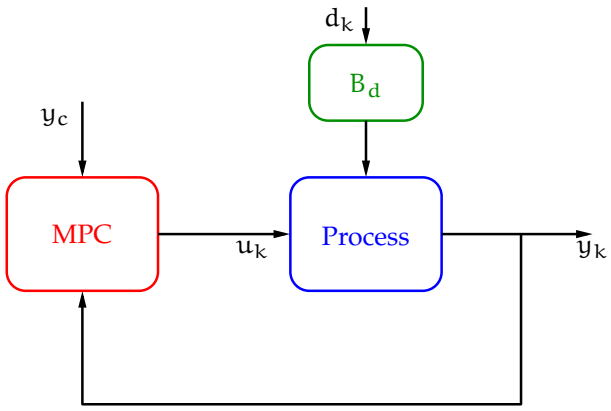

Figure 19: Input disturbance model

ODM: in this case, the disturbance is not filtered by the process, but it enters directly on the outputs, as it is possible to see in figure 20. For this kind of model, it is usual to make the following assumptions for the matrices

$$B_d = \mathbf{0}; \quad ; C_d = \mathbf{I}; \quad L_x = \mathbf{0}; \quad L_d = \mathbf{I}. \tag{2.16}$$

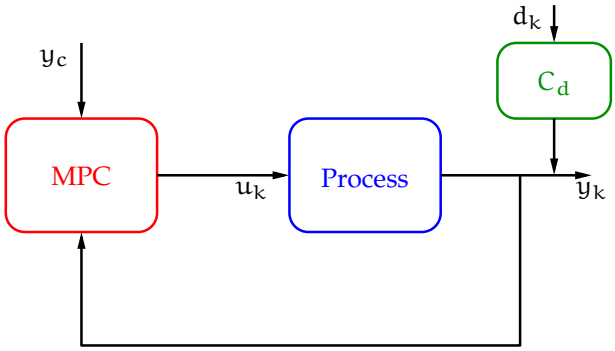

Figure 20: Output disturbance model

Another annotation can be made on the filter L_x, L_d: even if it is possible to select it in different ways, no one of them can be defined optimal. The problem of finding the optimal filter was solved by Kalman, who derived the expression for the so called Kalman filter. The description of Kalman filter is not in the objectives of this work so it will be briefly introduced here.
Consider the system in 2.17, where filter and state equations are synthesized in one expression

$$\begin{aligned} \hat{x}_{k+1|k} &= A\hat{x}_{k|k-1} + Bu_k + K(y_k - \hat{y}_k) \\ \hat{y}_k &= C\hat{x}_{k|k-1} \end{aligned} \tag{2.17}$$

Now suppose that

- w_k and v_k are statistically independent, that is they do not present correlation both in time and with each other;
- w_k and v_k have a Gaussian distribution with zero mean;
- process matrices A, B, C are exactly known;

subtracting the result in 2.17 from the real state equation, it is possible to obtain an expression for the variable $\tilde{x}_k = x_k - \hat{x}_{k|k-1}$

$$\tilde{x}_{k+1} = (A - KC)\tilde{x}_k + w_k + Kv_k \tag{2.18}$$

Kalman filter at time k is the K_k matrix for which the variance of $\tilde{x}_k$ results the minimum, that is the one which minimizes the difference between the real state value and the predicted one as in equation 2.19

$$K_k = \arg\min_K P_k = \arg\min_K E[\tilde{x}'_{k+1}\tilde{x}_{k+1}] \tag{2.19}$$

where the $E[\cdot]$ operator denotes the expected value.
Actually, the minimization problem in the (2.19) has an analytic solution, so no minimizing operations are needed at every sampling time and K_k values are calculated in a straight way.
This solution is presented in eq. 2.20

$$\begin{aligned} K_k &= P_k(P_k + \text{cov}\,(v_k))^{-1} \\ P_{k+1} &= P_k\text{cov}\,(v_k)(P_k + \text{cov}\,(v_k))^{-1} + \text{cov}\,(w_k) \end{aligned} \tag{2.20}$$

There are two different types of Kalman filter, that is the dynamic one and the static one: the first one refreshes its value at every sampling time k with P_k changing, while the second one considers that, for large k, $E[\tilde{x}'_{k+1}\tilde{x}_{k+1}]$ reaches the steady state value and so $P_k = P$ and a constant Kalman gain $K_k = K$ is obtained. In this work, the latter one is considered.

2.3.2 MPC tuning

MPC uses several weighting matrices as previously seen, and in particular in the formulation described above:

- Steady state module:
  - R(for QP) or r(for LP) to modify the costs (importance) of inputs;
  - $\underline{q}$ and $\overline{q}$ for the penalties in case of soft constraints overlap;
- Dynamic module:
  - Q for outputs to go to set-point;
  - S to manage the velocity of inputs changes;
  - $\underline{q}$ and $\overline{q}$ for soft constraints.

The MPC tuning derives from the interaction between these weighting matrices: they are diagonal matrices in which every diagonal element refers to an element in the vectors it multiplies. Using an MPC similar to the one presented in this work, the higher the weight on a variable, the more it costs, so the system tries to maintain it as closer to the steady state value as possible. This means as closer as possible to u_s for the inputs, to y_c for the outputs and to 0 for $\underline{\epsilon}$ and $\overline{\epsilon}$.
Usually, MPC tuning is based on the experience of the operator, who can decide the best values for weights. The use of a tuning method is not common, even if some of them are at disposition in the literature.

Two plot are reported to show effect of tuning in a qualitative way. In figure 21 and 22 the behavior of inputs and outputs using two MPC algorithms with different tunings. These data comes from a 2×2 system with no hard constraints both on inputs and outputs, and the tuning parameters are reported in table 4

Table 4: Tuning parameters for example in figures 21 and 22

	Figure 21	Figure 22
R_s	I	I
Q	I	0.01I
S	0.01I	100I

In figure 21 the case of heavy weights on inputs variations is reported, while in figure 22 the case of heavy weights on outputs

is plotted.
It is interesting to notice the differences between these two:

- in the first one, the outputs reach set-point values slower than in the second one, but inputs do not suffer for too fast changes;
- in the second one, the inputs show very fast changes but the outputs go to SP value quickly.

An optimal MPC tuning considers both the time required for outputs to reach set point and the magnitude of inputs variation, which should be not too fast to preserve actuators from heavy stresses. These characteristics are strongly case-dependent, because every plant presents different necessities in terms of production and control action smoothness. It is natural that an extensive knowledge of the plant is critical in this sense, because it permits to define which are optimal performances in every single case.

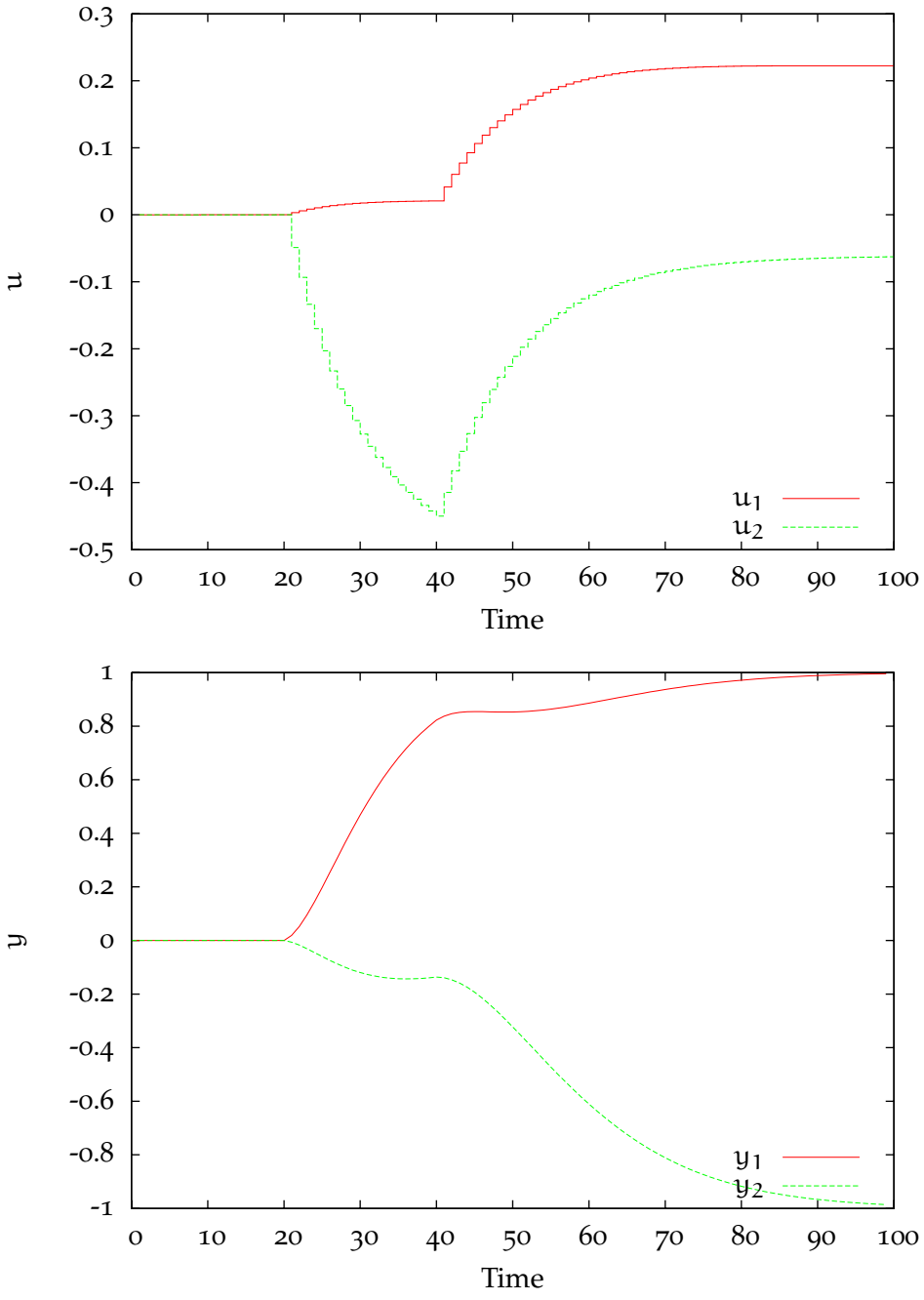

Figure 21: MPC tuning: heavy weights on input variations, inputs (top) and outputs (bottom)

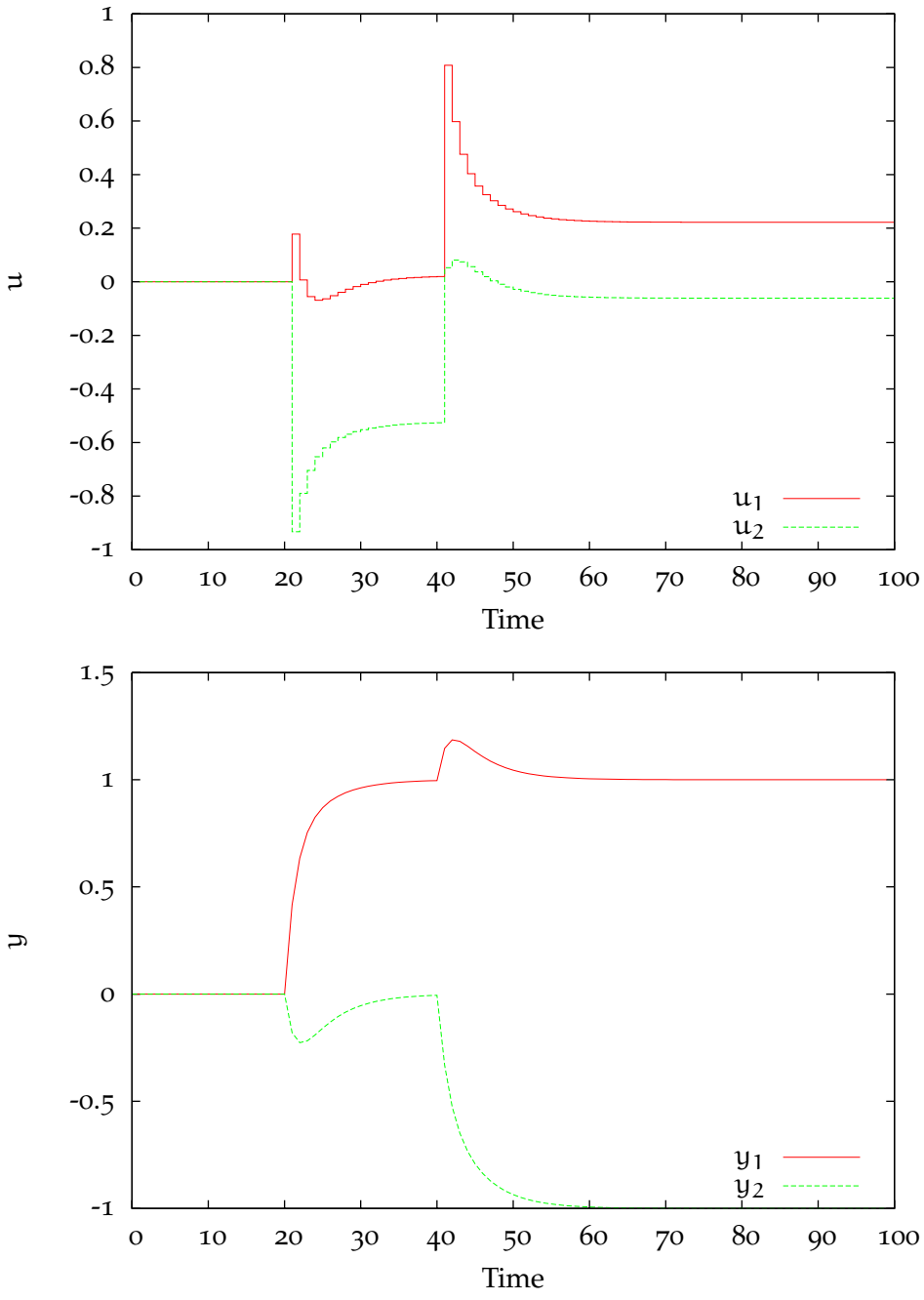

Figure 22: MPC tuning: heavy weights on set-points, inputs (top) and outputs (bottom)

3 COMPARISON OF INPUT SIGNALS IN SUBSPACE IDENTIFICATION OF MULTIVARIABLE ILL-CONDITIONED SYSTEMS

Contents

3.1 Introduction and previous work 45
3.2 Design and analysis of input signals for ill-conditioned processes 48
3.2.1 Design of rotated inputs 48
3.2.2 Issues in model order recovery 51
3.3 Case studies 52
3.3.1 Introduction 52
3.3.2 Inputs and outputs plots (Example #1) 53
3.3.3 Model order recovering (Example #1) 56
3.3.4 Quality of the identified models 57
3.3.5 Effect of the identified models on MPC closed-loop behavior 58
3.3.6 Practical issues in implementing rotated inputs 60
3.4 Conclusions 60

Ill-conditioned processes often produce data of low quality for model identification in general, and for subspace identification in particular, because data vectors of different outputs are typically close to collinearity, being aligned in the "strong" direction. One of the solutions suggested in the literature is the use of appropriate input signals, usually called "rotated" inputs, which must excite sufficiently the process in the "weak" direction. In this chapter open-loop (uncorrelated and rotated) random signals are compared against inputs generated in closed-loop operation, with the aim of finding the most appropriate ones to be used in multivariable subspace identification of ill-conditioned processes. Two multivariable ill-conditioned processes are investigated and as a result it is found that closed-loop identification gives superior models, both in the sense of lower error in the frequency response and in terms of higher performance when used to build a model predictive control system.

3.1 INTRODUCTION AND PREVIOUS WORK

During the last decades, a large number of valuable contributions were brought in system identification, especially for the definition of consistent, reliable and numerically efficient identification algorithms. In general, system identification can be per-

formed using two different approaches: Prediction Error methods (PE) and Subspace IDentification methods (SID). The first approach (PE) is based on the simple idea that, in absence of disturbances, the better the model recovered from data, the smaller the prediction error it generates, where the prediction error is the difference between the measured output and the model output data. The second approach (SID) instead involves particular matrices obtained from output and input data and performs projection operations to cancel out the noise contributions. Thus, the system model is obtained in state-space form using these projected data matrices. PE methods were principally developed and probably called with this name for the first time by Ljung (see [20] for a complete overview of PE algorithms), but, even before his works, it was common to minimize the prediction error for identification of process parameters. SID is a relatively young technique, developed mainly during the last fifteen years, even if its basis were posed several years before. Van Overschee and De Moor [35] and Verhaegen [37] introduced and solved the subspace problem using different approaches, called N4SID and MOESP, respectively, even though it was shown later in [36] that those methods use the same subspace, and that they only differ in the weighting matrices. Another successful class of subspace methods is that referred to as CVA algorithms, developed by Larimore [19]. During the years, several modifications and improvements were made on those algorithms with the aim of enhancing numerical stability and efficiency. In particular, several new subspace approaches were recently developed, such as those from Wang and Qin [38] and Huang et al. [15]. This latter algorithm is that used in the present work, with some modifications introduced to improve results in the computation of the system matrices.

As remarked by Zhu [42], however, a fundamental portion of the "identification problem", that is test design, received less attention. This issue was studied for the single-input single-output (SISO) case by Gevers and Ljung [9], who showed that it was possible/desirable to design inputs in dependence of intended model application. After this work, Koung and MacGregor [17, 18] addressed the problem of optimal test design and system identification for robust control of multi-input multi-output (MIMO) processes. Hjalmarsson et al. [14] emphasized that models of superior quality are obtained from closed-loop tests, particularly for model-based control purposes. Nonetheless, during these years (and in a large number of cases still at present days) standard industrial inputs for system identification were considered to be step tests. This approach is, in general, suboptimal because output data obtained in such a way lack of information about the system to be identified, since the inputs are rather poor in terms of frequency content. Alternatively, one can use binary random or pseudo-random inputs, because of their superior power spectrum with respect to that of step signals [20, 43].

Unlike SISO processes, MIMO systems may show "directions" (in the input vector space) in which the (steady-state or dynamic) effect of the inputs on the process outputs is much larger than in other directions. In such situations the process is said to be ill conditioned, and frequent examples of ill-conditioned systems are high-purity distillation columns. Ill-conditioned processes are usually difficult to be controlled because decentralized controllers are typically inadequate due to large interactions among the control loops, but multivariable model-based controllers may suffer from robustness issues [26, 32]. Moreover, ill-conditioned processes may be difficult to deal with from an identification point of view, because traditional uncorrelated open-loop step tests tend to excite the system mostly in high-gain directions [16, 17]. In fact, depending on the input directions, the output response can vary significantly in magnitude, even thousands times from the high-gain to the low-gain direction. Thus, the information coming out from the high-gain direction is predominant over that from the low-gain direction, often resulting in a model not suitable for control purposes [11, 18].

Koung and MacGregor [17, 18] developed a test design method using highly correlated input signals, which are able to excite the process both in high-gain and low-gain directions. However, highly correlated input signals may constitute a major trouble when SID methods are used, especially for system order recovering: in fact, matrices used for identification could be near to rank deficiency in systems of this type, and so algorithms could lead to incorrect numerical results [4]. Misra and Nikolaou [23] also presented a work dealing with this issue: differently from [17, 18], they focused only on order determination (and not on model recovering), and introduced the problem from the point of view of subspace identification. In particular, they proposed an open-loop test design based on "rotated" inputs, in which the angles between the inputs are obtained by trial and error. Conner and Seborg [5], similarly, indicated the same rotated inputs as the best solution for system identification. Other relevant contributions on test design were given by Stec and Zhu [34] who introduced a quite different design, composed by a part of high magnitude collinear inputs, used to estimate characteristics of the low-gain direction, and small uncorrelated inputs, ideal to estimate properties of the process in high-gain direction, by Cooley and Lee [6] who obtained control-relevant finite-impulse response (FIR) models, by Gopaluni, Pathwardan and Shah [10] who proposed a test design particularly suitable to recover good Model Predictive Control (MPC) oriented models (minimizing j-step ahead prediction error). Stec and Zhu expanded their work in [41], where they introduced another kind of test design, in which the input signal is the sum of a high amplitude linearly dependent signal and a low amplitude uncorrelated signal. Hjalmarsson [13] presented an all-inclusive paper spanning the system identification problem, test design included. Recently, Bruwer and MacGregor [3] extended the work

in [17, 18] to the (frequent) case of presence of physical and process constraints, also considered in a recent work by Zhan et al. [40].

The main objective of this work is the comparison of different test signals for subspace identification of ill-conditioned systems. In particular, advantages and disadvantages of rotated inputs are analyzed and these characteristics are compared with those of closed-loop signals. The identified models are then compared in terms of frequency error analysis and closed-loop performance when used in a Model Predictive Control algorithm. The rest of this chapter is organized as follows. Section 3.2 is focused on the analysis of the different types of signals with particular attention to their effects on system order recovering and overall quality of the identified models. Moreover, a generalization of the rotated inputs design procedure to non-square multivariable systems of arbitrary dimensions is presented. In Section 3.3 two case studies of ill-conditioned distillation columns are reported to show the results obtained using different kinds of signals and discuss the problems that could be experienced using rotated inputs. Finally, Section 3.4 summarizes the main achievements of this work, namely that closed-loop inputs give better models than open-loop rotated inputs.

3.2 DESIGN AND ANALYSIS OF INPUT SIGNALS FOR ILL-CONDITIONED PROCESSES

Ill-conditioned systems, as previously said, represent one of the most difficult kind of linear processes to be identified: *gains* appear very different depending on input direction, so several difficulties could be experienced. Models obtained for these processes often suffer from significant errors in order and gain recovery. Indeed, it can be seen in [23] that matrices used for subspace identification in a lot of cases have nearly collinear rows, and this can lead to incorrect order estimation and subsequently in erroneous models.

3.2.1 Design of rotated inputs

In order to avoid the generation of output data mostly aligned in the strong direction, Koung and MacGregor [18] and Misra and Nikolaou [23] propose the use of "rotated" inputs, in which the input vectors are strongly aligned in pre-specified directions. For a 2×2 system, this means that the input vectors form particular angles w.r.t. the principal axes in a (u_1, u_2) plane, and these angles are referred to as "rotation angles". The algorithm for the generation of rotated inputs in the 2×2 case [18, 23] is here extended to the most general case of non-square multivariable systems of arbitrary dimensions.

Let $G = C(I - A)^{-1}B \in \mathbb{R}^{p\times m}$ be the gain matrix of the system, and consider the steady-state relation $y^s = Gu^s$, in which from now on the superscript ^{s} indicates *steady state*, which can be written as follows by means of an SVD:

$$y^s = U_G S_G V_G' u^s \tag{3.1}$$

where $U_G \in \mathbb{R}^{p\times p}$, $S_G \in \mathbb{R}^{p\times m}$ and $V_G \in \mathbb{R}^{m\times m}$. Introducing a new variable $t = \min(p, m)$, i.e. the minimum between the number of outputs and the number of inputs of the system, it is possible to reduce these matrices to e $U_{Gr} \in \mathbb{R}^{p\times t}$, $S_{Gr} \in \mathbb{R}^{t\times t}$ and $V_{Gr} \in \mathbb{R}^{t\times m}$. It is easy to demonstrate that eq. 3.2 still holds

$$\begin{aligned} G_s &= U_G S_G V_G' = U_{Gr} S_{Gr} V_{Gr}' = \\ &= \begin{bmatrix} \mathbf{u}_{11} & \cdots & \mathbf{u}_{1t} \\ \vdots & & \vdots \\ \mathbf{u}_{p1} & \cdots & \mathbf{u}_{pt} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{s}_t \end{bmatrix} \begin{bmatrix} \mathbf{v}_{11} & \dots & \mathbf{v}_{m1} \\ \vdots & & \vdots \\ \mathbf{v}_{1t} & \dots & \mathbf{v}_{mt} \end{bmatrix} \end{aligned} \tag{3.2}$$

Now, perform the computation of the term $S_{Gr}V_{Gr}u_s$

$$\begin{aligned} y_s &= U_{Gr} S_{Gr} V_{Gr}' u_s = \\ &= \begin{bmatrix} \mathbf{u}_{11} & \dots & \mathbf{u}_{1t} \\ \vdots & & \vdots \\ \mathbf{u}_{p1} & \dots & \mathbf{u}_{pt} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1(\mathbf{v}_{11}u_1^s + \cdots + \mathbf{v}_{m1}u_m^s) \\ \vdots \\ \mathbf{s}_t(\mathbf{v}_{1t}u_1^s + \cdots + \mathbf{v}_{mt}u_m^s) \end{bmatrix} \end{aligned} \tag{3.3}$$

Introduce a new series of variables $\xi_h = \mathbf{s}_h(\mathbf{v}_{1h}u_1^s + \cdots + \mathbf{v}_{mh}u_m^s)$ and substitute their values in the previous equation

$$\begin{aligned} y_s &= \begin{bmatrix} \mathbf{u}_{11} & \cdots & \mathbf{u}_{1t} \\ \vdots & & \vdots \\ \mathbf{u}_{p1} & \cdots & \mathbf{u}_{pt} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \vdots \\ \xi_t \end{bmatrix} = \\ &= \xi_1 \begin{bmatrix} \mathbf{u}_{11} \\ \vdots \\ \mathbf{u}_{p1} \end{bmatrix} + \cdots + \xi_t \begin{bmatrix} \mathbf{u}_{1t} \\ \vdots \\ \mathbf{u}_{pt} \end{bmatrix}, \end{aligned} \tag{3.4}$$

Now, it is clear the problem arising with ill conditioned systems: in this kind of systems, since $\mathbf{s}_1 \gg \mathbf{s}_t$, the sequence output vectors (at steady-state) is mostly aligned over the space spanned by the first columns of the left singular vectors matrix U_G, i.e. in the stronger directions. So, if a sequence input vectors u^s is not *carefully* generated (e.g. a random or pseudo-random input vector sequence), outputs are heavily influenced only by the vector x_1 and possibly not good to be used for identification.. In other words, unless special care is taken in choosing the input vectors, it follows that $|\xi_1| \gg |\xi_t|$ and hence the last terms in the sum (3.4) become negligible, so that no information about the system's weaker directions is contained in the data.
In the spirit of [18], here the goal is to construct a vector (or a sequence of vectors) u^s such that the corresponding (steady-state) output vector contains information regarding all singular values

in *equal* magnitude. Since the columns of the left singular vector matrix U_G are normalized, such a goal can be achieved by imposing the following $t-1$ conditions[1]:

$$\xi_1 = \xi_2 = \cdots = \xi_t \,, \tag{3.5}$$

which can be explicitly written as:

$$\underbrace{\begin{bmatrix} \nu_{1,2} & \dots & \nu_{m,2} \\ \vdots & & \vdots \\ \nu_{1,t} & \dots & \nu_{m,t} \end{bmatrix}}_{\Theta} \begin{bmatrix} u_1^s \\ \vdots \\ u_m^s \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \,, \tag{3.6}$$

where $\nu_{\alpha,\beta} = s_1 \mathbf{v}_{\alpha,1} - s_\beta \mathbf{v}_{\alpha,\beta}$.

The linear problem (3.6) consists of $t-1$ equations in m variables; so let $z = m - t + 1$ be the number of degrees of freedom, i.e. the number of input components that can be chosen arbitrarily. From the definition of t, it is straightforward to see that $z \geqslant 1$. Assume that the first z components of u^s are chosen arbitrarily, the remaining $m - z = t - 1$ components can be computed so that (3.6) is satisfied by solving the following square linear system:

$$\underbrace{\begin{bmatrix} -\nu_{z+1,2} & \dots & -\nu_{m,2} \\ \vdots & & \vdots \\ -\nu_{z+1,t} & \dots & -\nu_{m,t} \end{bmatrix}}_{\Phi} \begin{bmatrix} u_{z+1}^s \\ \vdots \\ u_m^s \end{bmatrix} = \underbrace{\begin{bmatrix} \nu_{1,2} & \dots & \nu_{z,2} \\ \vdots & & \vdots \\ \nu_{1,t} & \dots & \nu_{z,t} \end{bmatrix}}_{\Psi} \begin{bmatrix} u_1^s \\ \vdots \\ u_z^s \end{bmatrix}$$

where the vector $\begin{bmatrix} u_{z+1}^s & \dots & u_m^s \end{bmatrix}'$ contains the unknowns. Assuming that Φ is invertible, the vector of unknown input components can be computed as:

$$\begin{aligned} \begin{bmatrix} u_{z+1}^s \\ \vdots \\ u_m^s \end{bmatrix} &= \begin{bmatrix} -\nu_{z+1,2} & \dots & -\nu_{m,2} \\ \vdots & & \vdots \\ -\nu_{z+1,t} & \dots & -\nu_{m,t} \end{bmatrix}^{-1} \cdot \\ &\cdot \begin{bmatrix} \nu_{1,2} & \dots & \nu_{z,2} \\ \vdots & & \vdots \\ \nu_{1,t} & \dots & \nu_{z,t} \end{bmatrix} \begin{bmatrix} u_1^s \\ \vdots \\ u_z^s \end{bmatrix} = \\ &= \Phi^{-1} \Psi \begin{bmatrix} u_1^s \\ \vdots \\ u_z^s \end{bmatrix} \end{aligned} \tag{3.7}$$

In conclusion, a strategy for the design of a rotated input sequence is the following, which represents an extension of the procedure proposed in [23] to higher dimension (possibly non-square) systems.

1. *Define a random (or pseudo-random) multivariable binary signal* $\begin{bmatrix} u_{1,k} & \cdots & u_{z,k} \end{bmatrix}'$ *of dimension* $z = m - t + 1$, *for* $k = 0, \dots, L-1$.

1 Notice that (3.5) is only one of the possible 2^{t-1} valid combinations of the most general conditions: $\xi_1 = \pm\xi_2 = \cdots = \pm\xi_t$.

2. *Perform an SVD of the system gain matrix* G *(or an approximation to it), compute the matrices* Φ^{-1} *and* Ψ *defined in* (3.2.1).

3. *For each sample time* k*, compute the remaining* $t-1$ *input components as:*

$$\begin{bmatrix} u_{z+1,k} \\ \vdots \\ u_{m,k} \end{bmatrix} = \Phi^{-1}\Psi \begin{bmatrix} u_{1,k} \\ \vdots \\ u_{z,k} \end{bmatrix} + \zeta_k \,, \tag{3.8}$$

in which $\zeta \in \mathbb{R}^{t-1}$ *is a white noise signal with "small" amplitude (dithering), used to avoid exact collinearity of the inputs.*

Notice that, in case Φ is not invertible, a possible alternative for obtaining the inputs sequence can be recovered starting from eq. 3.6. Computing a full rank matrix $\mathcal{N}_\Theta \in \mathbb{R}^{m\times z}$ such that $\Theta\mathcal{N}_\Theta = 0$ and defining, for each sampling time k, the whole input vector as:

$$\begin{bmatrix} u_{1,k} \\ \vdots \\ u_{m,k} \end{bmatrix} = \mathcal{N}_\Theta \mu_k + \zeta_k$$

in which $\mu \in \mathbb{R}^z$ is the primary random (or pseudo-random) multivariable binary signal and $\zeta \in \mathbb{R}^m$ is a white noise (dithering) signal, the inputs sequence can be obtained. The dithering signal must be added to the input sequence in both the cases to avoid a too high collinearity in the inputs, which would be problematic in identification as it will be showed in the following section.

Notice also that Φ and Ψ vectors are constant, so it is not necessary to re-calculate them for every sampling time.

3.2.2 Issues in model order recovery

As discussed in Section 1.4.2, the considered subspace algorithm performs linear projections of a matrix Z_f formed by output and input data. In absence of noise, by its own construction (see eqs. 1.35 and 1.39), it immediately follows that:

$$\operatorname{rank} Z_f = \operatorname{rank}\left(\begin{bmatrix} Y_f \\ U_f \end{bmatrix}\right) = \operatorname{rank}\left(\begin{bmatrix} \Gamma_r & H_r^u \\ 0 & I \end{bmatrix}\begin{bmatrix} X_f \\ U_f \end{bmatrix}\right) \leqslant mr + n$$

in which the equality sign holds under appropriate conditions on the input excitation [38, Lemma 1]. An SVD performed on[2] Z_f leads to $mr + \tilde{n}$ non-zero singular values, where $\tilde{n} \leqslant n$. As discussed in Section 3.2.1 and also remarked in Section 3.3.2, for ill-conditioned plants when open-loop random inputs are used, the sequences of outputs are close to collinearity. Therefore, Z_f shows a number of significant singular values less than $mr + n$,

2 Notice that the adopted identification algorithm performs an SVD of the projected data matrix Z, defined in (1.26). However, it is obvious from (1.26) that $\operatorname{rank} Z \leqslant \operatorname{rank} Z_f$, so that the rank bound on Z_f is a valid bound for Z as well.

the selected model order $\tilde{n}$ is less than n, and the model will be missing information about the weaker directions. In presence of noise, the number of significant singular values (and therefore the selected model order), may increase but still no information regarding the weaker directions is obtained because the random open-loop sequence of inputs does not excite the plant sufficiently in those directions.

As discussed, one of the solutions (and probably the most popular in the academic literature) that were proposed in the last years to avoid problems of order recovering in ill-conditioned processes is the use of "rotated" inputs (see [18, 23]). Such input design was generalized to (possibly non-square) systems of arbitrary dimension in Section 3.2.1. However, the rank analysis previously described emphasizes that if the input vectors are strongly aligned in particular directions, as required by the rotated input design approach, the matrix U_f may show nearly collinear rows, so that the overall matrix Z_f may be again near to rank deficiency, because some of mr singular values associated to U_f may not be significant. That is, although the rotated input design approach may be useful to excite the system in weaker directions, the strong alignment of the inputs in specific directions may constitute a problem in all subspace identification algorithms which project input data as well as output data.

Unfortunately, the algorithm defined in chapter 1 is one of this kind of methods, so there could be problems in

3.3 CASE STUDIES

3.3.1 Introduction

Two different case studies are presented to compare the behavior of rotated inputs vs. that of random OL inputs and CL inputs obtained with random setpoint signals. OL identification data are constructed by using as inputs Generalized Binary Noise (GBN) signals, either uncorrelated or correlated in the case of rotated inputs, as outlined at end of Section 3.2.1. Introduced by Zhu [43], GBN signals have many favorable features, in particular in terms of frequency content, which is typically superior to that of Pseudo-Random Binary Noise (PRBS) and of step signals. Closed-loop data are obtained by using an MPC regulator as that described in [28], based on a preliminary (erroneous) model of the system to be identified. In CL data collection, setpoints for the controlled variables are GBN signals. These simulations are performed in order to find the most appropriate test input design for subspace identification of ill-conditioned processes. The identified models are compared in terms of frequency responses and in terms of performance of (unconstrained) MPC regulators based on the identified models. The final goal of these studies is to explain why CL data collection is to be preferred for SID of ill-conditioned plants.

Example #1 is the "classical" two input - two output high-purity distillation column studied by Skogestad and Morari [33]. A transfer function model of the process is the following:

$$y = \begin{bmatrix} \frac{87.8}{194s+1} & \frac{-87.8}{194s+1} + \frac{1.4}{15s+1} \\ \frac{108.2}{194s+1} & \frac{-108.2}{194s+1} + \frac{-1.4}{15s+1} \end{bmatrix} u \,, \tag{3.9}$$

in which the outputs are logarithmic distillate purity and logarithmic bottom impurity, the inputs are reflux and boil-up rates and the time constants are in minutes. Normally distributed output noise with a noise-to-signal ratio of 0.10 is added to both outputs, and a sampling time of 5 minutes is considered. CL identification data are obtained by using an MPC regulator, based on the following "erroneous" model:

$$y = \begin{bmatrix} \frac{75}{180s+1} & \frac{-75}{180s+1} + \frac{1.6}{19s+1} \\ \frac{105}{180s+1} & \frac{-105}{180s+1} + \frac{-1.6}{19s+1} \end{bmatrix} u \,. \tag{3.10}$$

Example #2 is a 3×5 linear system that represents a high-purity distillation column for the separation benzene-toluene (see Table 5 for a detailed list of inputs and outputs). The condition number of its gain matrix is 140. This linear discrete-time system, omitted in the sake of space, is a representation of the rigorous nonlinear simulation model, developed in Octave[3]. A 5 minute time delay (i.e. 5 samples) is considered on head and bottom impurity outputs, and normally distributed output noise with a noise-to-signal ratio of 0.10 is added to all outputs. CL identification data are collected by using an MPC regulator based on a preliminary model, identified via step tests on each input variable independently. For both examples, all datasets contain 2500 samples.

Model validation is conducted, in both examples, by means of a frequency error analysis, as described. Defining $G^r(z)$ and $G^{id}(z)$ as the real and identified discrete transfer function models, respectively, the following scalar parameter is defined to measure the quality of the identified model:

$$\bar{\epsilon} = \frac{\|G^{id}(z) - G^r(z)\|_\infty}{\|G^r(z)\|_\infty} \triangleq \sup_{\omega>0} \underbrace{\left[\frac{\sigma_1\left(G^{id}(e^{i\omega T_s}) - G^r(e^{i\omega T_s})\right)}{\sup\limits_{\omega>0} \sigma_1\left(G^r(e^{i\omega T_s})\right)} \right]}_{\epsilon(\omega)} \tag{3.11}$$

in which $\|\cdot\|_\infty$ denotes the $\mathcal{H}_\infty$ norm (in discrete-time sense), T_s is the sampling time, and $\sigma_1(\cdot)$ denotes the largest singular value of $(\cdot)$. It is clear from (3.11) that the lower $\bar{\epsilon}$ the better the identified model.

3.3.2 Inputs and outputs plots (Example #1)

Figure 23 shows the input sequences for three different data collection schemes: OL data with random inputs, OL data with

3 Octave is freely available at `http://www.octave.org`.

rotated inputs and CL data with random setpoints. For the moment, the exact system gain matrix is used to generate the sequence of rotated inputs, according to the procedure outlined in Section 3.2.1. The effect of an incorrect rotation angle is discussed later in Section 3.3.6. Figure 24 reports the inputs in the plane (u_1, u_2) and the outputs in the plane (y_1, y_2) obtained from the three data collection schemes. It is possible to see that in case of random inputs, the outputs are placed in a narrow with a dominant direction, which is the high-gain direction. Thus, it is clear that information from the low-gain direction is "hidden", because it is much smaller in magnitude and it cannot be evaluated correctly from the dataset because of the presence of noise. In case of rotated inputs, output data are contained in a region where all the directions are of similar magnitude, and this shows that the rotated input design reached its goal. However, it is possible that inputs are not enough informative, because they are strongly aligned in a specific direction. Finally, the bottom plots show that outputs obtained in closed loop present the same advantages of those coming from rotated inputs, i.e. they are not aligned in a preferred direction. At the same time, it is possible to see that the inputs obtained in closed loop are not strongly aligned in a preferred direction either.

Table 5: Inputs and outputs for Example #2.

Input		Output	
u_1	Pressure	y_1	Distillate composition
u_2	Q reboiler	y_2	Bottom composition
u_3	Reflux rate	y_3	Valve opening
		y_4	Valve opening
		y_5	Valve opening

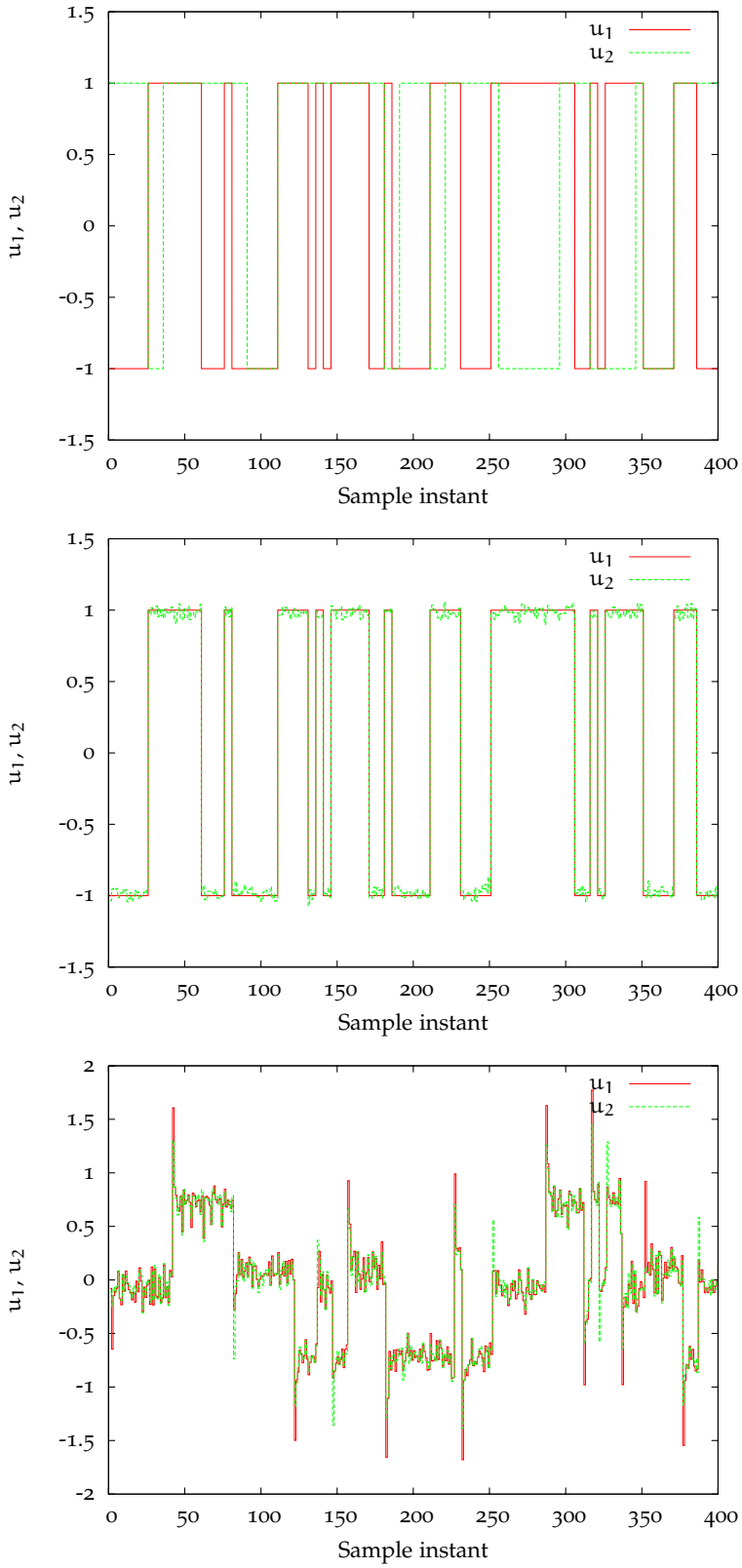

Figure 23: Example #1: standard GBN input (top), rotated GBN input (middle) and closed-loop input (bottom) signals.

3.3.3 Model order recovering (Example #1)

In this section, order recovering results are shown and discussed for Example #1. The past and future horizon is equal to $r = 30$, while a value of 0.05 is assumed for ρ in (1.48) to select the system order from the singular values of Z. During this work it was experienced that open-loop rotated inputs generate mistakes in order determination when used with subspace algorithms which project not only the future outputs matrix but also the future inputs matrix (such as the one adopted in this chapter). This occurs because the matrix U_f shows nearly collinear rows given the strong alignment of the inputs in a fixed direction as shown in Figure 24.

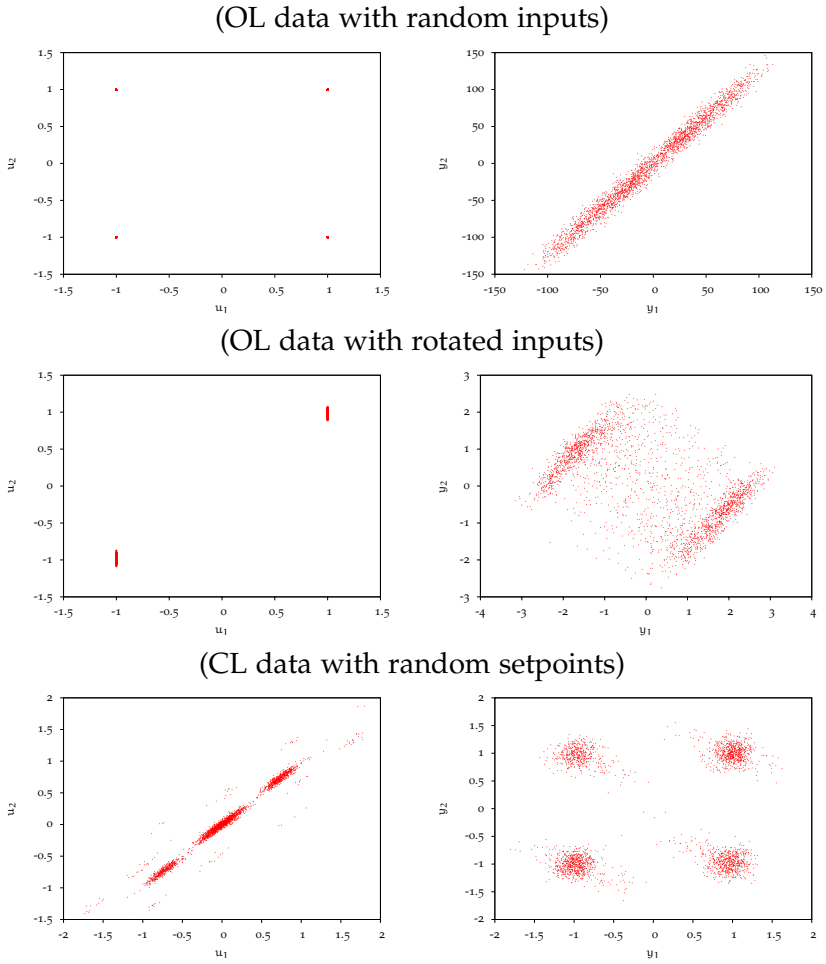

Figure 24: Example #1: alignment of inputs (left) and outputs (right) in three cases: OL data collection with random inputs, OL data collection with rotated inputs, CL data collection with random setpoints.

Singular values of projected matrix Z are computed in three different cases: OL data collection using uncorrelated GBN inputs, OL data collection using rotated GBN inputs, and CL data collection using uncorrelated GBN setpoints. For Example #1, it results that neither OL data obtained from uncorrelated inputs nor that obtained from rotated inputs allow the projection algorithm to compute the correct model order (which is 2): they give a model order of 5 and a model order of 12, respectively. On the other hand CL data permit the algorithm to recover the correct model order. In order to confirm that OL rotated inputs may work well with particular SID algorithms, it was verified that using Misra & Nikolaou's algorithm [23] on the same rotated inputs dataset leads to the correct model order. Notice that if OL uncorrelated inputs data are used, Misra & Nikolaou's identification method computes a model order of 1.

For Example #2, the orders of the models identified with the orthogonal projection method are 17 in case of OL random inputs, 17 in case of rotated OL inputs and 14 in case of CL random inputs, respectively. Notice that the correct order is 24 (including ten state variables for outputs delay).

3.3.4 Quality of the identified models

In this section, quality of the identified models is evaluated and compared in terms of frequency responses. Figure 26 shows the relative error $\epsilon(\omega)$ vs. frequency of three models identified from different datasets (OL with random inputs, OL with rotated inputs and CL with random setpoints). The top plot concerns with Example #1 and the bottom plot concerns with Example #2. From these results the superior quality of the model identified from closed-loop data clearly appears.

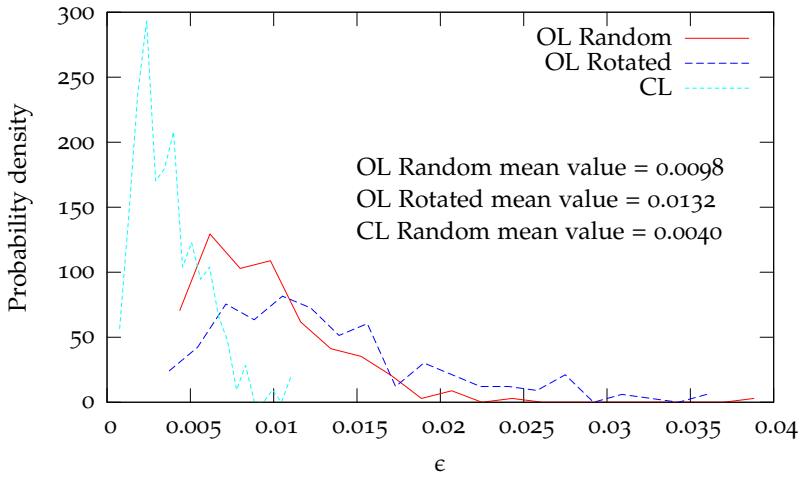

Figure 25: Example #1: density function of $\bar{\epsilon}$ obtained from a Monte-Carlo study.

It is well known that identification with subspace methods may suffer from sensitivity to different noise realizations. For

this reason, a Monte-Carlo study of 600 simulations was performed on Example #1: 200 simulations are conducted in OL using uncorrelated GBN input signals, 200 simulations are conducted with OL rotated GBN input signals, and 200 simulations are conducted in CL with uncorrelated GBN setpoints. The results of this study are reported in Figure 25, in which the density function the parameter $\bar{\epsilon}$ for the models identified from the three datasets is depicted. It can be seen that the curves do not follow a normal (symmetric) distribution and are right-skewed. Mean values are also reported in the plot, and it is possible to see that models obtained from CL data have superior accuracy (i.e. lower mean of $\bar{\epsilon}$). Moreover, it is possible to see that models obtained from CL data are more precise (i.e. lower variance of $\bar{\epsilon}$). In conclusion, this Monte-Carlo simulation study shows that CL data give superior models in terms of accuracy and precision compared to data obtained by OL random inputs and by OL rotated inputs.

3.3.5 Effect of the identified models on MPC closed-loop behavior

The closed-loop behavior of several MPC regulators, equivalent in all tuning parameters (omitted in the sake of space) but based on different models, is compared. Three MPCs are designed on the models identified from OL random, OL rotated and CL inputs, respectively; a fourth MPC is designed on the true model that can be found in the (3.9).
For Example #1, closed-loop inputs and outputs obtained for a setpoint change imposed at time 0, are shown in Figure 27. It is clear that regulator based on the model obtained from CL data shows superior performance (essentially equal to that of the regulator based on the true model). The regulators based on models identified from OL random and OL rotated inputs, instead, show a worse performance especially in terms of relevant fluctuations of the manipulated variables. These results are in definite agreement with the model error frequency responses shown in Figure 26.
Figure 28 shows closed-loop inputs and the first three outputs of Example #2, during a setpoint change on the head and bottom impurity outputs. Also for this case the regulator based on the model obtained from CL signals performs well, slightly better than the regulator based on the model identified from OL random signals (which shows a larger settling time in the output variables). For this case, the MPC based on the model identified from OL rotated signals shows poor performance with large oscillations of outputs and inputs. This poor behavior is a clear consequence of the large model error, especially at low frequency, shown in Figure 26.

From these simulation results, as well as from other case studies, it is possible to state that CL data collection is to be preferred for ill-conditioned processes, both in the sense of model error

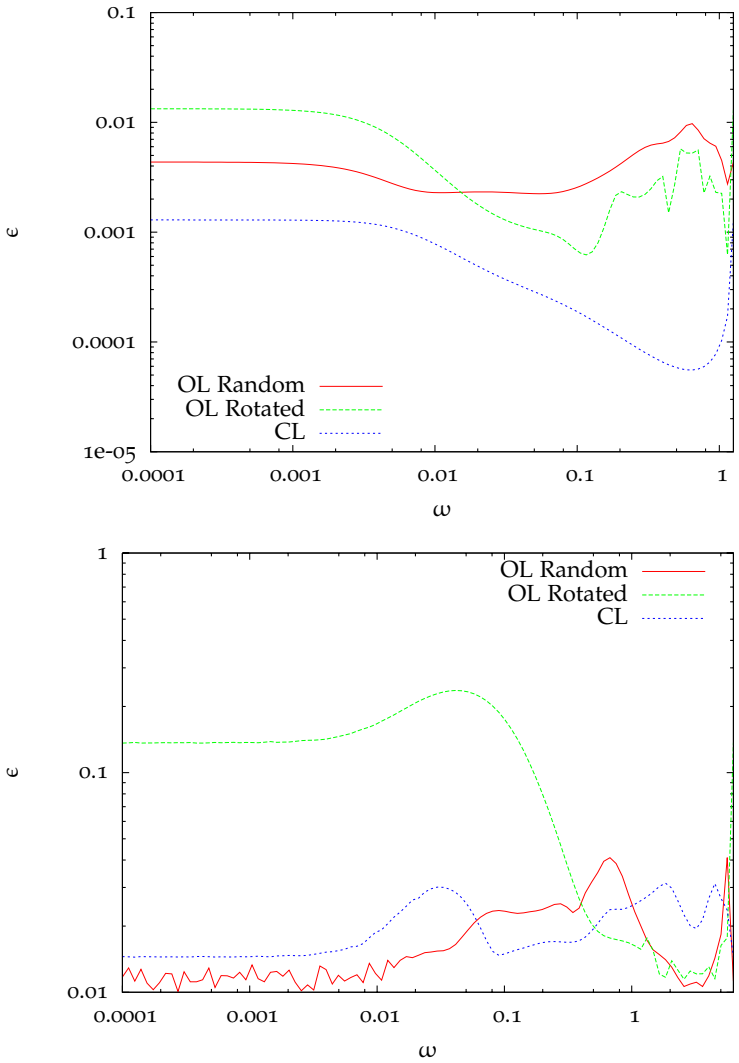

Figure 26: Model error parameter $\epsilon(\omega)$ vs. frequency for three identified models: Example #1 (top) and Example #2 (bottom).

frequency response, and in terms of superior closed-loop performance when the identified model is used inside an MPC regulator. OL data collection, even with a rotated input approach, is instead to be avoided for subspace identification of ill-conditioned processes.

These evidences can be explained, with the aid of the plots shown in Figure 24, as follows. CL data collection with random setpoints forces the outputs and the inputs of the system in many different directions, i.e. both in high-gain and low-gain directions. On the other hand, OL data collection with random inputs forces the outputs to be aligned mostly in high-gain directions, whereas OL data collection with rotated inputs certainly equalizes the contribution of high and low gain directions, but the corresponding inputs may be too strongly aligned in specific directions. This shortcoming is particularly relevant when subspace algorithms projecting the input data matrix (along with

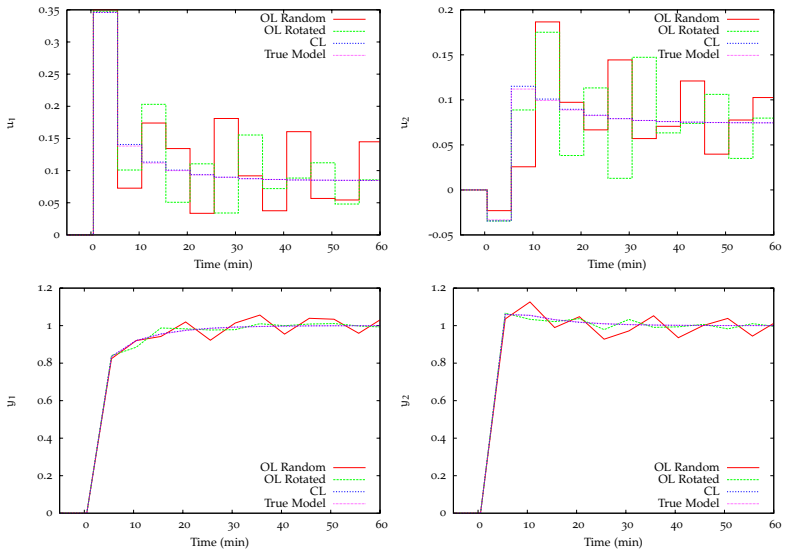

Figure 27: Example #1: closed-loop inputs and outputs during a set-point change in the direction $[1,\ 1]^T$.

the output data matrix) are applied. Other disadvantages of rotated inputs are discussed in the next paragraph.

3.3.6 Practical issues in implementing rotated inputs

Rotated inputs may guarantee good performances in model order recovery when used with specific identification methods (such as that in [23]). However, until now it has not been shown the sensitivity of this method to errors in the rotation angle, that is how rotated inputs work when the selected rotation angle is different from the optimal rotation angle, computed via SVD of the "exact" gain matrix. When the rotation angle is $\pi/3$ rather than about $\pi/4$ that is obtained from SVD of the exact system gain matrix, the model order identified from Misra & Nikolaou's algorithm [23] is 1. This confirms that rotated inputs may work well only if the correct rotation angle is applied. In practice, since the true model order is not known, it is quite difficult to tell from the singular values whether the applied rotation angle is correct or not.

3.4 CONCLUSIONS

Ill-conditioned processes are difficult to be identified from data because of problems that derive mainly from the lack of information in the weaker directions of the system. A possible solution, proposed in the literature and deeply discussed throughout this chapter, is the use of tailored inputs known as "rotated" inputs [18, 23], which excite the system equally in high-gain and low-gain directions. To construct these inputs, the system gain matrix (which is however not known precisely) can be de-

composed via singular value decomposition, and inputs are designed to generate output signals of the same magnitude both in weak and in strong directions. In this work this input design approach was generalized to non-square multivariable systems of arbitrary dimension and applied to two case studies of ill-conditioned processes. Results clearly show that, dealing with an open-loop data collection scheme, rotated inputs may grant better models than uncorrelated inputs, but these superior results are strongly dependent on the subspace identification algorithm used. In particular rotated inputs appear inappropriate for the use with the orthogonal projection method [15]. Moreover, the effectiveness of rotated inputs is strongly related to the accuracy of the applied rotation angle(s), which in general must be found by trial and error. Results show that closed-loop data collection, instead, guarantees superior models, both in terms of lower error in frequency response between the identified models and true process, and most importantly in terms of higher performance achieved by model predictive controllers based on the identified models. Furthermore, this closed-loop test design is to be preferred because random setpoints can be easily generated without necessity of several trials to find the most appropriate rotation angle(s) of the inputs.

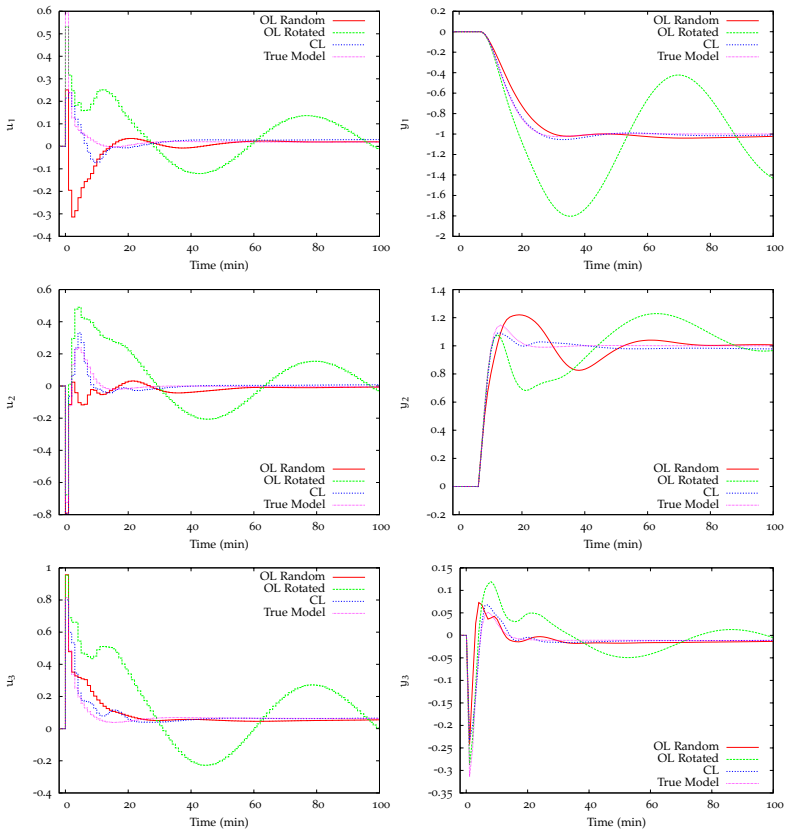

Figure 28: Example #2: closed-loop inputs and three outputs during a setpoint change on the product compositions.

4 SUBSPACE IDENTIFICATION OF UNSTABLE SYSTEMS

4.1 INTRODUCTION

Contents

4.1 Introduction 63
4.2 Basic definitions and assumptions 65
4.3 A subspace method for unstable systems 65
4.3.1 Issues in calculating B in unstable systems 66
4.3.2 Reorganizing the state space model of the process 67
4.3.3 Defining a stable predictor for the model 68
4.3.4 Computation of $\mathcal{A}(q)$ and $\mathcal{B}(q)$ 70
4.4 Case study 74
4.4.1 CSTR reactor and simulation generalities 75
4.4.2 Model identification results 77
4.4.3 Comparison between identified model based MPC 80
4.5 Conclusions 81

Open loop unstable (OLU) processes stabilized using a feedback controller are not so common in industrial practice, but a certain number of plants operates in this way, both for technical and economical reasons. The literature related to this kind of processes is poor, being only few works dealing with this topic. Furthermore, in papers dealing with identification algorithms, the capacity of handling data generated by an unstable system is usually unspecified.
This lack of information about the behavior of algorithms with data from unstable systems is not trivial, especially for subspace identification schemes. It will be showed in this chapter, actually, that a subspace method used for the identifications of stable plants is not guaranteed to work when tested on unstable processes.
On the other hand, prediction error methods suffer of weaker problems. It is sufficient that predictors are stable to guarantee consistent results in identification, and it can be showed that this feature depends mostly on kind of adopted model, as it possible to read in Ljung [20].

Examples of papers dealing with subspace identification for unstable system is represented by the one from Gabay et al. [8]. This work is quite old, so subspace identification methods were not well defined yet. For this reason, in this paper it is not pos-

sible to find a subspace method, but the algorithm that was presented is not too different from a SID one. Another paper is represented by the one from Moonen et al. [24]. Actually, this work does not focus the attention on unstable system identification, but it specifies that results are valid for OLU processes too.

Basis for identification of unstable systems via prediction error methods can be found in Ljung [20]. This book reports the main theoretical basis for this operation, and contains several demonstrations which guarantee the consistency of the estimates.
In Forssell & Ljung [7], an extension of these results was performed, in order to handle different types of models such as Output Error models and Box–Jenkins models, which could not be handled with the previous method. This extension considers that the unstable dynamic can be decomposed in a stable and in an unstable part and a new system based on this decomposition is introduced. This gives a stable predictor, so it is possible to identify it in an easy way. Forssell & Ljung also demonstrates that this introduced system converges asymptotically to the starting unstable system, obtaining in this way a consistent estimation of the unstable system parameters.

In the literature, even if it is not too much extended as previously said, different approaches to OLU identification can be found. In several papers, for example, systems to be identified are approximated with a continuous time plus time delay transfer function of the first or second order. The reason for this is the usability of obtained models for PI and PID tuning, which results increased in this way. This kind of approach cannot be used clearly for large MIMO systems, because of the low order of models.
This is for example the case of Marchetti et al. [22], which used ATV$^+$ identification technique, a particular kind of identification method based on system output analysis when marginal stability is imposed with a relay controller. This method gives as result a continuous time transfer function model with zero or first order Padè approximation of time delay. Another paper of this nature is the one from Ananth & Chidambaram [1], in which parameters are obtained from closed loop step responses. Using data coming from this configuration, a first order plus time delay model is constructed.

The aim of the work presented in this chapter is the definition of a subspace method that permits to obtain consistent models for unstable systems.
This chapter is organized as follows.

- In section 4.2 abbreviations and symbols used through the chapter are described.
- In section 4.3 theoretical basis of the proposed method are reported.
- In section 4.4 a practical example is tested.

- In section 4.5 conclusions are showed.

4.2 BASIC DEFINITIONS AND ASSUMPTIONS

In order to give the theoretical basis to the proposed identification method, a particular notation is introduced, and some assumption are presented. First of all, linear discrete time-invariant state-space systems in the following form are considered:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{4.1}$$

in which $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, $y \in \mathbb{R}^p$ is the output, $v \in \mathbb{R}^p$ is a stochastic noise, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{p\times n}$ are system matrices.
As said, some assumptions need to be introduced. In particular, the first one deals with stabilizability and observability of the system:

Assumption 4. *The pair* (A, B) *is stabilizable and the pair* (A, C) *is observable.*

Then, assumption 5 must hold for noise v in order the system can be identified using a subspace method:

Assumption 5. *The noise* v_k *is white, and statistically independent from past outputs and inputs, i. e.*$E\{y_k v_j'\} = 0$ *and* $E\{u_k v_j'\} = 0$ *for all* $j \geqslant k$.

Eventually, introducing the following assumption 6, the meaning of L and M indices is explained:

Assumption 6. *Data vectors* (u, y) *are collected for* $L = M + 2i - 1$ *sampling times, namely from sample time* 0 *to sample time* $L - 1$.

4.3 A SUBSPACE METHOD FOR UNSTABLE SYSTEMS

As previously stated in section 4.1, several identification methods can give sub-optimal results when tested on data generated by an unstable system. The reason for this behavior is purely computational, as it will be clear in the next paragraphs. In particular, it will be showed that subspace methods proposed in Pannocchia et al. [27] and reported in this work in chapter 1, and other similar identification methods lead very often to low-quality results. As it will be clear, the reason for this is the least squares approximation used in these methods to obtain B matrix.

In this work, a solution is proposed. A "step by step" least square problem is considered, which recalls the construction of a n-step ahead predictor, where n represents the order of the

system. This method recovers A and C matrices using the same subspace projection in Huang et al. [15] such as the method discussed in section 1.3. Experimental evidences have showed that this approach gives usually acceptable results even in case of data generated by an unstable system as it will be showed in case study section, so no modifications are needed for this purpose. Using the method introduced in this chapter, B is obtained adopting a novel regression on data, which does not suffer of numerical problems such as the classical least squares approach. This feature depends on its particular formulation, as it will be showed in section 4.3.1. From now on, $\hat{A}$ and $\hat{C}$ matrices, that is the estimations of A and C will be considered as known, calculated using the method introduced in chapter 1.

4.3.1 Issues in calculating B in unstable systems

Since A and C matrices do not present particular problems in recovering, even if data comes from unstable systems, B matrix estimation can be difficult. Several methods, such as the one presented in Pannocchia et al. [27] and reported in chapter 1 recover B matrix directly from data, performing a least square problem on predicted values of y_k.
In these works the predicted value of outputs, that is $\hat{y}_k$, is computed using a regression on data similar to the one in eq. 4.2

$$\hat{y}_k = \hat{C} \sum_{j=0}^{k-1} \hat{A}^j B u_j = f_k(B). \tag{4.2}$$

where $\hat{A}$ and $\hat{C}$ represent the estimation of real system matrices A and C. It can be noticed that in (4.2) a term $\hat{A}^j$ is present, and this explains why this kind of methods does not work with unstable systems. Supposing that $\hat{A}$ is a consistent estimate for A, it happens that being A unstable almost one of the eigenvalues of $\hat{A}$ is major than 1. The presence of that unstable eigenvalue leads to a rapid increase of $\hat{A}^j$ as j growth. For example, consider the $\hat{A} = A$ matrix reported in eq.4.3. Its eigenvalues, which can be easily obtained, are $\rho_1 = 1.17$ and $\rho_2 = 0.83$.

$$\hat{A} = A = \begin{bmatrix} 1.1 & 0.1 \\ 0.2 & 0.9 \end{bmatrix} \tag{4.3}$$

$\hat{A}^j$ assumes the values in table 6 when j increase: It is not rare that the maximum value of k in equation 4.2 is more than 1000, being k the number of available data. Such a situation leads clearly to numerical problems in least squares approximation, because a large number of terms in equation 4.2 become negligible (notice that in case $k = 1000$, a term $\hat{A}^{999}$ is present), and in particular the closest to time k. This situation is problematic because these occurrences are actually the ones that mostly influence the value y_k, as it is easy to understand.

Main idea presented in this work is to find $\hat{B}$, that is a consistent estimation of B, by defining a predictor based on an OE

Table 6: Values assumed by $\hat{A}^j$ with A from the (4.3) as j increases

j	$\hat{A}^j$
10	$\begin{bmatrix} 3.93 & 1.38 \\ 2.77 & 1.16 \end{bmatrix}$
50	$\begin{bmatrix} 2320.6 & 849.4 \\ 1698.8 & 621.8 \end{bmatrix}$
100	$\begin{bmatrix} 6.83 \cdot 10^6 & 2.5 \cdot 10^6 \\ 5 \cdot 10^6 & 1.83 \cdot 10^6 \end{bmatrix}$

model consistent with A and C matrices recovered using the method in section 1.3. This approach permits to bypass the mentioned issue for the presence of $\hat{A}^j$, as it will be showed in next sessions.

4.3.2 Reorganizing the state space model of the process

Consider the model in (4.1). Using the q operator, it is easy to reorganize the state equation of this formulation as in eq. 4.4

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ qIx_k &= Ax_k + Bu_k \\ (qI - A)\, x_k &= Bu_k \\ x_k &= (qI - A)^{-1} Bu_k \end{aligned} \tag{4.4}$$

Now, substituting this result in the outputs equation of the model, the (4.5) is obtained

$$y_k = C(qI - A)^{-1} Bu_k + v_k. \tag{4.5}$$

It is possible to see that in this equation A matrix appears with a power of 1. For this reason, if it were possible to use this equation for the calculation of B, it would not suffer for numerical problems unlike the methods previously mentioned.

For this reason, starting from (4.5), a suitable predictor needs to be defined in order to obtain the estimate $\hat{B}$. Being as previously stated $\hat{A}$ and $\hat{C}$ the estimates of process matrices obtained using the "standard" subspace method reported in chapter 1, it is easy to obtain the predictor in the (4.6), where the only unknown is B

$$\hat{y}_k = \hat{C}(qI - \hat{A})^{-1} Bu_k. \tag{4.6}$$

The presence in this expression of the term $(qI - A)^{-1}$ requires a particular attention. Indeed, it is not easy to use this predictor directly in an identification algorithm because of this term, which has not a simple analytic representation. For this reason, it must be converted in a useful expression for the calculation of B. In order to perform this task, recall the definition of inverse

matrix. Given a matrix $G \in \mathbb{R}^{q\times q}$, its inverse G^{-1} has the form reported in eq. 4.7

$$G^{-1} = \frac{\text{adj } G}{\det G} \tag{4.7}$$

where adj G is the adjugate matrix of G.
The adjugate matrix can be constructed in several steps, as indicated below:

- define the minor $\mathbf{M}_{ij}$ as the determinant of the matrix obtained removing the $i-th$ row and the $j-th$ column from matrix G
- define the cofactor matrix $\mathbf{C}$ elements as

$$\mathbf{C}_{ij} = -1^{i+j} M_{ij} \tag{4.8}$$

- find the adjugate matrix of G as the transpose of $\mathbf{C}$, that is adj $G = \mathbf{C}'$.

This definition confirms that it would be hard to construct an analytical expression for the adjugate matrix. However, it will be show that this difficulty can be bypassed using an indirect method for the calculation of adj G, such as the one described in this chapter.

4.3.3 Defining a stable predictor for the model

Combining the (4.5) and the (4.7), considering that $G = (qI - A)$, it is possible to obtain equation 4.9:

$$y_k = C\frac{\text{adj }(qI-A)}{\det(qI-A)}\hat{B}u_k + v_k \tag{4.9}$$

Now, analyze the term $\det(qI - A)$. It is a scalar equation in q (just recall that the determinant of a matrix is always a number), so it is possible to multiply both the sides of the equal sign for this term. Performing this,the (4.9) is transformed in eq. 4.10

$$\mathcal{A}(q)y_k = \mathcal{B}(q)u_k + \mathcal{A}(q)v_k \tag{4.10}$$

where it is straightforward to notice that

$$\begin{aligned}\mathcal{A}(q) &= \det(qI-A)\\ \mathcal{B}(q) &= C\text{adj }(qI-A)B\end{aligned} \tag{4.11}$$

At this point, a consideration must be done. Equation 4.10 presents a dynamic for the noise term v_k. This means that the noise contribution to the output value has not the characteristics of a white noise, but those of a colored noise (for the meaning of white and colored noise, see section 5.3.2).
For this reason, a least square method cannot be applied to eq. 4.10 without modifying its structure, because in this case the estimation $\hat{B}$ would not converge to the real value of B. One of the solutions could be to filter u_k through the dynamic of

$\mathcal{A}(q)^{-1}$ instead of multiplying both the sides of equal sign for $\mathcal{A}(q)$, obtaining in this way

$$y_k = \mathcal{B}(q)u_k^F + v_k \tag{4.12}$$

where $u_k^F = \mathcal{A}(q)^{-1}u_k$. Unfortunately, the instability of the filter $\mathcal{A}(q)$ does not permit this operation on u_k.

In order to address this problem, recalling equation 4.9 and using the variables $\mathcal{A}(q)$ and $\mathcal{B}(q)$ in equation 4.11 it is easy to obtain

$$y_k = \frac{\mathcal{B}(q)}{\mathcal{A}(q)}u_k + v_k \tag{4.13}$$

This represents an unstable Output Error model, for which an identification method can be found in the work by Forssell and Ljung [7]. This method starts from the decomposition of the unstable filter $\mathcal{A}(q)$ into two different parts

$$\mathcal{A}(q) = \mathcal{A}_s(q) \cdot \mathcal{A}_u(q) \tag{4.14}$$

$\mathcal{A}_s(q)$ and $\mathcal{A}_u(q)$ are respectively the stable and unstable part of the filter. Considering this decomposition, a new model can be defined, namely the following

$$y_k = \frac{\mathcal{B}(q)}{\mathcal{A}(q)}u_k + \frac{\mathcal{A}_u(q)^*}{\mathcal{A}_u(q)}v_k \tag{4.15}$$

where $\mathcal{A}_u(q)^*$ is the monic polynomial whose zeros are equal to the zeros of $\mathcal{A}_u(q)$ reflected into the unit disc, and so it is a stable filter. This filter has the following expression: defining $\mathcal{A}_u(q)$ as

$$\mathcal{A}_u(q) = 1 + a_1^u q^{-1} + \cdots + a_n^u q^{-n} \tag{4.16}$$

the filter $\mathcal{A}_u(q)^*$ is constructed as

$$\mathcal{A}_u(q)^* = 1 + \frac{a_1^u}{a_n^u}q^{-1} + \cdots + \frac{1}{a_n^u}q^{-n} \tag{4.17}$$

Forssell and Ljung demonstrated under mild conditions that the estimates of the model in the (4.13) and of the one in the (4.15) converge asymptotically to the same value when they are identified using a prediction error method.

The importance in defining the model in equation 4.15 is that $\mathcal{A}_u(q)^*$ is stable, and for this reason the predictor for this system is stable.

Actually, the predictor formulation results

$$\begin{aligned}\hat{y}_k &= \frac{\mathcal{B}(q)\mathcal{A}_u(q)}{\mathcal{A}(q)\mathcal{A}_u(q)^*}u_t + \left(1 - \frac{\mathcal{A}_u(q)}{\mathcal{A}_u(q)^*}\right)y_k = \\ &= \frac{\mathcal{B}(q)}{\mathcal{A}_s(q)\mathcal{A}_u(q)^*}u_k + \left(1 - \frac{\mathcal{A}_u(q)}{\mathcal{A}_u(q)^*}\right)y_k\end{aligned} \tag{4.18}$$

which represents clearly a stable predictor, being both $\mathcal{A}_s(q)$ and $\mathcal{A}_u(q)^*$ stable filters.

At this point, a stable predictor for the system is defined. As it is possible to see, it depends on $\mathcal{A}(q)$ and $\mathcal{B}(q)$ structures, which on the contrary were not defined yet. For this reason, next step will be the definition of a computation method for both these two.

4.3.4 Computation of $\mathcal{A}(q)$ and $\mathcal{B}(q)$

As it was showed, eq. 4.13 was obtained with a simple reorganization of the model in equation 4.1.
Actually, this operation is no more than a formal change, because it does not give any additional information on the calculation of B in practice. This is related to the fact that the structure of $\mathcal{A}(q)$ and $\mathcal{B}(q)$ has not been analyzed accurately till now, and so no method for the definition of their known part (namely the one which can be calculated starting from $\hat{A}$ and $\hat{C}$) has been presented. This task will be performed in this section.

Obtaining $\mathcal{A}(q)$

The first structure that will be analyzed is the structure of $\mathcal{A}(q)$, because its definition is straightforward. From equation (4.11) it is known that $\mathcal{A}(q) = \det(qI - A)$, that is $\mathcal{A}(q)$ is the characteristic polynomial of A matrix.
It is obvious that, being $\hat{A}$ the estimation of the real A matrix, in practice it will result $\mathcal{A}(q) = \det(qI - \hat{A})$.
So, being $\hat{A}$ note, and in particular being note its eigenvalues, it is possible to obtain the expression in (4.19)

$$\mathcal{A}(q) = \det\left(qI - \hat{A}\right) = \prod_{j=1}^{n}(q - \lambda_j) = \sum_{j=0}^{n} \alpha_{n-j} q^{n-j} \tag{4.19}$$

where λ_j are the eigenvalues of A and α_{n-j} the coefficients of the corresponding filter. Finally, n is the order of the system.

Obtaining $\mathcal{B}(q)$

As previously stated, it is not simple to derive an analytical expression for $\mathcal{B}(q)$ starting from its formal definition, so a different approach is needed. For this purpose, eq. (4.13) can be considered. Once $\mathcal{B}(q)$ is found, its value can be used to construct the new system in equation 4.15 and perform a linear regression. Looking to equation 4.13, it appears very similar to a TF expression. For this reason, express the system in eq. 4.1 in a general TF form. It results eq. 4.20

$$y_k = F(q)u_k + v_k = \begin{bmatrix} F_{1,1}(q) & \dots & F_{1,m}(q) \\ \vdots & \ddots & \vdots \\ F_{p,1}(q) & \dots & F_{p,m}(q) \end{bmatrix} u_k + v_k. \tag{4.20}$$

Everyone of the $F_{r,s}$ in (4.20) has the standard TF form reported in (4.21), where $p_j^{r,s}$ and $z_j^{r,s}$ correspond to the value of poles and zeros of the various system.

$$F_{r,s} = \frac{\prod_{j=1}^{l}(q - z_j^{r,s})}{\prod_{j=1}^{n}(q - p_j^{r,s})}. \tag{4.21}$$

At this point, consider that deriving (4.20) from (4.1), the poles are the same for everyone of the $F_{r,s}$. This statement can be expressed with equation 4.22

$$p_j^{r,s} = p_j \qquad \forall\, r, s. \tag{4.22}$$

In particular, for the same reason, the poles are equal to the eigenvalues of A, as it is reported in equation 4.23

$$p_j = \lambda_j \tag{4.23}$$

with λ_j from eq. 4.19. Now it is easy to see that, holding the (4.23), the (4.24) can be derived

$$\prod_{j=1}^{n}(q - p_j) = \prod_{j=1}^{n}(q - \lambda_j) = \mathcal{A}(q). \tag{4.24}$$

As it was expected, the first result is clear. The dynamic of the system depends only on $\mathcal{A}(q)$ and consequently on $\hat{A}$, from which $\mathcal{A}(q)$ derives directly. So, rewrite the (4.20) as the (4.25)

$$\begin{aligned} \hat{y}_k &= F(q)u_k + v_k = \\ &= \frac{\begin{bmatrix} O_{11}(q) & \dots & O_{1m}(q) \\ \vdots & \ddots & \vdots \\ O_{p1}(q) & \dots & O_{pm}(q) \end{bmatrix}}{\mathcal{A}(q)} u_k + v_k \end{aligned} \tag{4.25}$$

where $F_{ij} = \frac{O_{ij}}{\mathcal{A}(q)}$.
Multiplying both the sides of this equation by $\mathcal{A}(q)$, it is easy to obtain the eq. 4.26

$$\begin{aligned} \mathcal{A}(q)\hat{y}_k &= \begin{bmatrix} O_{11}(q) & \dots & O_{1m}(q) \\ \vdots & \ddots & \vdots \\ O_{p1}(q) & \dots & O_{pm}(q) \end{bmatrix} u_k + \mathcal{A}(q)v_k \\ &= \mathbf{O}(q)u_k + \mathcal{A}(q)v_k \end{aligned} \tag{4.26}$$

from which it results that

$$\mathbf{O}(q) = \mathcal{B}(q) \tag{4.27}$$

from a simple comparison with equation 4.10. This is an important results, because it is possible to say that $\mathcal{B}(q)$ is the matrix of numerators of the system expressed in the form of transfer function.

The task now is to recover $\hat{B}$ matrix from the (4.27). Using the Kronecker product, it is possible to rewrite the term $\mathbf{O}(q)u_k$ as in the (4.28)

$$\text{Vec}\, \mathcal{B}(q)u_k = (u_k' \otimes I_p)\text{Vec}\, \mathcal{B}(q) \tag{4.28}$$

Looking to $\mathcal{B}(q)$ in (4.11), it is clear that its expression is linear in $\hat{B}$. So, it is possible to differentiate that term w.r.t. the elements of $\hat{B}$ and rearrange, obtaining the (4.29)

$$\text{Vec}\, \mathcal{B}(q)u_k = (u_k' \otimes I_p)J^{\mathcal{B}}\text{Vec}\, B, \tag{4.29}$$

where

$$J^{\mathcal{B}}_{(r,s)} = \frac{\partial(\text{Vec } \mathcal{B}(q))_r}{\partial(\text{Vec } B)_s}. \tag{4.30}$$

Next steps to be performed are straightforward. Consider at this point, once $\mathcal{B}(q)$ is defined, the system in equation 4.15. Such as in every prediction error method, the predictor can be used is a linear regression on data in order to minimize the mismatch between real and predicted value for the system, that is $y_k - \hat{y}_k$. The expression for this difference in case of the model of equation 4.15 is reported in the (4.31).

$$y_k - \hat{y}_k = -\frac{\mathcal{B}(q)}{\mathcal{A}_s(q)\mathcal{A}_u(q)^*}u_k + \left(\frac{\mathcal{A}_u(q)}{\mathcal{A}_u(q)^*}\right) y_k \tag{4.31}$$

Furthermore, it is clear that in the right side of equation 4.31 the only unknown is represented by Vec B. This is because $\mathcal{B}(q)$ depends on this term, while y_k, u_k, $\mathcal{A}(q)$, $\mathcal{A}_u(q)$, $\mathcal{A}_s(q)$ and $\mathcal{A}_u(q)^*$ are known.

For this reason, this equation can be used for estimating Vec B. The (4.31) can be expressed in vectorial form considering the totality of available sampling times, and it results

$$\begin{aligned} \boldsymbol{e_n} &= -\left[(u_k^{Fn})' \otimes I_p\right] J^{\mathcal{B}} \text{Vec } B + \boldsymbol{y_n} = \\ &= -\Phi_n \text{Vec } B + \boldsymbol{y_n} \end{aligned} \tag{4.32}$$

where

$$\begin{aligned} \boldsymbol{e_n} &= \begin{bmatrix} y_0 - \hat{y}_0 \\ y_1 - \hat{y}_1 \\ \vdots \\ y_{L-1} - \hat{y}_{L-1} \end{bmatrix} \quad \boldsymbol{y_n} = \begin{bmatrix} y_0^{Fn} \\ y_1^{Fn} \\ \vdots \\ y_{L-1}^{Fn} \end{bmatrix} \\ \Phi_n &= \begin{bmatrix} \left](u_0^{Fn})' \otimes I_p\right] J^{\mathcal{B}} \\ \left[(u_1^{Fn})' \otimes I_p\right] J^{\mathcal{B}} \\ \vdots \\ \left[(u_{L-1}^{Fn})' \otimes I_p\right] J^{\mathcal{B}} \end{bmatrix} \end{aligned} \tag{4.33}$$

Φ_n and $\boldsymbol{y_n}$ can be computed from the $\mathcal{A}(q)$ and $\mathcal{B}(q)$ matrices and from the filtered data sequence y^{Fn}. y^{Fn} derives from the filtration of the data sequence y through the stable filter given by $(\mathcal{A}_u(q)^*)^{-1}$ and $\mathcal{A}_u(q)$, that is

$$y^{Fn} = \frac{\mathcal{A}_u(q)}{\mathcal{A}_u(q)^*}y \tag{4.34}$$

Similarly, it is possible to define the data sequence u^{Fn} using the filters $(\mathcal{A}_u(q)^*)^{-1}$ and $\mathcal{A}_s(q)^{-1}$

$$u^{Fn} = \frac{1}{\mathcal{A}_u(q)^*\mathcal{A}_s(q)}u \tag{4.35}$$

Finally, $\mathcal{A}(q)$ and $\mathcal{B}(q)$ values can be obtained using the same methods previously introduced for the case of absence of noise, because as it is possible to notice, they are the same of equation 4.13.

At this point, the operation to be performed in order to find the estimate Vec B and consequently the estimate $\hat{B}$ is

$$\hat{\text{Vec}}\, B = \arg\min_{\text{Vec } B} (\mathbf{y_n} - \Phi_n \text{Vec } B)' (\mathbf{y_n} - \Phi_n \text{Vec } B) \tag{4.36}$$

which solution is clearly given by

$$\hat{\text{Vec}}\, B = \Phi_n^+ \mathbf{y_n} = \left(\Phi_n' \Phi_n\right)^{-1} \Phi_n' \mathbf{y_n} \tag{4.37}$$

In this way, a consistent estimation of the matrix B is obtained. Unfortunately, no methods for the calculation of $J^{\mathcal{B}}(q)$ were introduced till now, so the last task is finding a way to obtain an expression for this one.

From $\mathcal{B}(q)(q)$ to Vec B

Equation 4.29 gives the analytical solution for the problem of recovering Vec B from $\mathcal{B}(q)$, but it is hard to apply it in practice because of the difficulties in expressing $J^{\mathcal{B}}(q)$.
For this reason, a shortcut procedure was developed for completing this task: recalling the (4.26), it is clear that everyone of the $\mathbf{O}_{r,s}(q)$ is a function of B terms.
Consider now that $y_k = \hat{y}_k + v_k$, or equivalently $\hat{y}_k = y_k - v_k$. Then ,define a new system, described in equation 4.38

$$\begin{aligned} \hat{x}_{k+1}^{r,s} &= A\hat{x}_k^{r,s} + E^{r,s} b^{r,s} u_k \\ \hat{y}_k^{r,s} &= C\hat{x}_k \end{aligned} \tag{4.38}$$

where $b^{r,s} = B(r,s)$.
$E_{r,s} \in \mathbb{R}^{n \times p}$ is a matrix defined as in the (4.39)

$$E^{r,s}(\bar{r},\bar{s}) = \begin{cases} 1 & \bar{r} = r,\ \bar{s} = s \\ 0 & \bar{r} \neq r,\ \bar{s} \neq s \end{cases} \tag{4.39}$$

From the latter equation it is clear that

$$B = \sum_{r=1}^{n} \sum_{s=1}^{m} E^{r,s} b^{r,s}. \tag{4.40}$$

Using the previous results, it is possible to demonstrate[1] the (4.41) and consequently the (4.42).

$$\hat{x}_k = \sum_{r=1}^{n} \sum_{s=1}^{m} \hat{x}_k^{r,s} \tag{4.41}$$

$$\hat{y}_k = \sum_{r=1}^{n} \sum_{s=1}^{m} y^{r,s} = C \sum_{r=1}^{n} \sum_{s=1}^{m} \hat{x}_k^{r,s} \tag{4.42}$$

In particular, the latter says that $\hat{y}_k$ is equal to the sum of the $n \cdot m$ models that generate the states sequences $\hat{x}^{r,s}$.
Using the latter result, equation 4.38 and the (4.40) it is possible

1 See the appendix at the end of this chapter

to obtain the (4.44), which shows clearly the separate contributions of the terms of B.

$$\hat{y}_k^{r,s} = C(qI - A)^{-1}E^{r,s}b^{r,s}u_k = \frac{\mathbf{O}^{r,s}(q)b^{r,s}}{\mathcal{A}(q)}u_k \tag{4.43}$$

In this equation, $\mathbf{O}^{r,s}(q)$ represents the matrix of numerators of the transfer function for every single system given by the matrices A, $E^{r,s}$ and C. A simple reorganization leads to

$$\mathcal{A}(q)\hat{y}_k^{r,s} = \mathbf{O}^{r,s}(q)b^{r,s}u_k = u_k' \otimes I_p \operatorname{Vec} \mathbf{O}^{r,s}b^{r,s} \tag{4.44}$$

Now, recall that $\hat{y}_k = y_k - v_k$ and apply the result in (4.44) to the (4.42)

$$\mathcal{A}(q)y_k = \left(u_k' \otimes I_p\right)\begin{bmatrix}\operatorname{Vec}\mathbf{O}^{1,1}(q) & \operatorname{Vec}\mathbf{O}^{2,1}(q) & \dots & \operatorname{Vec}\mathbf{O}^{p,m}(q)\end{bmatrix}\begin{bmatrix}b^{1,1}\\ b^{2,1}\\ \vdots \\ b^{p,m}\end{bmatrix} + \\ + \mathcal{A}(q)v_k$$

It is easy to see that in this equation there is the presence of a term equivalent to that in the (4.29), and so an useful expression for calculating $J^{\mathcal{B}}$ is obtained. From this and eq. (4.29), indeed, it results the (4.45)

$$J^{\mathcal{B}} = \begin{bmatrix}\operatorname{Vec}\mathbf{O}^{1,1}(q) & \operatorname{Vec}\mathbf{O}^{2,1}(q) & \dots & \operatorname{Vec}\mathbf{O}^{p,m}(q)\end{bmatrix} \tag{4.45}$$

So, the computation of $J^{\mathcal{B}}$ can be performed by the definition of the systems in the 4.38 in a TF form. In the end, few steps are needed to obtain an expression for the calculation of B:

- define a series of $n \cdot m$ state space models with matrices $\hat{A}$, $E^{r,s}$ and $\hat{C}$ with $E^{r,s}$ from the (4.39);
- find for everyone of them the correspondent TF expression;
- construct $J^{\mathcal{B}}$ as in the (4.45) using the numerators of the various TF.
- solve a least square problem to find Vec $\hat{B}$, the estimate of Vec B

4.4 CASE STUDY

In this section, a testing case study is reported. For this example, three different subspace algorithms were tested, that is the one presented in this chapter for unstable systems identification, the original projection method by Huang et al. [15] and N4SID algorithm in the formulation implemented in MATLAB. Method presented in Pannocchia et al. [27] was not compared with the others because of the numerical problems it presents when used on unstable system data.

Table 7: CVs, MVs and outputs for CSTR process of figure 29

	Variables
CV	T, h
MV	T_c, F
Outputs	c, T, h

4.4.1 CSTR reactor and simulation generalities

Proposed algorithm was tested on an unstable process, namely the one showed in figure 29. It is a stirred CSTR reactor, with CVs and MVs reported in table 7.

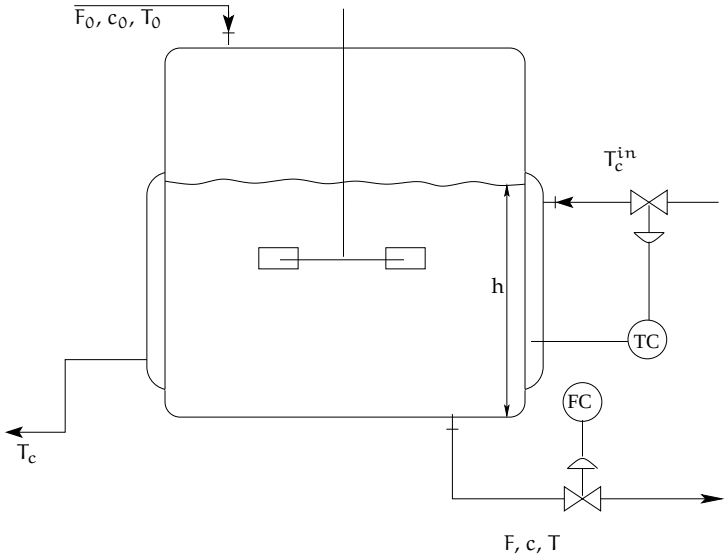

Figure 29: CSTR reactor

From the curves of heat and material balance for that reactor, it is possible to find the equilibrium states of the system.
It results that this system has three different equilibrium states. The curves are plotted in figure 30 while the value of process outputs at the various equilibrium states are reported in table 8

Table 8: Equilibrium points values for CSTR heat-material balance of figure 30

Eq. point	c[mol/l]	T[°C]	h[m]	stability
# 1	2.92	306.5	0.372	stable
# 2	1.68	355	0.372	unstable
# 3	0.175	414	0.372	unstable

For this analysis, the values for point #3 equilibrium state is selected.

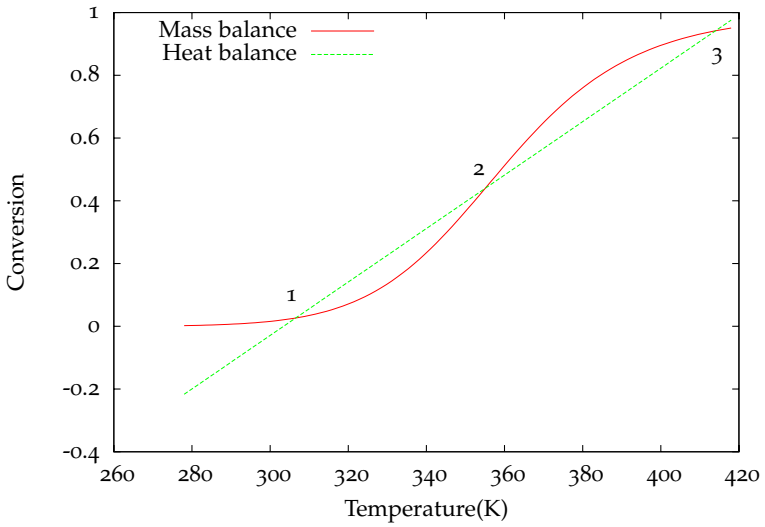

Figure 30: CSTR heat-material balance

Real system matrices for the system at this equilibrium point assume the following values

$$
\begin{aligned}
A &= \begin{bmatrix} 0.43 & -0.052 & 0.39 \\ -3.36 & 1.11 & -3.12 \\ 0 & 0 & 1 \end{bmatrix}, \\
B &= \begin{bmatrix} -17.13 & -0.02 \\ 116.1 & 0.71 \\ -82.71 & 0 \end{bmatrix}, \\
C &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\end{aligned}
\tag{4.46}
$$

This system has been used as real process in order to generate data for identification. A CL scheme for the process has been designed using an MPC algorithm. This controller was based on the following "erroneous" model, obtained from an identification on a data set which presented a very high noise to ratio level.

$$
\begin{aligned}
A &= \begin{bmatrix} 0.918 & 0.23 & 0.21 \\ 0.058 & 0.9 & -0.12 \\ -0.065 & -0.35 & 0.94 \end{bmatrix}, \\
B &= \begin{bmatrix} -108.2 & -4.1 \\ 160.6 & 2.9 \\ -329.2 & -4.9 \end{bmatrix}, \\
C &= \begin{bmatrix} -0.0023 & 0.071 & 0.034 \\ -0.42 & 0.53 & 0.4 \\ -0.01 & -0.019 & -0.005 \end{bmatrix}.
\end{aligned}
\tag{4.47}
$$

An output disturbance model for MPC structure was considered during data collection.

Results will be showed using two different approaches. As previously said, the proposed method derives from Huang et

al. [15], and permits to recover A matrix (that is the poles of the system) and simultaneously C matrix from the observability matrix of the system. So, for system dynamics identification, a Monte Carlo simulation will be performed in order to demonstrate that system poles value can be estimated consistently using the method adopted in this chapter. Furthermore, it will be showed that the obtained values for poles are better w.r.t. the ones coming from an identification performed using N4SID method. Finally, for poles identification, it will be showed that the quality of the estimates increases as the number of data increases using the proposed method. Even in this case, a comparison is made with poles values coming from a N4SID identification.

In the second approach for presenting results, the overall model evaluation of various methods is checked. It will be showed that numerical problems will not affect the proposed algorithm in performing this operation, and it will be showed that a model obtained using the approach introduced in this chapter results usually better than models obtained with other kinds of algorithms.
For this purpose, it is necessary to define a β parameter to measure the quality of obtained models. This is necessary because the particular system that has been considered presents an integrator, so it is not possible to compare the steady state gains. In case of the presence of an integrator in the system, indeed, the system steady state gain assumes an infinite value, so this parameter cannot be used. So, gains coming from the system are compared in a frequency limited domain, in order not to reach the steady state value. $\beta(i\omega)$ is defined as follows

$$\beta(\omega) = \left(\hat{G}(e^{i\omega}) - G(e^{i\omega})\right) ./ G(e^{i\omega}) \tag{4.48}$$

where G is the real system transfer function and $\hat{G}$ in the identified system transfer function. Using a MATLAB notation, the operator ./ indicates the element by element division, that is, given two matrices $W \in \mathbb{R}^{t\times s}$ and $V \in \mathbb{R}^{t\times s}$,

$$W./V = \begin{bmatrix} \frac{w_{1,1}}{v_{1,1}} & \cdots & \frac{w_{1,s}}{v_{1,s}} \\ \vdots & \ddots & \vdots \\ \frac{w_{t,1}}{v_{t,1}} & \cdots & \frac{w_{t,s}}{v_{t,s}} \end{bmatrix} \tag{4.49}$$

$\beta(\omega)$ is then a measure of gains difference between identified model and real process at each frequency. It is easy to see that, the closest the gain to the real value, the closest β to 0. Results for the proposed method will be compared with the ones coming from N4SID and standard orthogonal projection algorithm presented in Huang et al. [15].

4.4.2 Model identification results

The first operation that was performed was the recovering of poles from data. As clear, this operation shows how well the

proposed method can recover the value of A matrix from data sequence.
For this operation, a Monte-Carlo simulation was performed on 100 different data sequences, and the results are showed in figure 31. In this figure, proposed method and orthogonal projection method are reported in the same plot, because their approach to pole calculation is exactly the same. A gbn set-point changes sequence on outputs #2 and #3 is sent to the system for the generation of the data sequence. Figure 31 shows clearly

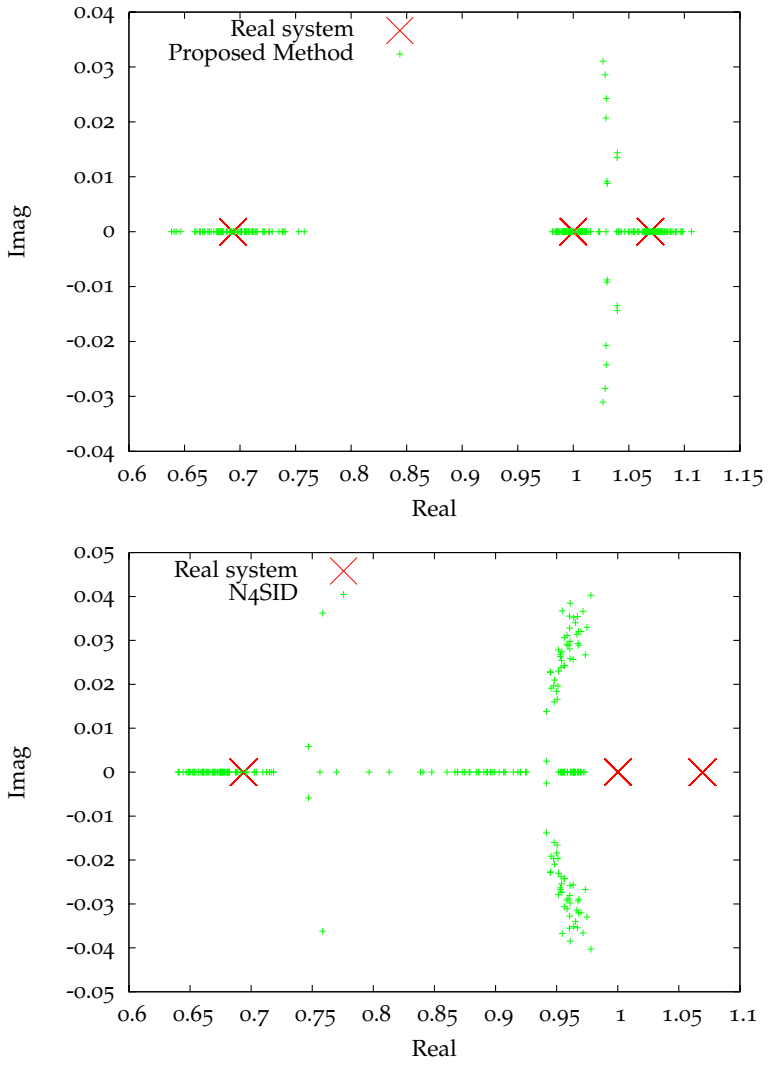

Figure 31: Poles values for system in eq. 4.46 for the proposed method and projection method (top) and for N4SID identification method (bottom)

that proposed method and projection method work better than N4SID for pole recovering.

Furthermore, it is interesting to investigate if the value of system poles converge to the real value when the number of data increase. For this reason, figure 32 is reported. In this figure, the continuous line represents the real value of poles, the

dashed line represents the proposed (and the orthogonal projection method) while the dot-dashed line represents the N4SID method in MATLAB implementation. It is possible to see that, except for some edges due to numerical matters in the bottom figure, the estimates of the pole converge to the real values as the number of samples increases.

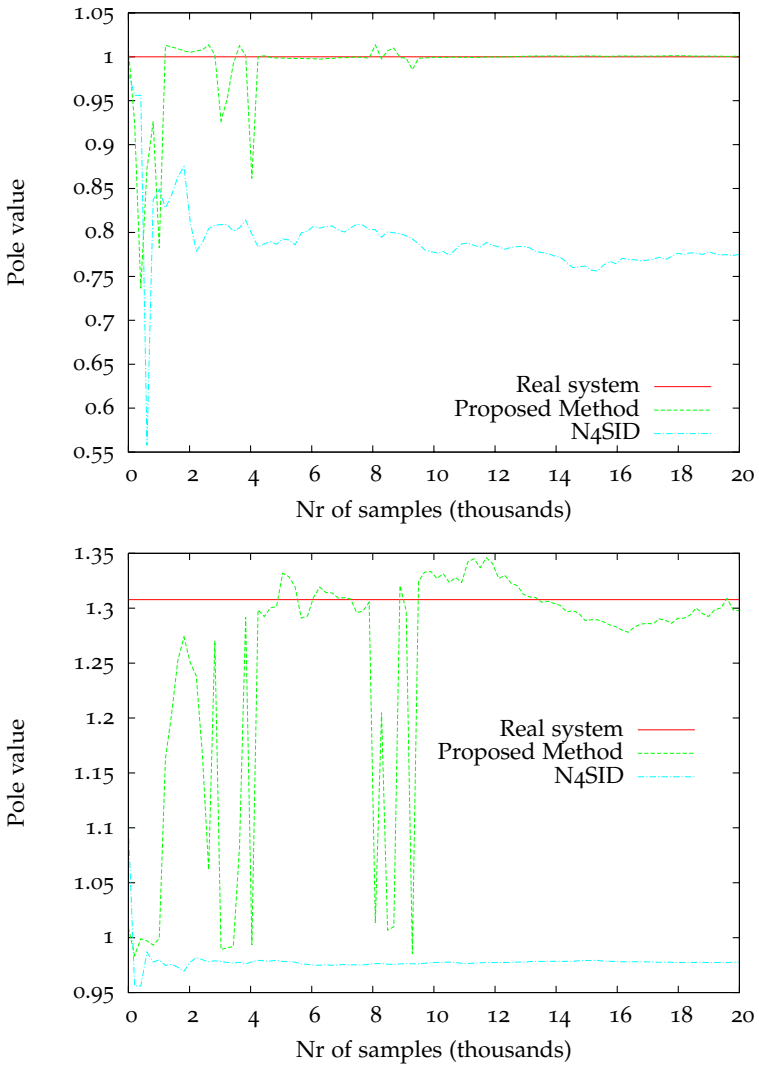

Figure 32: Poles convergence for system in eq. 4.46 for increasing number of data. #3 equilibrium point

As previously showed, the overall quality of identified models is compared using β parameter introduced in equation 4.48. A data set of 2000 sampling times was defined using a closed loop scheme, then the three different identification methods were tested on these data. β was calculated for the obtained models, and the values it assumed are reported in table 9. Noise magnitude for these data is about the 10% of the whole output signal magnitude in every channel.

Table 9: $\beta(\omega)$ values for proposed method(left), projection method(center), and N4SID(right)

	Proposed Method		Projection Method		N4SID	
	u_1	u_2	u_1	u_2	u_1	u_2
y_1	0.125	0.26	0.77	0.73	15.65	77.4
y_2	0.177	0.15	0.69	4.04	2.62	1.21
y_3	0.085	0.02	0.83	0.95	1.52	0.33

Looking at table 9, it results clear that the proposed method gives better results than the other methods in terms of overall model quality.

4.4.3 Comparison between identified model based MPC

Consider now a noisy data set for identification, generated using the model of equation 4.47. Matrices in table 10 were used for MPC tuning.

Table 10: MPC tuning matrices used to collect data for identification

Q	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
R	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
S	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
H_y	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Measurement noise covariance is equal to

$$\text{cov}\,(v) = 0.01I \tag{4.50}$$

and a GBN set-points sequence of magnitude 1 on outputs #2 and #3 is considered for identification.
Data were collected for 3000 sampling time, and several models were identified using a part of the whole data sequence. In particular, four models were identified, using respectively the first 1500, 2000, 2500 and 3000 occurrences of data matrices. Then, four different MPC based on these models were designed, and their performance in terms of SP following are measured.

This was done both making a visual comparison of the plots and considering the value of the cost function

$$\Phi = \frac{1}{2}\left[(y_k - y_s)'Q(y_k - y_s) + (u_k - u_{k-1})'S(u_k - u_{k-1})\right] \tag{4.51}$$

which represents a simplified version of the MPC dynamic problem.

In figure 33 the response to a SP step variation of the identified models is reported. A -0.5 step is sent to output #3 at sampling time 15, and a 0.5 step is sent to output #2 at time 30. For this comparison, non noise was used in order to show better the behavior of models. It is possible to see that, obviously, the more the data used for identification, the better the model in terms of "similarity" of its response to that of the true model based MPC.

In order to measure this "similarity", values of ϕ parameter in equation 4.51 for the various models are reported in table 11

Table 11: Values of Φ parameter for the MPC systems of figure 33

model	Φ value
True Process	0.869
1500 Samples	0.94
2000 Samples	0.9
2500 Samples	0.876
3000 Samples	0.874

It is easy to see that the model obtained using the 3000 data samples is better than the others using this approach too, as it was expected. Indeed, it represents the closest value to that of the MPC based on real process.

4.5 CONCLUSIONS

In this chapter, a subspace method for unstable system identification was presented. The method appears to work with data coming from OLU systems better than standard subspace identification methods such as N4SID, original projection method by Huang et al. [15] and method in Pannocchia et al. [27].

The method is based on the method by Huang et al. [15] for the calculation of the system dynamics, that is A matrix (and consequently C matrix). On the other hand B matrix is obtained from data using a particular regression that does not contains high powers of the unstable A matrix. Using the approach in Forssell and Ljung [7], a system with a stable predictor is defined, which converges to the value of the real system as data occurrences tends to infinity. This new system permits to filter u and y data using a stable filter. In such a way, a consistent estimate of B matrix can be obtained performing a regression on filtered data matrices.

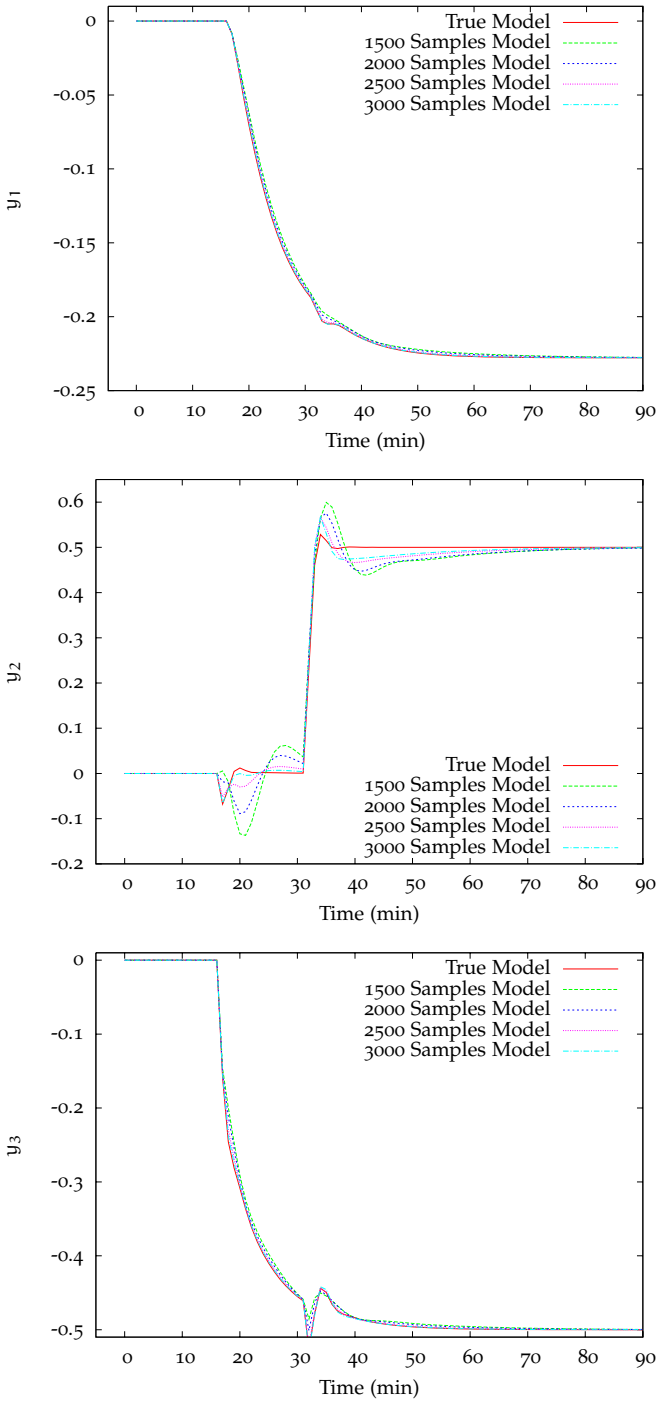

Figure 33: Step response for MPC based on identified models.

APPENDIX TO CHAPTER 4

In this appendix, a proof of the (4.41) is presented. Starting from the (4.1), it results that

$$x_k = \sum_{j=1}^{k} A^{j-1} B u_{k-j}$$

Consider now the (4.38). From this one, it result easily that

$$x_k^{r,s} = \sum_{j=1}^{k} A^{j-1} E^{r,s} b^{r,s} u_{k-j}.$$

Now, substitute the latter expression in the (4.41). Doing this, it results

$$x_k = \sum_{r=1}^{n} \sum_{s=1}^{m} \sum_{j=1}^{k} A^{j-1} E^{r,s} b^{r,s} u_{k-j} = \\ = \sum_{j=1}^{k} A^{j-1} \sum_{r=1}^{n} \sum_{s=1}^{m} E^{r,s} b^{r,s} u_{k-j} .$$

But for the 4.40 $B = \sum_{r=1}^{n} \sum_{s=1}^{m} E^{r,s} b^{r,s}$, so

$$x_k = \sum_{j=1}^{k} A^{j-1} B u_{k-j}$$

which demonstrates the thesis.

5 MPC MONITORING USING PREDICTION ERROR SEQUENCE

Contents

5.1 Introduction **85**
5.2 Generalities and notation **87**
5.2.1 Real system 87
5.2.2 MPC process model 87
5.2.3 Prediction error definition 88
5.3 Prediction Error Analysis **89**
5.3.1 Autocorrelation in a time series 89
5.3.2 Definition of θ index 92
5.3.3 Finding a generating process for the PE sequence 94
5.3.4 Definition of the PE generating process 95
5.3.5 Analysis of the observability of the PE generating system 97
5.4 Obtaining the observability matrix order from data **99**
5.4.1 Summary of the proposed monitoring method 99
5.5 Simulation example **100**
5.5.1 Generalities on simulations 101
5.5.2 Results in case of no mismatches 104
5.5.3 Mismatch in the process model 107
5.6 Conclusions **110**

5.1 INTRODUCTION

MPC algorithm, as introduced in the previous chapters of this thesis, is one of the most used optimization scheme in industrial life for driving chemical plants. As it is possible to read in the survey by Qin & Badgwell [30], especially during the last years there has been a large diffusion of this kind of controller, with many new applications.
MPC technology has been developed widely in the last twenty years, from several point of view. Actually, in the whole MPC theory, the field that has gained less attention is represented by performance monitoring field and related problems. This is an important lack, because a well designed monitoring technique permits to get the best possible results from an MPC, through correcting its design specifications in case of sub-optimal performances.

Actually, several papers dealing with the topic of MPC monitoring were presented. For example, a paper dealing with these

subjects is the one from Loquasto & Seborg [21]. In that one, the authors perform a pattern recognition of data using a PCA approach. This analysis is based on the idea that data coming from a plant which is not working in optimal conditions can be divided in several classes, in dependence of the problem that caused the loss of performance. In other words, if data coming from a system are analyzed using particular approaches such as the one described in the mentioned work, they belong to different pattern in case the system works in nominal (optimal) conditions or it is affected by some kind of mismatch.
Data sets are assigned to different patterns using some statistical properties: in case of the Loquasto & Seborg paper, for example, a PCA approach is used and correspondent T^2 and Q parameter are introduced. In this way, it results that it is possible to discern between the different sources of problems in case of performance degradation in the plant.

Argawal et al. [2] proposed a probabilistic approach to the MPC performance assessment based on the analysis of constraints. As a related result of this approach, the authors also introduced some rules for MPC tuning which derives directly by the considerations made on constraints. Their work is particularly interesting because they focused on the maximum achievable performance using MPC scheme. Actually, this is not the first example of paper introducing this concept, because it can be found in several works before. One of these works is for example the paper of Pathwardan & Shah [29], which deals with the most important limiting conditions for MPC performance, such as model uncertainty, non-linearities, sampling time etc..

Recently, another paper dealing with performance monitoring was presented by Yu & Qin [39]. It contains a statistical approach to the problem of performance monitoring, based on a data-driven covariance analysis. In other words, process performances are measured using a performance index which considers the actual variation of data with respect to the variation of a standard set. This standard set is a set of data collected during a period in which the performance of the controller can be considered satisfactory.

This chapter presents a way to assess the optimality of the control action of an MPC scheme working with a Kalman filter, and to discern the cause of low performances if present. With respect to the methods previously introduced, it deals with the analysis of the prediction error sequence, which will be introduced in the following chapters.
This method is based on a work by Harrison [12]. In that work, the problem of MPC monitoring was studied and a result was given in case the MPC action is linear and no set-point changes are given to the system. Unfortunately, MPC action is not linear in general, e. g. for the presence of bounds, and in addiction SP changes are not rare. These limits make the method in [12] useful only for certain types of data. In the present chapter, an

extension is performed, and a method to avoid those restriction is proposed.

This chapter is organized as follows:

- in section 5.2 notation and assumptions are introduced. Prediction error is also defined;
- in section 5.3 prediction error analysis theory is described;
- in section 5.4 selected method for recovering the observability matrix from data is introduced;
- in section 5.5 a case study for parameter estimation and method application is showed;
- in section 5.6 conclusion for this chapter are reported.

5.2 GENERALITIES AND NOTATION

In this chapter, MPC monitoring is performed analyzing the statistical properties of prediction error sequence.
Before presenting the proposed method, some basic concepts must be introduced. In particular, the expressions for describing real system and MPC model are introduced, and the prediction error is defined.

5.2.1 Real system

In this analysis, the real process is represented by a discrete time linear state space system. With respect to the standard model introduced in chapter 1, in this chapter the model is completed with B_d and C_d matrices as in equation 5.1. This expansion of the model permits to describe the case in which a disturbance is entering into the system.

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_d d_k + w_k \\ y_k &= Cx_k + C_d d_k + v_k \end{aligned} \tag{5.1}$$

In this equation, $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^m$ is the input, $y_k \in \mathbb{R}^p$ is the output, $d_k \in \mathbb{R}^p$ is a disturbance entering into the system. (A, B, C) represent the model of the process, while B_d and C_d are the real disturbance model. In the end, $w_k \in \mathbb{R}^n$ and $v_k \in \mathbb{R}^p$ are stochastic noises. For these two, the following assumption holds:

Assumption 7. *$w_k \in \mathbb{R}^n$ and $v_k \in \mathbb{R}^p$ are random white noises with normal distribution.*

5.2.2 MPC process model

Process model used in MPC is introduced in (5.2). It is an extended model, that is it includes a model for a certain number

of fictitious integral disturbances. This extension is required in order to guarantee the absence of offset on the controlled variables. As stated by Rawlings & Pannocchia[28], offset-free control in an MPC scheme can be achieved using as many fictitious disturbances as outputs, that is p in the notation introduced in this work. Roughly speaking, fictitious disturbances are additional states which "measure" the off-set in variables and use this value in the definition of the control action, giving an integral action to the controller.

As already said, in (5.2) the model used in MPC is presented

$$\begin{aligned} \begin{bmatrix} \hat{x}_{k+1|k} \\ \hat{d}_{k+1|k} \end{bmatrix} &= \begin{bmatrix} \hat{A} & \hat{B}_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_{k|k} \\ \hat{d}_{k|k} \end{bmatrix} + \begin{bmatrix} \hat{B} \\ 0 \end{bmatrix} u_k \\ \hat{y}_k &= \begin{bmatrix} \hat{C} & \hat{C}_d \end{bmatrix} \begin{bmatrix} \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} \end{aligned} \tag{5.2}$$

In this equation $\hat{x} \in \mathbb{R}^n$ is the calculated state, $\hat{d}_k \in \mathbb{R}^p$ is the integrating disturbance. $(\hat{A}, \hat{B}, \hat{C})$ are the system model matrices, $(\hat{B}_d, \hat{C}_d)$ are the disturbance model matrices. Note that dim $\{x_k\}$ = dim $\{\hat{x}_k\}$, that is the real process and the model have the same order.
The meaning of double indexes (like $k|k$) was already introduced in chapter 2: the left index indicates the time at which the value is predicted, while the right one indicates the time at which it is filtered using the real output value and the calculated one, such as in equation 5.3:

$$\begin{bmatrix} \hat{x}_{k|k} \\ \hat{d}_{k|k} \end{bmatrix} = \begin{bmatrix} \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} \begin{bmatrix} L_x \\ L_d \end{bmatrix} (y_k - \hat{y}_k) \tag{5.3}$$

where (L_x, L_d) is the Kalman filter for the process in eq.5.2.

5.2.3 Prediction error definition

Given the expressions for real system and process model, it is possible to give a definition of prediction error:

Definition 1. *Prediction error e_k is the difference at sampling time k between the real output value y_k and the output value $\hat{y}_k$ "predicted" by the model used in MPC , that is*

$$e_k = y_k - \hat{y}_k. \tag{5.4}$$

Following this definition, prediction error expression can be derived directly from equation 5.1 and equation 5.2, simply substituting these equations in the (5.4)

$$e_k = Cx_k + C_d d_k - \hat{C}\hat{x}_{k|k} - \hat{C_d}\hat{d}_{k|k} + v_k \tag{5.5}$$

Now, recall the filtering step for states and disturbances performed inside MPC. This is the operation that refreshes the values $\hat{x}_{k|k-1}$ and $\hat{d}_{k|k-1}$ giving the values $\hat{x}_{k|k}$ and $\hat{d}_{k|k}$, which was introduced in chapter 2 and is reported in equation 5.3 . It

is straightforward that this filtering can be expressed using the prediction error:

$$\begin{bmatrix} \hat{x}_{k|k} \\ \hat{d}_{k|k} \end{bmatrix} = \begin{bmatrix} \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} + \begin{bmatrix} L_x \\ L_d \end{bmatrix} e_k \tag{5.6}$$

where $\begin{bmatrix} L_x \\ L_d \end{bmatrix}$ is the Kalman filter for the process, as previously stated.

5.3 PREDICTION ERROR ANALYSIS

It has been stated that prediction error at time k is the difference between the real value of process outputs at that sampling time and the values predicted by the model used inside MPC at time $k-1$. Analyzing this definition, it is obvious that every problem that can arise in the controller and which can deviate the MPC action from the optimality has an influence on prediction error, because $\hat{y}_k$ and consequently e_k value depends directly on the process model and on the process filter.
This suggests that prediction error can be a useful parameter to measure the behavior of the controller. The main problem to be addressed is that prediction error is generated by different contributions, as it is possible to see in eq. 5.5. Indeed, it depends on real states x_k and real disturbances d_k, but also model states $\hat{x}_{k|k}$ and fictitious disturbances $\hat{d}_{k|k}$, and eventually on process and measurement noise, that is respectively w_k and v_k. For that complexity, it can be difficult to discern which one is the element that generates problems.
In the following section, a method is proposed to complete this task. First, an index which permits to recognize the presence of issues in the control system via a statistic analysis of prediction error is proposed, then a way for individuating the cause of low performances in case problems are found is suggested.

5.3.1 Autocorrelation in a time series

One of the methods that can show the presence of issues in control system is the analysis of autocorrelation for prediction error sequence. Indeed, the following property holds:

Remark 1. *If an MPC scheme works in optimal conditions, that is process matrices are exactly known and the Kalman filter associated to the system is optimal, the prediction error autocorrelation parameter is represented by a white noise sequence.*

This statement means that, checking the autocorrelation "whiteness" for the prediction error sequence of a system, it is possible to decide if that system works in optimal or in sub-optimal conditions.
Autocorrelation parameter is based on the autocovariance of a

system: given a scalar variable collected for N sampling times, that is $\epsilon \in \mathbb{R}^{1\times N}$, the definition of autocovariance for ϵ is reported in eq. 5.7

$$\mathcal{R}^{\epsilon}(\tau) = E\left[\epsilon_k \epsilon'_{k-\tau}\right] \tag{5.7}$$

Normalizing by the value $\mathcal{R}^{\epsilon}(0)$ (that is the covariance of ϵ) it is possible to obtain the autocorrelation parameter $\mathbf{r}(\tau)$

$$\mathbf{r}(\tau) = \frac{\mathcal{R}^{\epsilon}(\tau)}{\mathcal{R}^{\epsilon}(0)} = \frac{E[\epsilon_k \epsilon'_{k-\tau}]}{E[\epsilon_k \epsilon'_k]} \tag{5.8}$$

This definition can be extended to the case in which ϵ is not a scalar variable but a vector, that is $\epsilon \in \mathbb{R}^{p\times N}$ with p representing the number of outputs. This can be done simply considering a channel at time, that is the autocorrelation of the $i-th$ channel of ϵ, namely ϵ^i, is represented by

$$\mathbf{r}^i(\tau) = \frac{E[\epsilon^i_k \left(\epsilon^i_{k-\tau}\right)']}{E[\epsilon^i_k \left(\epsilon^i_k\right)']} \tag{5.9}$$

In case of several channels, everyone of them will be take into account in the definition of autocorrelation. During the following chapter, for simplicity of exposition, all the expression will be referred to a SISO system where not specified. Results will also hold for the MIMO case as well.

As introduced before, the task to be performed is to check the whiteness of autocorrelation sequence. Being $\mathbf{r}(\tau)$ a stochastic variable, this can be done defining a confidence interval for $\mathbf{r}(\tau)$ and checking for $\mathbf{r}(\tau)$ to respect it.
$\mathcal{F}$ confidence interval of a stochastic variable represents the band that, statistically, contains the $\mathcal{F}$% of the occurrences of that variable. This band is calculated around the expected value. Roughly speaking, defining in case of no autocorrelation

$$\bar{\mathbf{r}}(\tau) = E[\mathbf{r}(\tau)] \tag{5.10}$$

as the expected value for $\mathbf{r}(\tau)$, confidence interval is represented by

$$\bar{\mathbf{r}}(\tau) \pm \varepsilon_{\mathcal{F}}$$

where $\varepsilon_{\mathcal{F}}$ is the confidence bound (or confidence limit).
$\varepsilon_{\mathcal{F}}$ has the following expression:

$$\varepsilon_{\mathcal{F}} = \frac{\iota_{\mathcal{F}}}{\sqrt{N}} \tag{5.11}$$

where N represents the number of data of ϵ, and $\iota_{\mathcal{F}}$ derives directly from the cumulative function of the distribution of ϵ. In this work, the $\mathcal{F} = 95$ confidence limit was considered: having assumed that ϵ follows a Gaussian distribution, ι_{95} has a value of 1.96, that is

$$\varepsilon_{95} = \frac{\iota_{95}}{\sqrt{N}} = \frac{1.96}{\sqrt{N}} \tag{5.12}$$

An example is reported to explain this concept. In figure 35 two different noise sequences are plotted. On the left, there is a standard white noise sequence, while on the right there is a colored

Figure 34: Generation of a colored noise sequence

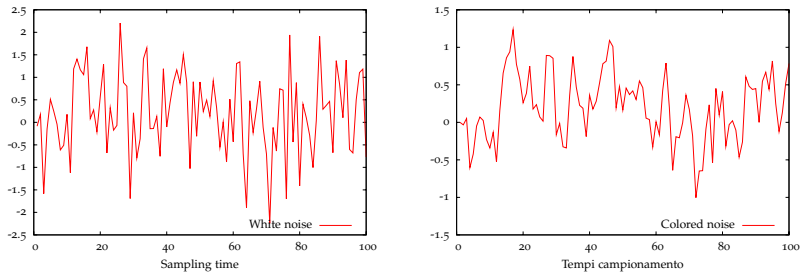

Figure 35: Plots of a white noise sequence (left) and of a colored noise sequence (right)

noise sequence. A colored noise is the result of the filtration of a white noise through a dynamic block, as shown in figure 34. As a consequence of this filtration, colored noise sequence presents autocorrelation because everyone of its occurrences depends on the previous ones.
Analyzing figure 35, it is clear that is very hard to distinguish between the two different kinds of noises simply basing on their time plots. Performing the calculation of autocorrelation for $\tau = 0 \rightarrow \tau_{max}$ on these two sequences, figure 36 is obtained, in which $\tau_{max} = 100$ was chosen. In both the graphs of this figure, the dashed lines represent the confidence bounds.
From this figure it is possible to see that, for the white noise sequence, the limits are respected (the 95% of the occurrences of the variable are inside the confidence region), while in the case of colored noise sequence this is not true.
Furthermore, in the second case there are high deviations from the limits, so it results that the colored noise sequence presents autocorrelation.

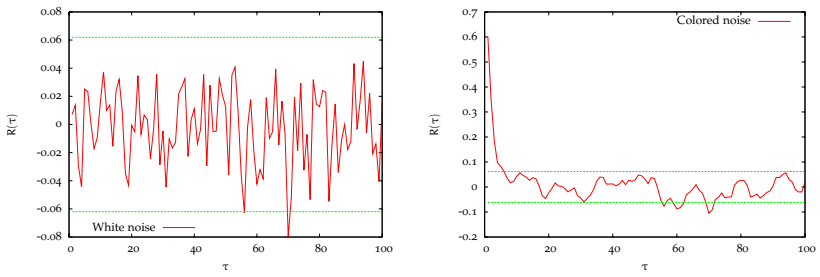

Figure 36: Autocorrelation plots for a white noise sequence (left) and for a colored noise sequence (right)

5.3.2 Definition of θ index

As previously stated, the presence of autocorrelation can be noticed from the autocorrelation sequence plot. However, this operation has a major issue, that is it lacks of objectivity. Indeed, the use of a simple check on the plot does not guarantee an exact rule to discern between the case of autocorrelation presence and the case of no autocorrelation presence. For this reason, different people could take different decisions for the same plot.
Furthermore, the direct analysis of the plot is an operation that presents problems to be performed in automatic, so the presence of an operator is required.

To address these problems, in this work an index θ is introduced. Its formulation, reported in eq. 5.13, is such that the system is considered affected by autocorrelation if $\theta > 1$.

$$\theta = \frac{1}{2}\left(\frac{n^{oc}}{n^{oc}_{max}} + \frac{oc^{max}}{oc^{max}_{lim}}\right) \tag{5.13}$$

θ is composed by two different parts, that is $\frac{n^{oc}}{n^{oc}_{max}}$ and $\frac{oc^{max}}{oc^{max}_{lim}}$. The first one takes into account the number of times the autocorrelation sequence violates the 95% bounds, while the second one considers the magnitude of violation.
Both of these terms are analyzed separately:

$\frac{n^{oc}}{n^{oc}_{max}}$: n^{oc} represents the number of violations of the limits in the sequence, that is the number of star symbols (∗) in figure 37, while n^{oc}_{max} is the maximum admissible number for them. n^{oc}_{max} must be set by the user. It should be $0.05\tau_{max}$ to respect the limit in case $\tau_{max} \to \infty$, but for the limited number of points in the autocorrelation sequence (in this work, $\tau_{max} = 100$ was chosen as previously said, and so statistic properties could be affected) it is suggested to consider a larger value.
A standard selection, that has showed acceptable results during this experience, is $n^{oc}_{max} = 0.07\tau_{max}$ (and then $n^{oc}_{max} = 7$ in case $\tau_{max} = 100$).Considerations that have brought to this choice will be analyzed later in the chapter.

$\frac{oc^{max}}{oc^{max}_{lim}}$: oc^{max} is the absolute value of the largest violation from the confidence bounds, evidenced in figure 37, while oc^{max}_{lim} is the maximum admissible value for that violation. Experimental evidences have demonstrated that a value $oc^{max}_{lim} = 1.5\varepsilon_{95}$ gives acceptable results, as it will be stated in the next session.

From this definition, it results directly that in case $n^{oc}_{max} = n^{oc}$ and $oc^{max} = oc^{max}_{lim}$, $\theta = \frac{1}{2}(1+1) = 1$. So, this is considered as the limit condition for which the method suggests there is no autocorrelation in the system, that is

$$\begin{cases} \theta \leqslant 1 & \to \quad \text{no autocorrelation} \\ \theta > 1 & \to \quad \text{autocorrelation} \end{cases} \tag{5.14}$$

In table 12 the values for θ and the various parameters used in θ definition are reported for both the sequences of figure 36. It is possible to see that autocorrelation is correctly found only in the colored noise sequence.

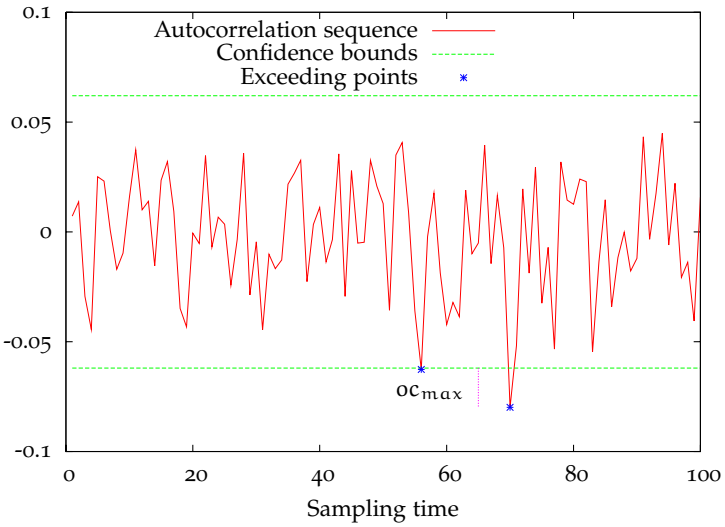

Figure 37: Graphical representation of the parameters for the calculation of θ

Table 12: θ value for autocorrelation sequences in figure 36

	White noise	Colored noise
n^{oc}	2	13
n^{oc}_{max}	7	7
oc^{max}	0.08	0.6
oc^{max}_{lim}	0.093	0.093
θ	0.57	4.15
Results	**no autocorrelation**	**autocorrelation**

As already introduced, the guidelines adopted for selecting the value of n^{oc}_{max} and oc^{max}_{lim} are reported extensively in the following sections. For now, it is sufficient to say that two Monte-Carlo simulations were performed in order to find the most acceptable values for those variables. For further details, it is recommended to refer to section 5.5. The system reported in equation 5.31 will be used for this purpose.

In case of several channels, it is considered that prediction error sequence is autocorrelated when the mean value of θ for the various channels is higher than 1. So, for example, if $\theta^1 =$

0.9 and $\theta^2 = 1.2$, the mean value is 1.05 and so the system is considered autocorrelated.

5.3.3 Finding a generating process for the PE sequence

As previously said, θ index gives the possibility of discovering the presence of autocorrelation in a time series. At this point, another question arises: if θ shows autocorrelation, which is the cause that generates this autocorrelation?
In general, it is common to consider three different causes for suboptimal control (and consequently for autocorrelation in prediction error sequence):

Model mismatch: model adopted inside MPC scheme and for the definition of the Kalman filter for the process is different from the real system, that is $A, B, C \neq \hat{A}, \hat{B}, \hat{C}$;

Noise covariance mismatch: covariances matrices used for the calculation of the Kalman filter do not describe the real noise covariance which is present in the system, so the filter results suboptimal;

Entering disturbances: one or more unmeasured disturbance are entering into the system, causing a sudden and unexpected variation of the real system states and output that cannot be predicted by the MPC, resulting in a suboptimal control action.

Actually, the last two causes have the same solution, that is the redefinition of the noise covariance matrices for the system, so it is not important to discern between them.
Harrison [12] has analyzed this problem in the past, and he has introduced a method for recognize which one of these three causes of low performance affects the controller. His methods is based on the analysis of the rank of the observability matrix of the system that generates the prediction error. This system derives from a reorganization of the process equation, as it will be showed later in this chapter.

The whole method of Harrison is not reported here,but an analysis of his results can be done. On the first, it results clear that Harrison's work has several interesting points, such as it is usually faster than other kind of methods. Furthermore, its structure gives the possibility of considering simple on-line applications for that method, which represents probably one of its most interesting feature.
On the other hand, Harrison's method suffers for several limitations, in particular due to its basilar hypothesis, that is the linearity of MPC action. Actually, the control action generated by an MPC is piece-wise linear, i. e.it is linear for each different combination of active constraints. Therefore, whenever the set of active constraints changes, the resulting control action is not linear. On the contrary, Harrison's hypothesis can be considered valid only in case the set of active constraints remains the same

in time, a condition which is not respected in many cases.
Another problem about Harrison's method is that it is not valid in case a set-point change is sent to the system, because this possibility was not considered in the definition of the algorithm.

Both of these issues are not trivial, because they introduce limitations on data that can be analyzed using this method. For this reason, an accurate data selection should be performed before using the method, with a consequent waste of time. Moving from this issues, an extension for Harrison's method will be proposed in this work in order to handle a wider class of data, possibly every data coming from the plant, without loosing the interesting features of Harrison's method.

5.3.4 Definition of the PE generating process

As previously said, Harrison's method is based on the analysis of the observability matrix of the system that generates the PE sequence. In other and simpler words, "the system that generates the PE sequence" is the one for which the PE sequence represents the output. The proposed method moves from a similar consideration, so even in this case it is needed to define that model. The difference is that, in case of the proposed method, no consideration will be made on the linearity of control action, and the presence of set-point changes can also be possible. For this last point, indeed, Harrison introduced linearity in control action and developed the u_k control action term, but he did not consider that it depends also on set-point values, avoiding in this way the usage of his method in case set-point changes are present.

The expression of the desired model can be obtained with a reorganization of eq. 5.1 and eq. 5.2. These two are synthesized in a unique model in eq. 5.15:

$$\begin{aligned}
\begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1|k} \\ \hat{d}_{k+1|k} \end{bmatrix} &= \begin{bmatrix} A & 0 & 0 \\ 0 & \hat{A} & \hat{B}_d \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_{k|k} \\ \hat{d}_{k|k} \end{bmatrix} + \begin{bmatrix} B \\ \hat{B} \\ 0 \end{bmatrix} u_k + \\
&+ \begin{bmatrix} I & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_k \\ v_k \end{bmatrix} + \begin{bmatrix} B_d \\ 0 \\ 0 \end{bmatrix} d_k \\
e_k &= \begin{bmatrix} C & -\hat{C} & -\hat{C}_d \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} + \\
&+ \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} w_k \\ v_k \end{bmatrix} + C_d d_k
\end{aligned} \tag{5.15}$$

This model comes doing some simple mathematics: it is easy to notice that its states are composed of both the real system and the process model states. For this reason, its order $\tilde{n}$ is given by the following expression:

$$\begin{aligned}
\tilde{n} &= \text{real sys. ord.} + \text{proc. mod. ord.} + \#\text{ of fict. dist.} = \\
&= n + n + p = 2n + p
\end{aligned} \tag{5.16}$$

$\hat{x}_{k|k}$ and $\hat{d}_{k|k}$ in equation 5.15 must be expanded, because they depend on the value e_k as it results from eq. 5.6. This leads to

$$\begin{aligned}
\begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1|k} \\ \hat{d}_{k+1|k} \end{bmatrix} &= \begin{bmatrix} A & 0 & 0 \\ 0 & \hat{A} & \hat{B}_d \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} + \begin{bmatrix} B \\ \hat{B} \\ 0 \end{bmatrix} u_k + \\
&+ \begin{bmatrix} 0 \\ \hat{L}_x \\ L_d \end{bmatrix} e_k + \begin{bmatrix} I & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_k \\ v_k \end{bmatrix} + \begin{bmatrix} B_d \\ 0 \\ 0 \end{bmatrix} d_k \\
e_k &= \begin{bmatrix} C & -\hat{C} & -\hat{C}_d \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} + \\
&+ \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} w_k \\ v_k \end{bmatrix} + C_d d_k
\end{aligned} \tag{5.17}$$

where $\hat{L}_x = AL_x + B_d L_d$. This system is not in the standard state-space form, because of the e_k term, which represents the output variable of the model, appears in the state equation too, as it is easy to notice.
In particular, the "filtering part" of that equation must be reorganized, namely the one in the (5.18)

$$\begin{bmatrix} 0 \\ \hat{L}_x \\ L_d \end{bmatrix} e_k \tag{5.18}$$

Substituting in this equation the value of e_k from the (5.17), it results the (5.19)

$$\begin{aligned}
\begin{bmatrix} 0 \\ \hat{L}_x \\ L_d \end{bmatrix} e_k &= \begin{bmatrix} 0 & 0 & 0 \\ \hat{L}_x C & -\hat{L}_x \hat{C} & -\hat{L}_x \hat{C}_d \\ L_d C & -L_d \hat{C} & -L_d \hat{C}_d \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} + \\
&+ \begin{bmatrix} 0 & 0 \\ 0 & \hat{L}_x \\ 0 & L_d \end{bmatrix} \begin{bmatrix} w_k \\ v_k \end{bmatrix} + \begin{bmatrix} 0 \\ \hat{L}_x C_d \\ L_d C_d \end{bmatrix} d_k
\end{aligned} \tag{5.19}$$

Now it is simple to substitute the (5.19) in the (5.17), in order to obtain the desired model. In this way, the formulation in eq. 5.20 is obtained:

$$\begin{aligned}
\tilde{x}_{k+1} &= \tilde{A}\tilde{x}_k + \tilde{B}u_k + \tilde{G}\tilde{w}_k + \tilde{B}_d d_k \\
e_k &= \tilde{C}\tilde{x}_k + \tilde{D}\tilde{\omega}_k + \tilde{C}_d d_k
\end{aligned} \tag{5.20}$$

The meaning of the matrices in (5.20) is reported below

$$\tilde{A} = \begin{bmatrix} A & 0 & 0 \\ \hat{L}_x C & \hat{A} - \hat{L}_x \hat{C} & \hat{B}_d - \hat{L}_x \hat{C}_d \\ L_d C & -L_d \hat{C} & I - L_d \hat{C}_d \end{bmatrix}$$

$$\tilde{B} = \begin{bmatrix} B \\ \hat{B} \\ 0 \end{bmatrix}$$
$$\tilde{G} = \begin{bmatrix} I & 0 \\ 0 & \hat{L}_x \\ 0 & L_d \end{bmatrix}$$
$$\tilde{B}_d = \begin{bmatrix} B_d \\ \hat{L}_x C_d \\ L_d C_d \end{bmatrix}$$
$$\tilde{C} = \begin{bmatrix} C & -\hat{C} & -\hat{C}_d \end{bmatrix}$$
$$\tilde{D} = \begin{bmatrix} 0 & I \end{bmatrix} \quad \tilde{C}_d = C_d$$
$$\tilde{x}_k = \begin{bmatrix} x_k \\ \hat{x}_{k|k-1} \\ \hat{d}_{k|k-1} \end{bmatrix} \quad \tilde{\omega}_k = \begin{bmatrix} w_k \\ v_k \end{bmatrix}$$

At this point, the model generating the PE is completely defined. In next section, the rank of the observability matrix of this system will be analyzed and some consideration will be made about its value.

5.3.5 Analysis of the observability of the PE generating system

Observability matrix expression for a standard state space system was introduced in chapter 1. Now, recalling that definition, it is possible to say that, starting from $\tilde{A}$ and $\tilde{C}$, the observability matrix of the system in the (5.20) has the form

$$O = \begin{bmatrix} \tilde{C} \\ \tilde{C}\tilde{A} \\ \dots \\ \tilde{C}\tilde{A}^{\tilde{n}-1} \end{bmatrix} \tag{5.21}$$

where $\tilde{n}$ is the order of the system, that is

$$\tilde{n} = 2n + p.$$

as previously showed.

Now, analyze the block $\tilde{C}\tilde{A}$. It is possible to substitute both the expression for $\tilde{A}$ and $\tilde{C}$ in its expression, resulting in the (5.22)

$$\tilde{C}\tilde{A} = \begin{bmatrix} \tilde{C}A_{11} & \tilde{C}A_{12} & \tilde{C}A_{13} \end{bmatrix} \tag{5.22}$$

where the blocks $\tilde{C}A_{11}$, $\tilde{C}A_{12}$ and $\tilde{C}A_{13}$ can be expanded as

$$\begin{aligned} \tilde{C}A_{11} &= CA - \hat{C}\hat{L}_x C - \hat{C}_d L_d C \\ \tilde{C}A_{12} &= -\hat{C}\hat{A} + \hat{C}\hat{L}_x\hat{C} + \hat{C}_d L_d \hat{C} \\ \tilde{C}A_{13} &= -\hat{C}\hat{B}_d + \hat{C}\hat{L}_x\hat{C}_d - \hat{C}_d + \hat{C}_d L_d \hat{C}_d \end{aligned} \tag{5.23}$$

Now, suppose to be in case in which there is no model mismatch $(\hat{A}, \hat{B}, \hat{C}) = (A, B, C)$. For this assumption, terms in eq. (5.23) become

$$\begin{aligned}
\tilde{CA}_{11} &= CA - C\hat{L}_x C - \hat{C}_d L_d C \\
\tilde{CA}_{12} &= -CA + C\hat{L}_x C + \hat{C}_d L_d C \\
\tilde{CA}_{13} &= -C\hat{B}_d + C\hat{L}_x \hat{C}_d - \hat{C}_d + \hat{C}_d L_d \hat{C}_d
\end{aligned} \tag{5.24}$$

Analyzing the blocks $\tilde{CA}_{11}$ and $\tilde{CA}_{21}$ in equation (5.24) it is clear that $\tilde{CA}_{21} = -\tilde{CA}_{11}$, so

$$\begin{aligned}
\tilde{C}\tilde{A} &= \begin{bmatrix} \tilde{CA}_{11} & \tilde{CA}_{12} & \tilde{CA}_{13} \end{bmatrix} = \\
&= \begin{bmatrix} \tilde{CA}_{11} & -\tilde{CA}_{11} & \tilde{CA}_{13} \end{bmatrix}
\end{aligned} \tag{5.25}$$

In the same way, it is possible to calculate the following blocks of the observability matrix, that is

$$\tilde{C}\tilde{A}^2 = \tilde{C}\tilde{A} \cdot \tilde{A},$$

$$\tilde{C}\tilde{A}^3 = \tilde{C}\tilde{A}^2 \cdot \tilde{A}$$

and so on.
Analyzing now the term $\tilde{C}\tilde{A}^2$, the following expression results:

$$\tilde{C}\tilde{A}^2 = \begin{bmatrix} \tilde{CA}_{21} & \tilde{CA}_{22} & \tilde{CA}_{23} \end{bmatrix}. \tag{5.26}$$

The blocks of this matrix can be expanded in this way:

$$\begin{aligned}
\tilde{CA}_{21} &= \tilde{CA}_{11} A - \tilde{CA}_{11} \hat{L}_x C - \tilde{CA}_{13} L_d C \\
\tilde{CA}_{22} &= -\tilde{CA}_{11} A + \tilde{CA}_{11} \hat{L}_x C + \tilde{CA}_{13} L_d C \\
\tilde{CA}_{23} &= -\tilde{CA}_{11} \hat{B}_d + \tilde{CA}_{11} \hat{L}_x \hat{C}_d - \tilde{CA}_{13} + \tilde{CA}_{13} L_d \hat{C}_d
\end{aligned} \tag{5.27}$$

or equivalently

$$\begin{aligned}
\tilde{C}\tilde{A}^2 &= \begin{bmatrix} \tilde{CA}_{21} & \tilde{CA}_{22} & \tilde{CA}_{23} \end{bmatrix} \\
&= \begin{bmatrix} \tilde{CA}_{21} & -\tilde{CA}_{21} & \tilde{CA}_{23} \end{bmatrix}
\end{aligned} \tag{5.28}$$

It is easy to verify that, substituting in the same way the following elements of the observability matrix, this assumes the formulation in eq. (5.29)

$$O = \begin{bmatrix} \tilde{CA}_{11} & -\tilde{CA}_{11} & -\tilde{CA}_{13} \\ \tilde{CA}_{21} & -\tilde{CA}_{21} & \tilde{CA}_{23} \\ \tilde{CA}_{31} & -\tilde{CA}_{31} & \tilde{CA}_{33} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{5.29}$$

This equation shows that in case the model of the process used for the definition of the MPC scheme and of the Kalman filter have no mismatches with the real system, the first two block columns are linearly dependent, and so the order of this matrix is not the maximum (that is $\tilde{n}$). Namely, in case there is no model mismatch, the order is $n + p$, while if a model mismatch is present, the order of the whole system is greater than $n + p$ (at most $2 \cdot n + p$).

5.4 OBTAINING THE OBSERVABILITY MATRIX ORDER FROM DATA

It has been already showed in chapter 1 that the first step of a typical subspace algorithm is the calculation of observability matrix from data sequence. This is a particularly interesting feature in this case, because it coincides with the next step required for prediction error analysis. So, it results that subspace methods (and consequently the one in section 1.4.2 for this case) are particularly suitable for this kind of analysis.

It was stated that the reported subspace algorithm has two different formulation, in case the data come from a CL or an OL system. Referring to equation (5.20), a consideration in that sense can be done. This is a CL system, because the output (the prediction error) is used for the calculation of the input at the following step, so an appropriate projection matrix must be set for identification with SID method.

Once the observability matrix is obtained, the objective is searching for its rank. Even in this case, the previously introduced subspace method is useful: indeed, this method computes the rank of that matrix to recover the order of the system.
Recalling the formulation in chapter 1, a method which can be used in order to find its rank is checking the singular values $\sigma_1, \sigma_2, \dots$ of that matrix and selecting them following the rule

$$\frac{\sigma_q}{\sum_{g=1}^{q} \sigma_g} < \delta \tag{5.30}$$

where the rank of the matrix is equal to the smallest q for which 5.30 holds. δ is a user defined threshold parameter.

Harrison's method suggests to use an AIC criterion for identifying this order, but it does not seem actually the best choice. This is because AIC requires a computation time that can be long in case the system has a large number of states, inputs and outputs.

5.4.1 Summary of the proposed monitoring method

In the end, a possible way for monitoring and MPC scheme is the following:

1. calculate the θ index as in (5.13);
2. if $\theta > 1$, calculate the order $\tilde{n}$ of the observability matrix from data using the method previously mentioned;
3. if $\tilde{n}$ is less or equal to $n + p$ the supposed covariance matrices for w and v are wrong, so new ones must be calculated from the data. If $\tilde{n}$ is equal or larger than $n + p$, then the model of the system used in the MPC is not valid, and a new identification is required for the process.

This method seems to work well with low order systems, while for high order system it is common to experience problems. In

particular, it was seen during some identification session for the definition of the system order, that the method takes a long time in order to identify a model. A possible solution can be found modifying the previously reported algorithm, to limit the dimension of the system to be identified.

As a first step, a consideration on singular values has to be made. In a high order system, for numerical reasons, it is possible that the identified model order has a certain variation, for the effect of noise in data.
For this reason, it could be useful not to consider exactly the order $n+p$ as the decision limit for the system, but it is better to define a region around the value $n+p$ in which the system could be considered affected by an error in the definition of the noise matrices even if $\tilde{n} > n+p$. This region can be considered a function of system order n, defining the parameter $\gamma = \text{round}(0.1 \cdot \tilde{n})$, in MATLAB notation, as a threshold parameter. So, the system should be considered affected by noise if the identified order is higher than $n+p+\gamma$.

Once this step is defined, consider equation 1.32 of chapter 1. In this equation, identification matrices are constructed using an r parameter which should be equal to system order or higher. This parameter has an important influence on the time required for system identification. The higher r in eq. 1.32, the longer the time required for identification. Then, take a $r = n+p+2\cdot\gamma$, in order to define a value for r close to the maximum admissible value in case of mismatch in noise covariance matrices. If the system identified with this value for r has an order higher than $n+p+\gamma$, it is possible to say that the process that has generated the analyzed prediction error sequence is affected by a model mismatch, or if the order is equal to or lower than $n+p+\gamma$ the system is affected by noise covariance matrices mismatch.

Such an implementation permits to limit at most the value for r, giving a faster computation time, and on the same time gives a variable value to the system order $\tilde{n}$ which can be less restrictive for the definition of the problem affecting the system.

5.5 SIMULATION EXAMPLE

The proposed method was tested on a 2×2 system introduced in the next section. Three cases were investigated:

- the case of no mistakes in process or noise covariance matrices definition
- the case of mistakes in the definition of noise covariance matrices and consequently in the Kalman filter
- the case of mistakes in the model used for the definition of both the system and the filter

In this chapter, results for this system will be presented and discussed.

5.5.1 Generalities on simulations

The model adopted for the verification of the method is reported in equation 5.31 in a state space form

$$\begin{aligned} A &= \begin{bmatrix} 0.8 & 0 \\ 0 & 0.84 \end{bmatrix} \\ B &= \begin{bmatrix} 0.4 & 0.3 \\ -0.2 & 0.4 \end{bmatrix} \\ C &= \begin{bmatrix} 0.1 & 0.3 \\ -0.3 & 0.2 \end{bmatrix} \end{aligned} \tag{5.31}$$

In order to show that the method works even in case of set point changes on the output variables, two set point changes of magnitude -0.2 and 0.7 were introduced respectively at time $k = 1500$ on the first output and at time $k = 1000$ on the second output. The model adopted in the MPC algorithm is extended using an output disturbance model.
Covariance matrices $Q_w = \begin{bmatrix} 4 \cdot 10^{-2} I & 0 \\ 0 & 4 \cdot 10^{-4} I \end{bmatrix}$ and $R_v = 4 \cdot 10^{-2} I$ has been used respectively for the definition of process noise and measurement noise of the extended model. Data were collected for $N = 3000$ sampling times.

Next section will be dedicated to the determination of satisfactory values for n_{oc} and oc_{max}^{lim} parameters used in the definition of θ, via a Monte-Carlo simulation.
After that, three different cases for the process in the (5.31) will be examined. In the first one, both the process and the covariances matrices are exactly known, so the control action is optimal. In this case, it will be showed that no autocorrelation is present as expected. The second case is the one for which erroneous matrices for noise covariances are reported, and consequently process filter deviates from the optimality. This case shows autocorrelation, and consequently and identification for the system order is performed. Finally, the third case is the one in which the model used inside MPC and in the calculation of the filter as well differs from the real process. Even in this case, an identification is performed in order to obtain the value of observability matrix rank.

n_{max}^{oc} *and* oc_{lim}^{max} *determination*

In order to determine a correct value for the parameters n_{max}^{oc} and oc_{lim}^{max}, a Monte-Carlo simulation with 100 different noise sequences was performed on the system in equation 5.31. For this simulation, the following parameters were assumed.

- The real covariance of process noise is

$$Q_w = \begin{bmatrix} 3 \cdot 10^{-3} I & 0 \\ 0 & 3 \cdot 10^{-5} I \end{bmatrix}$$

  while the real covariance of the measurement noise

$$R_v = 2 \cdot 10^{-3} I$$

Table 13: Number of autocorrelated systems in the Monte-Carlo simulation varying n_{max}^{oc} value. $oc_{lim}^{max} = 1.5\varepsilon_{95}$

n_{max}^{oc}	Autocorrelated systems
4	71
5	51
6	29
7	13
8	7

with I identity matrix of suitable dimension.

- Data are collected for 2500 sampling times.
- Two set point changes of magnitude 0.1 and 0.1 were introduced respectively at time $k = 1250$ on the first output and at time $k = 834$ on the second output.

MPC scheme and Kalman filter are designed in case there are no mismatches, and figure 38 was obtained. In that figure, the value of θ is reported in case of different values for the parameter n_{max}^{oc} and a value for oc_{lim}^{max} parameter of $1.5\varepsilon_{95}$.
The line corresponds to $\theta = 1$, which represents the lower limit

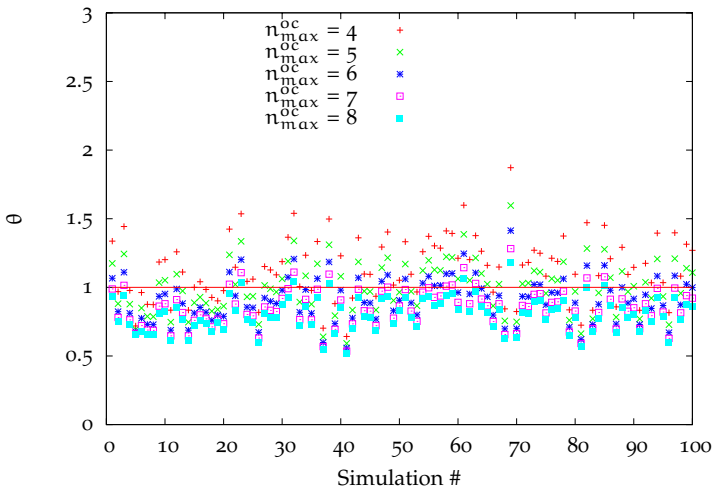

Figure 38: Values assumed by θ parameter when parameter n_{max}^{oc} varies in case of no model nor noise covariance mismatch. $oc_{lim}^{max} = 1.5\varepsilon_{95}$

for the system to be considered affected by autocorrelation as previously said. In table 13 the number of times that θ shows autocorrelation is reported for every single n_{max}^{oc} value. Being the system optimal, this number should be as close as possible to 0. It is possible to see in table that autocorrelation is found in several cases, even if it is not present. The value of 7 was selected because, even if 8 seems to grant better results, experimental evidences showed that sometimes it can give erroneous results in

Table 14: Number of autocorrelated systems in the Monte-Carlo simulation varying oc_{lim}^{max} value. $n_{max}^{oc} = 7$

oc_{lim}^{max}	Autocorrelated systems
2	2
1.5	13
1.33	26
1.25	35
1.2	44

case autocorrelation is present. This should be avoided, in order not to consider optimal a system that presents autocorrelation. The reason for the presence of this problem is the influence of the number of available data on θ value. If that number increases, the properties of the autocorrelation sequence tend to be closer to the theoretical value, so $n^{oc} = 8$ seems to be a too conservative choice.

After this, considering a system with $n_{max}^{oc} = 7$, the same Monte-Carlo simulation was performed varying parameter oc_{lim}^{max} in order to find the best value for that parameter. Figure 39 has been obtained. In table 14 the number of autocorrelated data sets found in the Monte-Carlo simulation is reported.

In this case too, even if $oc_{lim}^{max} = 2\varepsilon_{95}$ seems to work better, a value of $1.5\varepsilon_{95}$ was selected for the same reason for which $n_{max}^{oc} = 7$ was used instead of $n_{max}^{oc} = 8$.

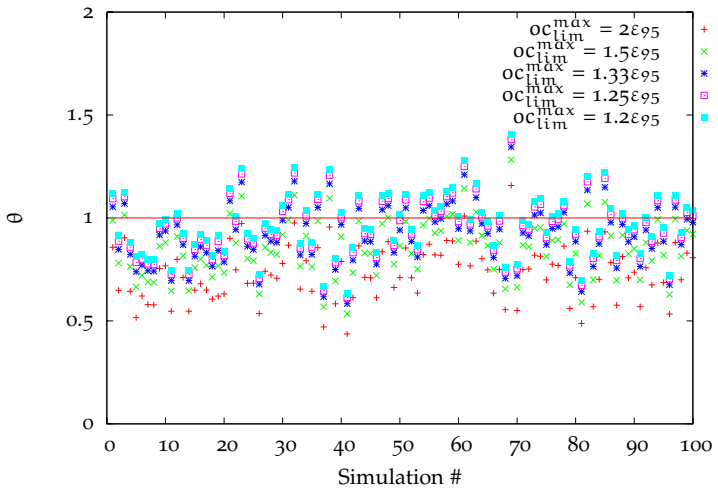

Figure 39: Values assumed by θ parameter when parameter oc_{lim}^{max} varies in case of no model nor noise covariance mismatch. $n_{max}^{oc} = 7$

Now, a Monte-Carlo simulation with $n_{max}^{oc} = 7$ and $oc_{lim}^{max} = 1.5\varepsilon_{95}$ was performed in case of model mismatch and noise covariance matrices mismatch: the results are reported in table 15.

Table 15: Number of autocorrelated systems in two Monte-Carlo simulations with mismatches. $oc_{lim}^{max} = 1.5\varepsilon_{95}$, $n_{max}^{oc} = 7$

Case	Autocorrelated systems recovered
model mismatch	100
noise covariance mismatch	100

It is possible to see that autocorrelation is exactly found in every case.

5.5.2 Results in case of no mismatches

As previously introduced, this is the nominal case. The MPC algorithm for this simulation is designed using the real process in eq. 5.31 as model, so no mismatches are present. The Kalman filter for this system is obtained using the real noise covariance matrices and the real process and for this reason it is the optimal one. With these premises, in this case there should be no autocorrelation in prediction error sequence. In figure 40 the inputs and the outputs sequences are reported for this system. In figure 41 the prediction error sequence is and the corresponding $\mathbf{r}(\tau)$ parameter are reported. Calculating the parameter θ for both the prediction error channel, it results eq. 5.32

$$\theta^1 = 0.8 \qquad \theta^2 = 0.71 \tag{5.32}$$

the superscript indicating the channel.
As it was expected, θ is less than 1 both for the first and the second channel, so it is confirmed that no autocorrelation is present in this system, and its performances are optimal.

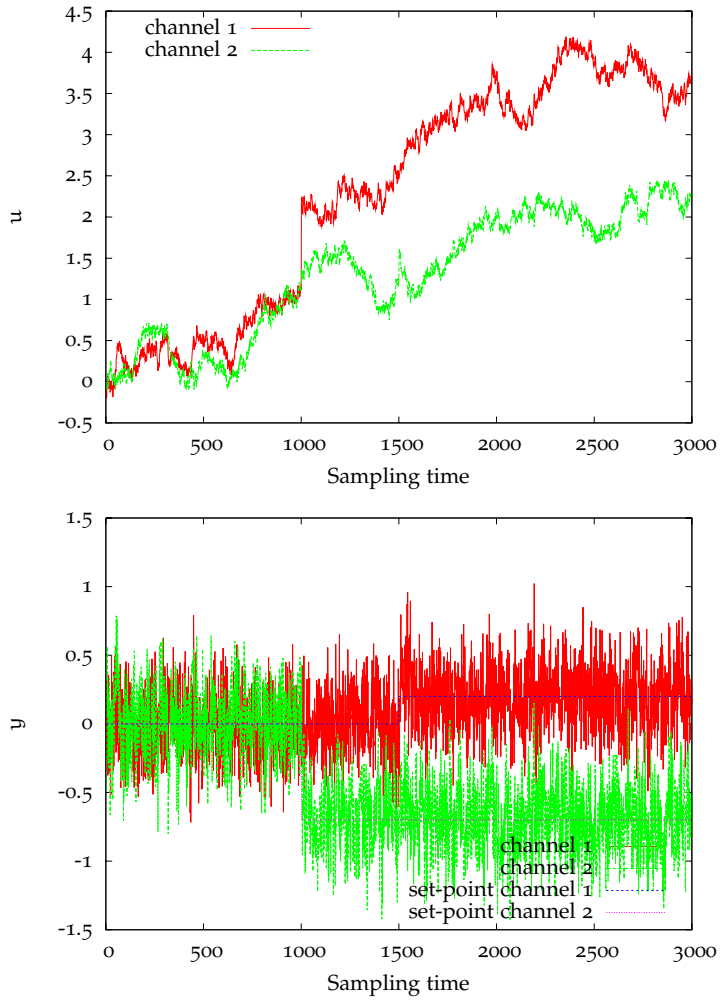

Figure 40: Inputs(top) and outputs(bottom) for the case of no model nor noise covariance mismatch

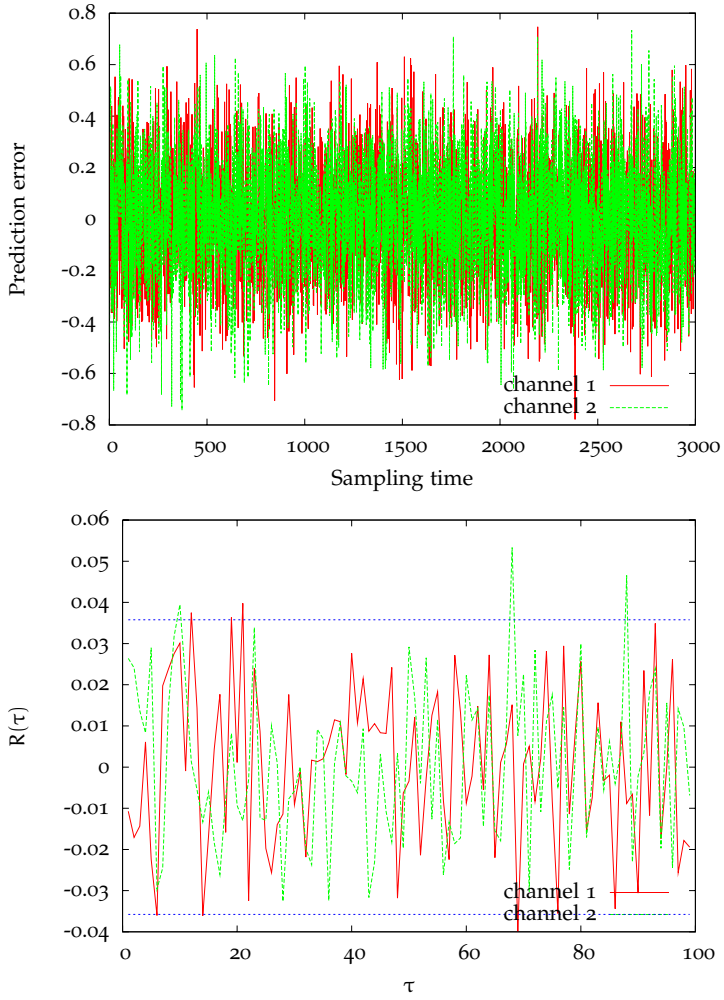

Figure 41: Prediction error sequence (top) and relative $\mathbf{r}(\tau)$ parameter (bottom) for the case of no model nor noise covariance mismatch

5.5.3 Mismatch in the process model

In this case, a wrong model is used to design the MPC and the Kalman filter. This model is the same of chapter 3 and its state space formulation is reported in equation 5.33.

$$\begin{aligned}\hat{A} &= \begin{bmatrix} 0.9 & 0.2 \\ 0 & 0.7 \end{bmatrix} \\ \hat{B} &= \begin{bmatrix} 0.3 & 0.1 \\ -0.2 & 0.2 \end{bmatrix} \\ \hat{C} &= \begin{bmatrix} 0.3 & 0.3 \\ -0.3 & 0.8 \end{bmatrix}\end{aligned} \tag{5.33}$$

In figure 42 input and output plots are reported, while the prediction error sequence and the $\mathbf{r}(\tau)$ parameter plot are in figure 43.

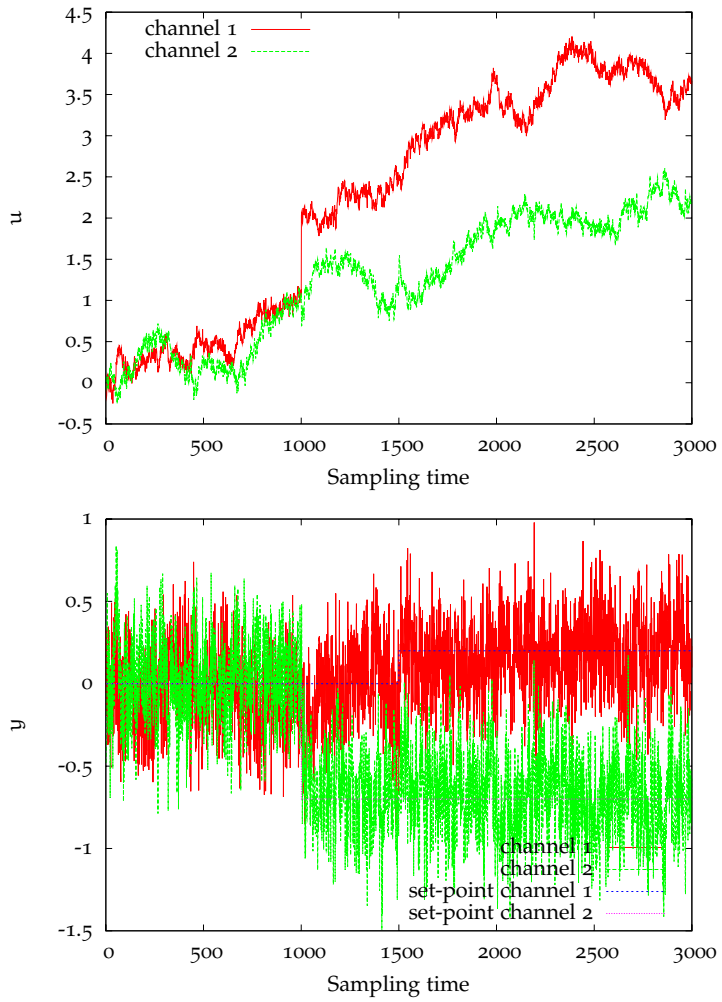

Figure 42: Inputs(top) and outputs(bottom) for the case of model mismatch

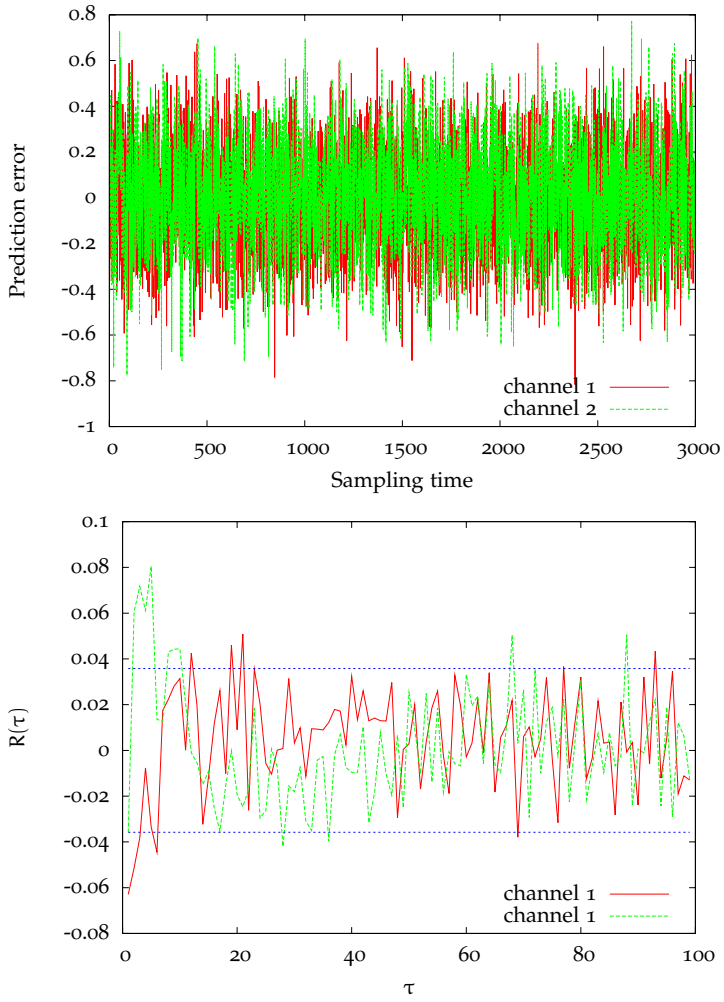

Figure 43: Prediction error sequence(top) and relative $\mathbf{r}(\tau)$ parameter (bottom) for the case of model mismatch

Calculating the parameter θ for this sequence,it results

$$\theta^1 = 1.3 \qquad \theta^2 = 1.61 \tag{5.34}$$

Being $\theta > 1$ the presence of autocorrelation is confirmed, as it was expected. For this reason, the following step is the identification of the rank of the observability matrix for the system. Performing that task following the guidelines previously introduced, and using a $\delta = 0.1$ it results

$$\operatorname{rank} \Gamma = 6 \tag{5.35}$$

Being $6 > n + p + \gamma$, the presence of a model mismatch is correctly reported, and so it is possible to say that the method works properly in this case.

Mismatch in noise covariances

The last simulation for the process in equation 5.31 refers to the case in which wrong noise covariance matrices are used

for the Kalman filter construction. namely $Q_w = 0.08I$ and $R_v = 0.0008I$.

No model mismatches are present, so MPC algorithm is based on the real process matrices of equation 5.31. Even in this case, the presence of autocorrelation should be noticed. In figure 44 input and output data for this case are reported, while prediction error sequence and $\mathbf{r}(\tau)$ parameter are plotted in figure 45. Defining θ parameter for this system it results:

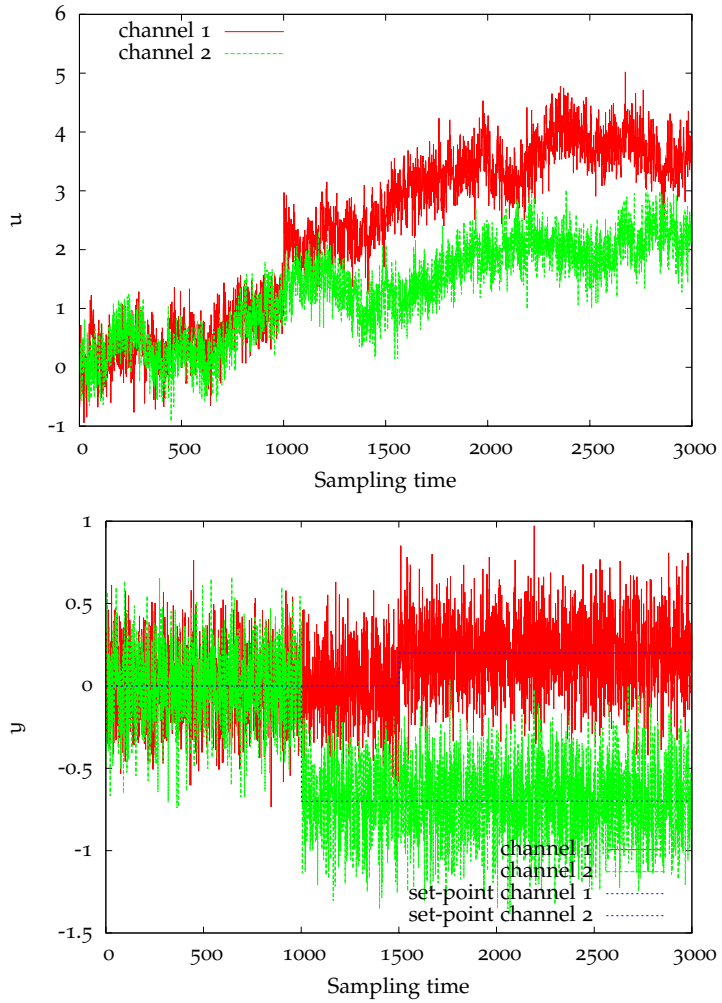

Figure 44: Inputs(top) and outputs(bottom) for the case of covariance matrices mismatch

$$\theta^1 = 5.57 \qquad \theta^2 = 5.37 \tag{5.36}$$

and so autocorrelation is correctly indicated as expected.

Even in this case, for the reason that autocorrelation is found in prediction error, an identification from data is required. This step result in

$$\operatorname{rank} \Gamma = 2 \tag{5.37}$$

2 is less than $n + p + \gamma$. Recalling the results previously introduced, the system is affected by a mismatch in the noise covariances used for the definition of the Kalman filter. This is actually

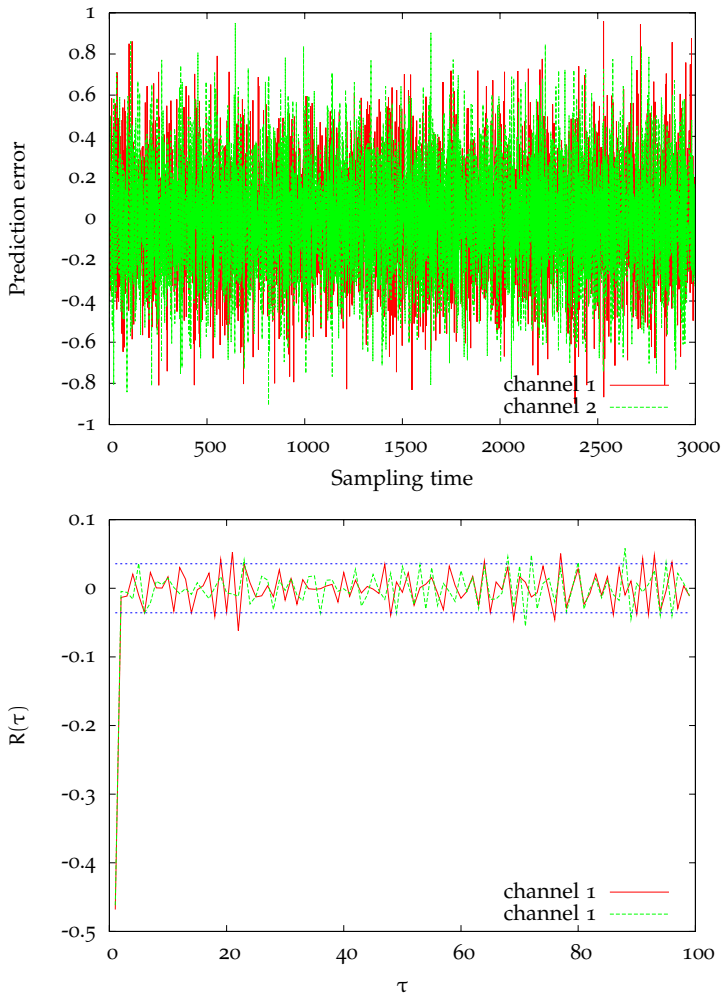

Figure 45: Prediction error sequence(top) and relative $\mathbf{r}(\tau)$ parameter (bottom) for the case of covariance matrices mismatch

the real situation, so it is possible to say that even for this case the method has worked in a correct way.

5.6 CONCLUSIONS

A novel method for MPC monitoring has been presented, which is not computationally demanding. It can be used to decide between the cases of optimal and suboptimal behavior of the MPC scheme. Furthermore, in case of suboptimal control action, it is possible to discern the cause of performance degradation selecting between model or noise matrices mismatch. The method does not take into account the internal structure of the MPC algorithm, so it can be used regardless of linearity of the control action w.r.t the states of the system. Finally, it does not show problems in case of presence of set-point changes, giving the possibility of considering a wider class of data.
Some difficulties can arise in the identification of the order of

observability matrix of the system from data. It was noticed, indeed, that in several cases the results could lead to incorrect estimation of the causes generating the problem. This is probably due to the high noise-to-signal ratio that is present in the prediction error sequence, which avoids the system to recognize easily the deterministic contribution to the prediction error sequence.

For this reason, it is suggested that future researches in this field would consider the possibility of finding a tailored identification method for obtaining the rank of the observability matrix of the composite system.

6 CONCLUSIONS

In this work, system identification methods and MPC monitoring techniques have been considered.

In the field of identification, the attention has been focused on the definition of reliable methods for obtaining consistent models for particular families of systems, that is unstable and ill-conditioned systems. These two kinds of systems usually present problems in identification, and it is common to obtain models for them that can not describe the system in an appropriate way.
Unstable systems have been analyzed considering two separated steps. The analysis of the dynamics was conducted using a classic subspace identification method, in particular the one suggested by Huang et al.[15]. It has been showed that such an approach grants consistent estimates of the poles, that is consistent estimates for the system matrix A in a state space sense, and consequently for matrix C.
In the second step, in order to obtain consistent estimates of matrix B the work by Forssell & Ljung [7] was considered. They introduced a method for identifying unstable systems given in the form of an output-error model. In particular, in this method B is obtained defining a new unstable model which asymptotically converges to the real process, but which on the contrary has a stable predictor and so can be used to perform a linear regression on data. Such a model can be constructed separating the dynamics of the system in a stable part and in an unstable part, and introducing a filter deriving from the unstable part of the system but which presents stable poles.
In the end, in this research an identifying method for unstable systems was defined. This method presents the advantages of subspace methods for the identification of the system dynamic, that is it is usually faster than predictor error methods. Furthermore, it can also handle data coming from unstable plant, such as prediction error methods usually do, without giving problems.

As a second objective, the definition of a valuable approach in the identification of ill-conditioned systems has been considered. In particular, the attention has been focused in input design, that is in finding the kind of data that can grant the best model using a subspace identification approach. As known, ill-conditioned systems present a "strong direction" for which, usually, the information contained in data is very high, and a "weak direction", which on the contrary cannot be recovered easily from data because it present low level of information. In order to address this problem, in literature rotated inputs were introduced. These are

a kind of inputs which excites the weak direction dynamic with the same power of the strong direction dynamic, with the aim of obtaining data with the maximum information for both the directions. As a first result of this research, a general design method for this kind of inputs was defined, which can handle square and non-square systems. Actually, it has to be considered that rotated input design is hard for the most of the systems because it is needed a good approximation of the steady state gain matrix for the system, which is usually not available. It has been showed, however, that the best models for control are obtained in case of a CL data set instead of rotated inputs.

In the end, in this research a MPC monitoring method has been presented. It represent an extension of a previous work by Harrison [12], which presents problems in handling some kind of data, such as data in which the combination of active constraints is not fixed and data in which a set-point change is present. Both cases appear frequently in industrial practice. The proposed method analyzes the prediction error sequence of the system in order to obtain information on the action of MPC controller, in particular on its optimality or sub-optimality. A statistical analysis is performed on that sequence, in order to see if it shows autocorrelation. In case autocorrelation is present, the control scheme it has been showed that the controller is working in sub-optimal conditions, and so a further investigation is needed in order to find the problems that affect the system. For this purpose, the method presented in this work performs an identification on the system that generates the prediction error sequence, in order to check for its observability. Using a rank analysis, it has been demonstrated that the order for that system should be $n + p$ in case the controller has been defined using wrong matrices for noise covariance, where n is the number of states of the model and p the number of the outputs. On the other hand, the order should be $2n + p$ in theory in case the system presents a model mismatch. Actually, an additional parameter γ was introduced for taking into account fictitious poles due to noise in the system that could arise especially with high order systems. For this reason, γ parameter has been defined in dependence of the system order.

In conclusion, the presented method for MPC monitoring can handle a wide class of data without necessities of pre-treatment or selection on sets.

BIBLIOGRAPHY

[1] I. Ananth and M. Chidambaram. Closed loop identification of transfer function model for unstable systems. 336(7):1055–1061, Sep 1999. (Cited on page 64.)

[2] N. Argawal, B. Huang, and E.C. Tamayo. Assessing model predictive control (mpc) performance. 1. probabilistic approachfor constrain analysis. *Ind. Eng. Chem. Res.*, 46:8101–8111, 2007. (Cited on page 86.)

[3] M. J. Bruwer and J. F. MacGregor. Robust multi-variable identification: Optimal experimental design with constraints. *J. Proc. Cont.*, 16:581–600, 2006. (Cited on page 47.)

[4] A. Chiuso and G. Picci. On the ill-conditioning of subspace identification with inputs. *Automatica*, 40:575–589, 2004. (Cited on page 47.)

[5] J. S. Conner and D. E. Seborg. An evaluation of MIMO input designs for process identification. *Ind. Eng. Chem. Res.*, 43:3847–3854, 2004. (Cited on page 47.)

[6] B. L. Cooley and J. H. Lee. Control-relevant experiment design for multivariable systems described by expansions in orthonormal bases. *Automatica*, 37:273–281, 2001. (Cited on page 47.)

[7] U. Forssell and L. Ljung. Identification of unstable systems using output error and box–jenkins model structure. *IEEE Trans. Auto. Cont.*, 45(1):137–141, Jan 2000. (Cited on pages 64, 69, 81, and 113.)

[8] E. Gabay and S. J. Merhav. Parametric identification of unstable linear systems. *IEEE Trans. Auto. Cont.*, 23(3):500–502, Jun 1978. (Cited on page 63.)

[9] M. Gevers and L. Ljung. Optimal experiment designs with respect to the intended model application. *Automatica*, 22:543–554, 1986. (Cited on page 46.)

[10] R. B. Gopaluni, R. S. Patwardhan, and S. L. Shah. Experiment desing for MPC relevant identification. In *Proceedings of American Control Conference*, pages 2713–2718, Anchorage, AK (USA), 2002. (Cited on page 47.)

[11] K. E. Häggblom and J. M. Böling. Multimodel identification for control of an ill-conditioned distillation column. *J. Proc. Cont.*, 8(2):209–218, 1998. (Cited on page 47.)

[12] Christopher A. Harrison. *Towards the Performance Monitoring of Constrained Control Systems*. PhD thesis, The University of Texas at Austin, 2006. (Cited on pages 86, 94, and 114.)

[13] H. Hjalmarsson. From experiment design to closed-loop control. *Automatica*, 41:393–438, 2005. (Cited on page 47.)

[14] H. Hjalmarsson, M. Gevers, and F. de Bruyne. For model-based control design, closed-loop identification gives better performance. *Automatica*, 32:1659–1673, 1996. (Cited on page 46.)

[15] B. Huang, S. X. Ding, and S. J. Qin. Closed-loop subspace identification: an orthogonal projection approach. *J. Proc. Cont.*, 15:53–66, 2005. (Cited on pages 25, 26, 27, 28, 46, 61, 66, 74, 77, 81, and 113.)

[16] E. W. Jacobsen and S. Skogestad. Inconsistencies in dynamic models for ill-conditioned plants: Application to low-order models of distillation columns. *Ind. Eng. Chem. Res.*, 33:631–640, 1994. (Cited on page 47.)

[17] C. W. Koung and J. F. MacGregor. Design of identification experiments for robust control. A geometric approach for bivariate processes. *Ind. Eng. Chem. Res.*, 32:1658–1666, 1993. (Cited on pages 46, 47, and 48.)

[18] C. W. Koung and J. F. MacGregor. Identification for robust multivariable control: the design of experiments. *Automatica*, 30:1541–1554, 1994. (Cited on pages 46, 47, 48, 49, 52, and 60.)

[19] Wallace E. Larimore. Automated and optimal system identification by canonical variables. In *American Control Conference, Baltimore, MD*, volume 2, pages 1640–1644, Piscataway, NJ, 1994. IEEE. (Cited on page 46.)

[20] L. Ljung. *System identification: theory for the user*. Prentice–Hall, 1999. (Cited on pages 5, 7, 13, 14, 15, 27, 46, 63, and 64.)

[21] F. Loquasto and D.E. Seborg. Model predictive controller based on pattern classification and PCA. In *ACC (American Control Conference)*, pages 1968–1973, Denver, Colorado (USA), 2003. (Cited on page 86.)

[22] G. Marchetti, C. Scali, and D. R. Lewin. Identification and control of open loop unstable processes by relay methods. *Automatica*, 37:2049–2055, 2001. (Cited on page 64.)

[23] Pratik Misra and Michael Nikolaou. Input design for model order determination in subspace identification. *AIChE J.*, 49(8):2124–2132, Aug 2003. (Cited on pages 47, 48, 50, 52, 57, and 60.)

[24] M. Moonen, B. De Moor, J. Ramos, and S. Tan. A subspace identification algorithm for descriptor systems. *Sys. Cont. Let.*, 19(1):47–52, Jul 1992. (Cited on page 64.)

[25] Brian J. Odelson, Murali R. Rajamani, and James B. Rawlings. A new autocovariance least-squares method for estimating noise covariances. *Automatica*, 42:303–308, 2006. (Cited on page 11.)

[26] G. Pannocchia. Robust disturbance modeling for model predictive control with application to multivariable ill-conditioned processes. *J. Proc. Cont.*, 13:693–701, 2003. (Cited on page 47.)

[27] G. Pannocchia, A. Micchi, R. Bulleri, A. Brambilla, and G. Marchetti. Multivariable subspace identification and predictive control of a heat-integrated superfractionator rigorous model. In *Proceedings of ADCHEM 2006 (International Symposium on Advanced Control of Chemical Process)*, volume I, pages 421–426, Gramado (Brazil), 2006. (Cited on pages 27, 65, 66, 74, and 81.)

[28] G. Pannocchia and J. B. Rawlings. Disturbance models for offset-free model predictive control. *AIChE J.*, 49:426–437, 2003. (Cited on pages 52 and 88.)

[29] R.S. Patwardhan and S.L. Shah. Issues in performance diagnostic of model-based controller. *J. Proc. Cont.*, 12:413–427, 2002. (Cited on page 86.)

[30] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Cont. Eng. Pract.*, 11:733–764, 2003. (Cited on page 85.)

[31] S. J. Qin, W. Lin, and L. Ljung. A novel subspace identification approach with enforced causal models. *Automatica*, 41:2043–2053, 2005. (Cited on page 27.)

[32] S. Skogestad and M. Morari. Implications of large RGA-elements on control performance. *Ind. Eng. Chem. Res.*, 26:2323–2330, 1987. (Cited on page 47.)

[33] S. Skogestad and M. Morari. LV-control of a high purity distillation column. *Chem. Eng. Sci.*, 43:33–48, 1988. (Cited on page 53.)

[34] P. Stec and Y. Zhu. Some study on identification of ill-conditioned processes for control. In *Proceedings of American Control Conference*, pages 1202–1207, Arlington, VA (USA), 2001. (Cited on page 47.)

[35] P. Van Overschee and B. De Moor. N4SID: Subspace algorithms for the identification of combined deterministic stochastic systems. *Automatica*, 30:75–93, 1994. (Cited on page 46.)

[36] P. Van Overschee and B. De Moor. A unifying theorem for three subspace system identification algorithms. *Automatica*, 31:1853–1864, 1995. (Cited on page 46.)

[37] M. Verhaegen. Identification of the deterministic part of MIMO state space models given in innovation form from input-output data. *Automatica*, 30:61–74, 1994. (Cited on page 46.)

[38] J. Wang and S. J. Qin. A new subspace identification approach based on principal component analysis. *J. Proc. Cont.*, 12:841–855, 2002. (Cited on pages 26, 27, 46, and 51.)

[39] J. Yu and S. J. Qin. Statistical MIMO controllers performance monitoring. part i:data-driven covariance benchmark. *J. Proc. Cont.*, 18:277–296, 2008. (Cited on page 86.)

[40] Q. Zhan, T. Li, and C. Georgakis. Steady state optimal test signal design for multivariable model based control. *Ind. Eng. Chem. Res.*, 45:8514–8527, 2006. (Cited on page 48.)

[41] Y. Zhu and P. Stec. Simple control-relevant identification test methods for a class of ill-conditioned processes. *J. Proc. Cont.*, 16:1113–1120, 2006. (Cited on page 47.)

[42] Yucai Zhu. Multivariable process identification for MPC: the asymptotic method and its applications. *J. Proc. Cont.*, 8(2):101–115, 1998. (Cited on page 46.)

[43] Yucai Zhu. *Multivariable system identification for process control*. Elsevier Science, 2001. (Cited on pages 46 and 52.)