Robust Empirical Model-Based Algorithms for Nonlinear Processes

by

Juan Rosendo Díaz Mendoza

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Chemical Engineering

Waterloo, Ontario, Canada, 2010

Abstract

This research work proposes two robust empirical model-based predictive control algorithms for nonlinear processes. Chemical process are generally highly nonlinear thus predictive control algorithms that explicitly account for the nonlinearity of the process are expected to provide better closed-loop performance as compared to algorithms based on linear models. Two types of models can be considered for control: first-principles and empirical. Empirical models were chosen for the proposed algorithms for the following reasons: (i) they are less complex for on-line optimization, (ii) they are easy to identify from input-output data and (iii) their structure is suitable for the formulation of robustness tests.

One of the key problems of every model that is used for prediction within a control strategy is that some model parameters cannot be known accurately due to measurement noise and/or error in the structure of the assumed model. In the robust control approach it is assumed that processes can be represented by models with parameters' values that are assumed to lie between a lower and upper bound or equivalently, that these parameters can be represented by a nominal value plus uncertainty. When this uncertainty in control parameters is not considered by the controller the control actions might be insufficient to effectively control the process and in some extreme cases the closed-loop may become unstable. Accordingly, the two robust control algorithms proposed in the current work explicitly account for the effect of uncertainty on stability and closed-loop performance.

The first proposed controller is a robust gain-scheduling model predictive controller (MPC). In this case the process is represented within each operating region by a state-affine model obtained from input-output data. The state-affine model matrices are used to obtain a state-space based MPC for every operating region. By combining the state-affine, disturbance and controller equations a closed-loop representation was obtained. Then, the resulting mathematical representation was tested for robustness with linear matrix inequalities (LMI's) based on a test where the vertices of the parameter box were obtained by an iterative procedure. The result of the LMI's test gives a measure of performance referred to as γ that relates the effect of the disturbances on the process outputs. Finally, for the gain-scheduling part of the algorithm a set of rules was proposed to switch between the available controllers according to the current process conditions. Since every combination of the controller tuning parameters results in a different value of γ, an optimization problem was proposed to minimize γ with respect to the tuning parameters. Accordingly, for the proposed controller it was ensured that the effect of the disturbances on the output variables was kept to its minimum. A bioreactor case study was presented to show the benefits of the proposed algorithm. For comparison purposes a non-robust linear MPC was also designed. The results show that the proposed algorithm has a clear advantage in terms of performance as compared to non-robust linear MPC techniques.

The second controller proposed in this work is a robust nonlinear model predictive controller (NMPC) based on an empirical Volterra series model. The benefit of using a Volterra series model for this case is that its structure can be split in two sections that account for the nominal and uncertain parameter values. Similar to the previously proposed gain-scheduled controller the model parameters were obtained from input-output data. After identifying the Volterra model, an interconnection matrix and its corresponding uncertainty description were found. The interconnection matrix relates the process inputs and outputs and is built according to the type of cost function that the controller uses. Based on the interconnection representing the system a robustness test was proposed based on a structured singular value norm calculation (SSV). The test is based on a min-max formulation where the worst possible closed-loop error is minimized with respect to the manipulated variables. Additional factors that were considered in the cost function were: manipulated variables weighting, manipulated variables restrictions and a terminal condition. To show the benefits of this controller two case studies were considered, a single-input-single-output (SISO) and a multiple-input-multiple-output (MIMO) process. Both case studies show that the proposed controller is able to control the process. The results showed that the controller could efficiently track set-points in the presence of disturbances while complying with the saturation limits imposed on the manipulated variables. This controller was also compared against a non-robust linear MPC, non-robust NMPC and non-robust first-principles NMPC. These comparisons were performed for different levels of uncertainty and for different values of the suppression or control actions weights. It was shown through these comparisons that a tradeoff exists between nominal performance and robustness to model error. Thus, for larger weights the controller is less aggressive resulting in more sluggish performance but less sensitivity to model error thus resulting in smaller differences between the robust and non-robust schemes. On the other hand when these weights are smaller the controller is more aggressive resulting in better performance at the nominal operating conditions but also leading to larger sensitivity to model error when the system is operated away from nominal conditions. In this case, as a result of this increased sensitivity to model error, the robust controller is found to be significantly better than the non-robust one.

Acknowledgements

I would like to thank Prof. Hector Budman for his supervision, guidance, support and help provided during the PhD program.

The financial support provided by NSERC (Natural Sciences and Engineering Research Council of Canada) is gratefully acknowledged.

I would like to thank the following professors from Universidad Nacional Autonoma de México (UNAM) for teaching me and giving me the tools to keep learning, in alphabetical order: Prof. Fernando Barragán-Aroche, Prof. Eduardo Bárzana-García, Prof. Martín Hernández-Luna, Prof. Celestino Montiel-Maldonado and Prof. Pedro Roquero-Tejeda.

Dedication

To my mother, a mi mamá, a su Señoría, Thank you for everything

Contents

List of Tables **xv**

List of Figures **xvii**

1 Introduction **1**

2 Literature review **5**

2.1 Empirical models 6

2.1.1 State-Affine 7

2.1.2 Volterra series 7

2.2 Model predictive control 9

2.2.1 Linear MPC 15

2.2.2 Nonlinear MPC 16

2.3 Robust control 18

2.3.1 Linear matrix inequalities based test 20

2.3.2 Structured singular value 21

2.4 Robust model predictive control 22

2.4.1 Robust LMPC 22

2.4.2 Robust NMPC 24

3 Robust state-affine model-based gain-scheduling MPC **26**

3.1 Introduction 27

3.2 State-Space formulation of the closed-loop system 31

3.2.1 Model uncertainty and disturbances 31

3.2.2 State-Space formulation of the MPC controller 33

3.2.3 Closed-Loop state-space model for robustness analysis 36

3.3 Robust stability and robust performance tests 40

3.4 Robust gain-scheduled MPC design 42

3.5 Case study results 51

3.6 Conclusions 69

4 Additional work for the gain-scheduling MPC 71

4.1 Case study 73

4.2 Conclusions 80

5 Structured singular valued based robust nonlinear model predictive controller using Volterra series models 82

5.1 Introduction 83

5.2 Volterra series models 86

5.3 Calculation of the worst case in the presence of model error 88

5.3.1 Manipulated variables movement penalization 92

5.3.2 Manipulated variables constraints 93

5.3.3 Terminal condition 95

5.3.4 Robust NMPC control law 96

5.4 Case studies 98

5.4.1 SISO system 98

5.4.2 MIMO system 105

5.5 Conclusions 109

6 Comparison between a robust nonlinear empirical model-based predictive controller and a first-principles nonlinear model-based predictive controller 111

6.1 Introduction 112

6.2 Robust nonlinear model predictive control based on Volterra series models 114

6.3 Nonlinear model predictive control based on first-principles model 120

6.4 Case study 122

6.5 Conclusions 134

7 Conclusions 136

7.1 Robust gain-scheduling NMPC 136

7.2 Volterra series based robust NMPC 139

7.2.1 SISO case study 140

7.2.2 MIMO case study 141

7.3 Directions for future research 142

Bibliography 143

Appendices

A State-Affine model parameters for chapter 3 153

B State-Affine model parameters for chapter 4 155

C Additional information for the interconnection matrix 157

D Interconnection matrix construction 160

E Disturbance characteristics for chapter 5 173

F Disturbance characteristics for chapter 6 **175**

G Volterra series model parameters for chapter 5 **176**

H Volterra series model parameters for chapter 6 **178**

I Matlab codes **179**

List of Tables

3.1 List of differences between (Gao, 2004) and the current work 31

3.2 List of parameter values 52

3.3 Optimal nominal GSMPC and LMPC γ values for case study A 55

3.4 Optimal nominal GSMPC and LMPC γ values for case study B 56

3.5 Optimal GSMPC γ and $\mathbf{R}$ values for case study A 56

3.6 Optimal GSMPC γ and $\mathbf{R}$ values for case study B 56

3.7 Optimal LMPC γ and $\mathbf{R}$ values for case study A 57

3.8 Optimal LMPC γ and $\mathbf{R}$ values for case study B 57

3.9 Simulation results for case study A 67

3.10 Simulation results for case study B 67

4.1 Values for normalization 74

4.2 Optimal nominal GSMPC γ and manipulated variables weight matrix for $c_e = 0$ 75

4.3 Optimal nominal LMPC γ and manipulated variables weight matrix for $c_e = 0$ 75

4.4 Optimal GSMPC γ and manipulated variables weight matrix for $c_e = 0$ . . 76

4.5 Optimal LMPC γ and manipulated variables weight matrix for $c_e = 0$ . . . 76

4.6 Optimal GSMPC γ and manipulated variables weight matrix for $c_e = 0.1$. 77

4.7 Optimal GSMPC γ and manipulated variables weight matrix for $c_e = 0.25$ 77

4.8 Optimal LMPC γ and manipulated variables weight matrix for $c_e = 0.1$ . . 77

4.9 Optimal LMPC γ and manipulated variables weight matrix for $c_e = 0.25$. 77

4.10 Comparison of γ optimal average for several values of c_e 77

4.11 Simulation results for $c_e = 0$ 79

4.12 Simulation results for $c_e = 0.1$ 79

4.13 Simulation results for $c_e = 0.25$ 79

4.14 Maximum absolute value of the manipulated variables achieved during simulation for GSMPC 80

4.15 Maximum absolute value of the manipulated variables achieved during simulation for LMPC 80

5.1 IAE comparisons between robust NMPC, non-robust NMPC and non-robust LMPC for SISO case study C 103

5.2 IAE comparisons between robust NMPC, non-robust NMPC and non-robust LMPC for SISO case study D 105

5.3 Process set-points and operating conditions for MIMO case study 107

5.4 Comparison between robust NMPC, non-robust NMPC and linear MPC controllers for MIMO case study 109

6.1 Process operating conditions 124

6.2 Values for normalization 125

6.3 IAE results for case 1 $[\mathrm{IAE}_{\mathrm{FP-NMPC}}, \mathrm{IAE}_{\mathrm{RNMPC}}]$ 129

6.4 IAE results for case 2 $[\mathrm{IAE}_{\mathrm{FP-NMPC}}, \mathrm{IAE}_{\mathrm{RNMPC}}]$ 129

6.5 IAE results for case 3 $[\mathrm{IAE}_{\mathrm{FP-NMPC}}, \mathrm{IAE}_{\mathrm{RNMPC}}]$ 129

6.6 IAE results for case 4 $[\mathrm{IAE}_{\mathrm{FP-NMPC}}, \mathrm{IAE}_{\mathrm{RNMPC}}]$ 130

6.7 IAE results for case 5 $[\mathrm{IAE}_{\mathrm{FP-NMPC}}, \mathrm{IAE}_{\mathrm{RNMPC}}]$ 130

6.8 IAE results for case 6 $[\mathrm{IAE}_{\mathrm{FP-NMPC}}, \mathrm{IAE}_{\mathrm{RNMPC}}]$ 130

6.9 Average IAE for case studies 1 to 6 130

6.10 IAE for case study 7 134

6.11 IAE for case study 8 134

A.1 Value of the matrices $\mathbf{F}_0$, $\mathbf{F}_1$ and $\mathbf{F}_2$ 153

A.2 Value of the matrices $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$ 153

A.3 Value of the matrix $\mathbf{W}_F$ 154

B.1 Value of the matrices $\mathbf{F}_0$, $\mathbf{F}_1$ and $\mathbf{F}_2$ 155

B.2 Value of the matrices $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$ 155

B.3 Value of the matrix $\mathbf{W}_F$ 156

E.1 Disturbance characteristics for SISO case study C (k = sampling instant) . 173

E.2 Disturbance characteristics for SISO case study D, $\beta\,(k = 1) = 0.3$ 174

E.3 Disturbance characteristics for MIMO case study, $q_2\,(k = 1) = 0.5500$ . . . 174

F.1 Disturbance characteristics for MIMO case study (k = sampling instant) . 175

G.1 Volterra series model parameters and uncertainty for SISO case study . . . 176

G.2 Nominal Volterra series model parameters for MIMO case study 177

G.3 Uncertainty associated to the Volterra series model parameters for MIMO case study 177

H.1 Nominal Volterra series model parameters for MIMO case study 178

H.2 Uncertainty associanted to the Volterra series model parameters for MIMO case study 178

I.1 List of programs to calculate $\mathbf{d}$ 179

I.2 List of programs to calculate γ 180

I.3 List of programs to calculate $\mathbf{M}$ for SISO case example 190

I.4 List of programs to calculate $\mathbf{M}$ for MIMO case example with $M = 2$ . . . 195

I.5 List of programs to calculate $\mathbf{M}$ for MIMO case example with $M = 3$ . . . 201

I.6 List of programs to calculate μ-test 210

I.7 List of programs to generate the input sequence used for the identification process 213

List of Figures

3.1 Disturbance used for case studies A and B 59

3.2 Controlled variables, nominal LMPC versus GSMPC for case study A when the process is operating around $S_{\mathrm{in}} = 2.0$ g/L 61

3.3 Manipulated variables, nominal LMPC versus GSMPC for case study B when the process is operating around $S_{\mathrm{in}} = 2.0$ g/L 61

3.4 Controlled variables results for case study A when the process is operating around $S_{\mathrm{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC) 62

3.5 Manipulated variables results for case study A when the process is operating around $S_{\mathrm{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC) 62

3.6 Controlled variables results for case study A when the process is operating around $S_{\mathrm{in}} = 3.0$ g/L, (blue=GSMPC), (red=LMPC) 63

3.7 Manipulated variables results for case study A when the process is operating around $S_{\mathrm{in}} = 3.0$ g/L, (blue=GSMPC), (red=LMPC) 63

3.8 Controlled variables results for case study B when the process is operating around $S_{\mathrm{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC) 64

3.9 Manipulated variables results for case study B when the process is operating around $S_{\mathrm{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC) 64

3.10 Controlled variables results for case study B when the process is operating around $S_{\mathrm{in}} = 4.0$ g/L, (blue=GSMPC), (red=LMPC) 65

3.11 Manipulated variables results for case study B when the process is operating around $S_{\mathrm{in}} = 4.0$ g/L, (blue=GSMPC), (red=LMPC) 65

3.12 Disturbance used for transition between S_{in} studies 68

3.13 Controlled variables for transition between S_{in} studies 68

4.1 Disturbance used for simulation studies, (a) System operating around $S_{\text{in}} =$ 2.0 g/L, (b) System operating around $S_{\text{in}} = 3.0$ g/L 78

5.1 Uncertainty description for an upper LFT 90

5.2 Controlled and manipulated variables for SISO case study A, (black $W_i^{\Delta u} =$ 2.5), (blue $W_i^{\Delta u} = 2.0$), (red $W_i^{\Delta u} = 1.5$), (magenta $W_i^{\Delta u} = 1.0$) 101

5.3 Controlled and manipulated variables for SISO case study B, $W_i^{\Delta u} = 1.5$. 102

5.4 Disturbance for SISO case study C 103

5.5 Controlled and manipulated variables for SISO case study C, $p = 10$, $m = 2$, $W_i^{\Delta u} = 0.75$ 104

5.6 pH neutralization system 106

5.7 Controlled variables profiles for MIMO case study, $p = 10$, $m = 2$, $W_i^{\Delta u_1} =$ 0.75, $W_i^{\Delta u_2} = 1.5$, (black = robust NMPC), (blue = non-robust NMPC), (red = non-robust LMPC) (The information is in deviation variables) . . . 108

5.8 Manipulated variables profiles for MIMO case study, $p = 10$, $m = 2$, $W_i^{\Delta u_1} =$ 0.75, $W_i^{\Delta u_2} = 1.5$ (The information is in deviation variables) 108

6.1 Uncertainty description for an upper LFT 117

6.2 pH neutralization system 123

6.3 Disturbance profile 127

6.4 Set-point changes profile 127

6.5 Controlled variables profiles, $W_i^{\Delta u_1} = 0.25$, $W_i^{\Delta u_2} = 0.10$, $Cv = 8.75$, $n =$ 0.5, (blue=FP-NMPC), (red=RNMPC) 131

6.6 Controlled variables profiles, $W_i^{\Delta u_1} = 0.30$, $W_i^{\Delta u_2} = 0.10$, $Cv = 8.75$, $n =$ 0.5, (blue=FP-NMPC), (red=RNMPC) 132

Chapter 1

Introduction

Chemical process plants are designed to satisfy a set of operational, economic and environmental objectives. In order to achieve them a control strategy must be implemented to operate the plant and make the necessary corrections to achieve the plant goals. Effective control strategies are based on plant models that describe the interrelationship between different plant variables and how they affect the plant operation and objectives. Thus, it is important to have an accurate plant model to achieve good closed-loop performance.

There are two types of models that can be used for control purposes: first-principles and empirical. First-principles models are based on the mass, energy and momentum balances of the system. These models are difficult to obtain since they typically involve a large set of thermo-physical parameters that must be accurately identified and quantified. In some cases the value of the model parameters cannot be obtained because the instrumentation required is expensive or is not accurate enough. In addition, first-principles models have generally a complex structure because they attempt to represent the real process behavior by high-order ordinary and partial differential equations. These complex model structures are generally unsuitable for model-based control.

On the other hand, empirical models are obtained from input-output data that can be regressed to obtain a specific structure to describe the process behavior. The accuracy of these models depends on the quality of data used for calibration. Empirical models are generally easier to obtain than first-principles models because of their generally lower dimensionality but they often provide less accurate predictions for operating conditions that are outside the range used for model calibration.

Currently one of the most popular model-based control strategies is model predictive control (MPC) referred also to as receding horizon control. Model predictive control calculates a set of future plant corrections or manipulated variables moves, based on an optimization that minimizes at each sampling instant a cost function that penalizes the difference between the future process outputs, calculated from a plant model, and the prescribed set-points. After the calculated control actions are implemented in the plant a new set of plant measurements is taken and the optimization is solved again using the new information. The MPC formulation can be improved by considering additional terms in the cost function to account for: process constraints, manipulated variables movement weighting and terminal conditions. Because almost every chemical process exhibits nonlinear behavior it is generally advantageous to use a control strategy that explicitly considers the nonlinearity of the process. Accordingly, nonlinear model predictive control (NMPC) uses a nonlinear model to calculate the predicted outputs value.

The NMPC control strategy is expected to provide performance improvements only if the model represents to a great extent the real behavior of the process. However, there will always be a mismatch between the model, being first-principles or empirical and the real process that results from inaccurate knowledge of the model parameters or from inaccuracy in the identified process dynamics. If the control strategy does not consider the model uncertainty, the process goals might not be attained and in extreme cases the closed-loop may become unstable.

Robust control considers that the model parameters are not exactly known. Instead it considers that the parameters can be assumed to be bounded between lower and upper bounds. These bounds are referred heretofore as model uncertainty. Robust control tools allow to assess whether the process will remain stable (robust stability) and if it will satisfy the required specifications (robust performance) in the presence of uncertainty. Thus, they serve as an effective tool to analyze and synthesize control strategies in the presence of model uncertainty. The effect that this model uncertainty has on the MPC control strategy can be assessed from robust control tests that can be formulated through linear matrix inequalities (LMI) or structured singular value calculations (SSV).

Several methods for designing and analyzing robust MPC controllers based on linear models have been presented in the literature. However, the number of methods available to design a robust NMPC controller is limited. The two possible approaches for designing robust NMPC controllers are: simulation based or analytical testing. The simulation based approach is based on a min-max formulation where the objective function of the worst plant is minimized for all possible combinations between the parameters and its uncertainty description. The main disadvantage of this approach is that it is computationally demanding. In the analytical testing approach robust control tests are used that ensure that the system has robust stability and robust performance. The analytical testing approach requires that the uncertainty can be mathematically factorized from the nominal process model. This is generally not possible in first-principles models since the physical parameters are present in the model in nonlinear functions that are not suitable for effective factorization of the parameters' uncertainty from their nominal values, e.g., the activation energy in Arrhenius exponential expressions. On the other hand it is shown in this thesis that certain types of nonlinear empirical models such as nonlinear state-affine and Volterra series models can be used for the formulation of robust stability and robust performance tests. Then, a robust NMPC controller can be obtained as long as the plant can be represented by these empirical models.

The lack of a design methodology for obtaining a robust NMPC controller based on an empirical model is the motivation of this work. Therefore, the main objective and novelty of this thesis consists in proposing a methodology to design a robust NMPC controller based on an empirical model of the process. For this purpose two different methods are proposed:

1. A state-affine model-based gain-scheduled robust NMPC that uses the LMI based test.

2. A Volterra series model based robust NMPC that uses the SSV concept.

This thesis is organized as follows: chapter 2 contains a literature review of the relevant aspects for this thesis involving empirical models, model predictive control and robust control methods. Chapter 3 contains the methodology to obtain a state-affine model-based gain-scheduled robust NMPC. In chapter 4 the methodology presented in chapter

3 is modified to consider the effect of the manipulated variables in the cost function. Chapter 5 presents the methodology for designing a Volterra series model based robust NMPC. Chapter 6 compares the performance of the controller presented in chapter 5 against a non-robust first-principles NMPC. Finally chapter 6 presents the conclusions of this work. Chapters 3, 5 and 6 were written in journal format, each one with its own abstract, introduction, methodology, results and conclusions.

Chapter 2

Literature review

Effective control strategies are based on representative models of the process to be controlled. There are basically two types of models that can be used, first-principles models and empirical models. The MPC strategy requires a model for calculating the optimal control actions based on the prediction of the output. Therefore, the accuracy of these models in representing the actual process behavior will be of key importance to provide good closed-loop performance.

The parameters of the model will never be accurate due to: nonlinearity, numerical errors and incorrect model identification. Thus, effective control strategies must consider parameter variation within their formulation. Robust control considers that there is a degree of uncertainty in the parameters. Robust control methods allow quantifying how much uncertainty can the system tolerate before it can no longer achieve its specifications thus resulting in useful controller designs.

Since MPC uses a model of the process, the controller needs to consider a certain degree of parameter error. On that ground, it is logical to apply robust control tools for analyzing and designing MPC algorithms.

This chapter is divided as follows: section 2.1 reviews process models and introduces two empirical models that will be considered in this thesis: state-affine models and Volterra series models. Section 2.2 introduces MPC. Section 2.3 provides an introduction to robust control and explains the robust control tools and methodologies that are used in this thesis. Finally, section 2.4 contains a review of robust control studies related to MPC.

2.1 Empirical models

Chemical processes generally exhibit significant nonlinear behavior (Bequette, 2003). Thus it is generally difficult to obtain very accurate first-principles models due to insufficient amount of information or because the process is too complex for accurate modeling. Also, first-principles models are less amenable for robust control design since the uncertainty in parameters cannot be mathematically factorized from the nominal values of these parameters which is a requirement for formulating robustness tests. For these reasons empirical models can be used instead. Empirical models are obtained from input-output data that must be analyzed and processed to obtain a model, they can be divided in parametric and non-parametric models. However, one of the major drawbacks of empirical models is that they only provide a good representation of the process in the region of operating conditions where these models were identified but they may be inaccurate for predicting behavior outside of these regions.

There are two main types of empirical models: linear models and nonlinear models. Linear models provide an appropriate representation of the process in a small neighborhood of an operating point. However, when the process is operated outside this constrained region, the model predictions will not be accurate. On the other hand, nonlinear models tend to capture more accurately the process behavior, making them adequate for controlling a real process in a wide region of operation. Some examples of nonlinear models that were used to represent and control chemical processes are: Hammerstein (Fruzzetti et al., 1997), polynomial autoregressive moving average (ARMA) (Hérnandez and Arkun, 1993), state-affine (Gao, 2004), Volterra series (Doyle III et al., 1995), (Maner et al., 1996) and (Maner and Doyle III, 1997), Volterra-Laguerre (Parker and Doyle III, 2001) and Wiener (Norquay et al., 1999), (Gerkšič et al., 2000) and (Jeong et al., 2001). In this thesis two types of models will be considered: state-affine and Volterra series. Brief descriptions of these empirical models are reviewed in the following subsections.

2.1.1 State-Affine

State-Affine models have been proposed to represent the behavior of nonlinear processes. The concept of state-affine models involves the construction of response maps which map the input sequence to an output sequence (Sontag, 1979). The general form of a state-affine model is as follows:

$$\begin{aligned} x(k+1) =& \mathbf{A}(u(k))\,x(k) + \mathbf{B}(u(k)) \\ y(k) =& \mathbf{C}(u(k))\,x(k) \end{aligned} \tag{2.1}$$

where $\mathbf{A}$ and $\mathbf{C}$ are represented by matrices (Leontaritis and Billings, 1985a,b). Furthermore, every element of $\mathbf{A}$ and $\mathbf{C}$ is a polynomial in $u(k)$, i.e.,

$$\begin{aligned} \mathbf{A}(u(k)) = \mathbf{A}_0 + \mathbf{A}_1 u(k) + \mathbf{A}_2 u(k)^2 + \dots \\ \mathbf{C}(u(k)) = \mathbf{C}_0 + \mathbf{C}_1 u(k) + \mathbf{C}_2 u(k)^2 + \dots \end{aligned} \tag{2.2}$$

A method to identify the model coefficients, i.e., **A**, **B**, **C** and **D** by means of a behavior matrix was proposed by (Sontag, 1979). The methodology requires knowing the Volterra kernels values. A different method was proposed by (Diaz and Desrochers, 1988) that calculates the coefficients from a difference equation approximation to the response map. Contrary to the method of (Sontag, 1979), the method of (Diaz and Desrochers, 1988) provides as a by-product of the identification process the Volterra kernels values.

A more detailed review of the use of state-affine models for control applications will be given in chapter 3.

2.1.2 Volterra series

A Volterra series model relates the output of a process to its previous inputs. For a single-input-single-output (SISO) process the relationship between the output and input can be written as follows:

$$y(t) = y_1(t) + y_2(t) + \dots \tag{2.3}$$

where $y_i(t)$ is defined as

$$y_i(t) = \int_0^\infty \ldots \int_0^\infty h_i(\sigma_1, \ldots, \sigma_i)\, u(t-\sigma_1), \ldots, u(t-\sigma_i)\, \mathrm{d}\sigma_1, \ldots, \mathrm{d}\sigma_i \tag{2.4}$$

The $h_i's$ are the coefficients of the Volterra series model. These $h_i's$ are also known as the Volterra kernels. The Volterra series can be viewed as an extension of the Taylor series expansion used for linear systems (Boyd and Chua, 1985). For practical purposes the series is truncated to include a finite number of terms resulting in the following equation in discrete time:

$$y(k) = h_0 + \sum_{v=1}^{N} \sum_{i_1=0}^{M-1} \ldots \sum_{i_v=0}^{M-1} h_v(i_1, \ldots, i_v)\, u(k-i_1), \ldots, u(k-i_v) \tag{2.5}$$

where M is referred to as the memory of the system. A linear convolution model is a particular case of the Volterra series where it is assumed that $h_0 = 0$ and $M = 1$

$$y(k) = \sum_{i=1}^{M} h_i u(k-i) \tag{2.6}$$

Generally, for control applications researchers have considered up to $N = 2$ (Parker and Doyle III, 2001) and the corresponding structure of the series is the following:

$$\hat{y}(k) = \sum_{n=0}^{M-1} h_n u(k-n) + \sum_{i=0}^{M-1} \sum_{j=1}^{M-1} h_{i,j} u(k-i)\, u(k-j) \tag{2.7}$$

The h_n, $n = 0, \ldots, (M-1)$ are the values of the linear Volterra series coefficients also known as first-order Volterra kernels, the $h_{i,j}$, $i = 1, \ldots, (M-1)$, $j = i, \ldots, (M-1)$ are the values of the quadratic terms of the Volterra series coefficients, also known as second-oder Volterra kernels. One of the main advantages of these models is that they provide a good approximation for a wide range of qualitative phenomena, but they cannot be formulated to exhibit output multiplicities or chaotic dynamics (Parker et al., 2001).

Additional details regarding the use of Volterra series models for control applications will be given in chapter 5.

2.2 Model predictive control

Model Predictive Control or Receding Horizon Control is a strategy that minimizes a cost function that considers the future errors with respect to the manipulated variables. Only the first manipulated variable move of the optimized profile is implemented. After the implementation, the optimization problem is solved again at the next time interval but taken in consideration the new conditions of the system. An overview of the earlier development and applications of MPC can be found at (Qin and Badgwell, 2003).

The objective function of a MPC controller is based on a norm that penalizes the deviation of the predicted controlled variables from its pre-specified set-points. The two norms that are commonly used on MPC applications are the two norm $||\cdot||_2$ and the infinity norm $||\cdot||_\infty$. For example, in continuous time the two norm objective function for a SISO system has the following form:

$$J = \min_{\Delta u(t)} \int_{t_0}^{\infty} \left(y^{\mathrm{sp}}(t) - \hat{y}(t)\right)^2 \mathrm{d}t \tag{2.8}$$

where $y^{\mathrm{sp}}(t)$ is the desired set-point, $\hat{y}(t)$ is the predicted output calculated from either a first-principles model or from an empirical model and $\Delta u(t)$ is the manipulated variable movement change.

In some cases instead of optimizing with respect to the manipulated variable change $\Delta u(t)$ the optimization is with respect to the manipulated variable value $u(t)$

$$J = \min_{u(t)} \int_{t_0}^{\infty} \left(y^{\mathrm{sp}}(t) - \hat{y}(t)\right)^2 \mathrm{d}t \tag{2.9}$$

Often the algorithm is applied in discrete time. In this case the objective function in terms of the manipulated variable movement change $\Delta u(k)$ can be rewritten as

$$J = \min_{\Delta u(k)} \sum_{k=k_0}^{\infty} \left(y^{\mathrm{sp}}(k) - \hat{y}(k)\right)^2 \tag{2.10}$$

or if the optimization is done with respect to the manipulated variable $u(k)$

$$J = \min_{u(k)} \sum_{k=k_0}^{\infty} \left(y^{\mathrm{sp}}(k) - \hat{y}(k)\right)^2 \tag{2.11}$$

As can be seen in equations (2.8) to (2.11), the integration or summation limits go from an initial time t_0 or initial time interval k_0 for the continuous and discrete case, respectively, to infinity. However, in practice the upper limit is selected to be finite and the prediction is done up to a prediction horizon with a length of p time intervals. Also, the optimization problem is formulated to calculate a specific number of control moves m in the future. The number of control moves is referred to as the control horizon m. For intervals after the $m-$th time interval it is assumed that the value of the manipulated variable is kept constant and is equal to $u(m)$ Following these definitions equation (2.10) is rewritten in the following way:

$$\begin{aligned} J &= \min_{\Delta u(k), \Delta u(k+1), \ldots, \Delta u(k+m)} \sum_{k=k_0}^{k_0+p} \left(y^{\mathrm{sp}}(k) - \hat{y}(k)\right)^2 \\ \text{subject to :} &\begin{cases} p \geq m \\ \Delta u(k+m+1) = \ldots = \Delta u(k+m+p) = 0 \end{cases} \end{aligned} \tag{2.12}$$

or if the optimization is with respect to $u(k)$

$$\begin{aligned} J &= \min_{u(k), u(k+1), \ldots, u(k+m)} \sum_{k=k_0}^{k_0+p} \left(y_{\mathrm{sp}}(k) - \hat{y}(k)\right)^2 \\ \text{subject to :} &\begin{cases} p \geq m \\ u(k+m) = u(k+m+1) = \ldots = u(k+m+p) \end{cases} \end{aligned} \tag{2.13}$$

The MPC formulation presents two clear advantages when compared to other control strategies:

1. Because the optimization considers the behavior of the system up to the prediction horizon p, it allows the controller to take an appropriate corrective action in response to a future predicted error (Eaton and Rawlings, 1992).
2. Input and output constraints can be explicitly considered within the optimization problem (Henson, 1998).

Constraints on the manipulated variables arise due to actuator and rate of change limits. Constraints on the output variables are due to equipment specifications, safety, environmental and economic considerations. After considering the optimization constraints, equation (2.10) can be reformulated as follows

$$J = \min_{\Delta u(k),\Delta u(k+1),\ldots,\Delta u(k+m)} \sum_{k=k_0}^{k_0+p} \left(y^{\text{sp}}(k) - \hat{y}(k)\right)^2$$

$$\text{subject to}: \begin{cases} p \geq m \\ \Delta u(k+m+1) = \ldots = \Delta u(k+m+p) = 0 \\ u_{\min} \leq u(i) \leq u_{\max},\ i = 1,\ldots,(k+m) \\ (\Delta u)_{\min} \leq \Delta u(i) \leq (\Delta u)_{\max},\ i = 1,\ldots,(k+m) \\ y_{\min} \leq y(j) \leq y_{\max},\ j = 1,\ldots,(k+p) \end{cases} \tag{2.14}$$

The problem in equation (2.10) can be easily generalized to multiple-input-multiple-output (MIMO) systems using the following representation:

$$J = \min_{\Delta\mathbf{u}(k),\Delta\mathbf{u}(k+1),\ldots,\Delta\mathbf{u}(k+m)} \left[\mathbf{y}^{\mathrm{T}}\mathbf{Q}\mathbf{y}\right] \tag{2.15}$$

or it the optimization is with respect to $\mathbf{u}(k)$

$$J = \min_{\mathbf{u}(k),\mathbf{u}(k+1),\ldots,\mathbf{u}(k+m)} \left[\mathbf{y}^{\mathrm{T}}\mathbf{Q}\mathbf{y}\right] \tag{2.16}$$

where $\mathbf{Q}$ is a positive definite weighting matrix, $\mathbf{y}$ is equal to

$$\mathbf{y} = \begin{bmatrix} y_1^{\mathrm{sp}}(k_0) - \hat{y}_1(k_0) \\ y_1^{\mathrm{sp}}(k_0+1) - \hat{y}_1(k_0+1) \\ \vdots \\ y_1^{\mathrm{sp}}(k_0+p) - \hat{y}_1(k_0+p) \\ y_2^{\mathrm{sp}}(k_0) - \hat{y}_2(k_0) \\ y_2^{\mathrm{sp}}(k_0+1) - \hat{y}_2(k_0+1) \\ \vdots \\ y_2^{\mathrm{sp}}(k_0+p) - \hat{y}_2(k_0+p) \\ \vdots \\ y_{n_y}^{\mathrm{sp}}(k_0+p) - \hat{y}_{n_y}(k_0+p) \end{bmatrix} \tag{2.17}$$

where n_y is the number of outputs, $\mathbf{u}$ is equal to

$$\mathbf{u} = \begin{bmatrix} u_1(k_0) \\ u_1(k_0+1) \\ \vdots \\ u_1(k_0+m) \\ u_2(k_0) \\ \vdots \\ u_2(k_0+m) \\ \vdots \\ u_{n_u}(k_0+m) \end{bmatrix} \tag{2.18}$$

where n_u is the number of inputs, and $\Delta\mathbf{u}$ is equal to

$$\Delta\mathbf{u} = \begin{bmatrix} \Delta u_1(k_0) \\ \Delta u_1(k_0+1) \\ \vdots \\ \Delta u_1(k_0+m) \\ \Delta u_2(k_0) \\ \vdots \\ \Delta u_2(k_0+m) \\ \vdots \\ \Delta u_{n_u}(k_0+m) \end{bmatrix} \tag{2.19}$$

To prevent excessive wear of the actuators or robustness issues related to aggressive control actions, a term that penalizes the control actions is added to the objective function as follows:

$$J = \min_{\Delta\mathbf{u}(k),\Delta\mathbf{u}(k+1),\ldots,\Delta\mathbf{u}(k+m)} \left[\mathbf{y}^{\mathrm{T}}\mathbf{Q}\mathbf{y} + \Delta\mathbf{u}^{\mathrm{T}}\mathbf{R}\Delta\mathbf{u}\right] \tag{2.20}$$

or if the optimization is with respect to $\mathbf{u}$

$$J = \min_{\mathbf{u}(k),\mathbf{u}(k+1),\ldots,\mathbf{u}(k+m)} \left[\mathbf{y}^{\mathrm{T}}\mathbf{Q}\mathbf{y} + \mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u}\right] \tag{2.21}$$

where $\mathbf{R}$ is a positive definite weighting matrix.

In MPC the prediction horizon p, control horizon m and weight matrices $\mathbf{Q}$ and $\mathbf{R}$ are tuning parameters. If these parameters are not tuned correctly the system might be closed-loop unstable (Chen and Allgöwer, 1998b). Instability is generally related to model error or the presence of right hand plane (RHP) zeros. Also (Muske and Rawlings, 1993b,a) have proven the theoretical nominal stability of MPC controllers with an infinite horizon. However, in practice infinite horizon based controllers are generally conservative. This fact is one of the main reasons that motivated the search for algorithms with guaranteed stability for finite prediction and control horizons (Findeisen and Allgöwer, 2002). In general proposed algorithms with guaranteed stability use a series of extra terms and restrictions within the objective function. One of the simplest algorithms consists in solving the following optimization problem

$$\begin{aligned} J = \min_{\mathbf{u}(k),\mathbf{u}(k+1),\ldots,\mathbf{u}(k+m)} & \left[\mathbf{y}^{\mathrm{T}}\mathbf{Q}\mathbf{y} + \mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u}\right] \\ \text{subject to} : & \ \mathbf{y}_{\text{final}} = \mathbf{0} \end{aligned} \tag{2.22}$$

where

$$\mathbf{y}_{\text{final}} = \begin{bmatrix} y_1^{\text{sp}}(k_0+p) - \hat{y}_1(k_0+p) \\ y_2^{\text{sp}}(k_0+p) - \hat{y}_2(k_0+p) \\ \vdots \\ y_{n_y}^{\text{sp}}(k_0+p) - \hat{y}_{n_y}(k_0+p) \end{bmatrix} \tag{2.23}$$

In the previous formulation the optimization forces the system to return to the origin at the end of the prediction horizon due to the constraint given in equation (2.22).This type of constraint is usually referred to as a terminal condition. The disadvantage of this approach is that in some cases the control actions can be too aggressive resulting in time consuming calculations and possible infeasibility. Another formulation that can be used is the following:

$$\begin{aligned} J = \min_{\mathbf{u}(k),\mathbf{u}(k+1),\ldots,\mathbf{u}(k+m)} & \left[\left[\mathbf{y}^{\mathrm{T}}\mathbf{Q}\mathbf{y} + \mathbf{u}^{\mathrm{T}}\mathbf{R}\mathbf{u}\right] + \mathbf{y}_{\text{final}}^{\mathrm{T}}\mathbf{P}\mathbf{y}_{\text{final}}\right] \\ \text{subject to} : & \ \mathbf{y}_{\text{final}} \in \Omega \end{aligned} \tag{2.24}$$

In this case instead of requiring that at the end of the prediction horizon $\mathbf{y}_{\text{final}} = \mathbf{0}$, it is required that $\mathbf{y}_{\text{final}}$ is within some neighborhood of the origin. Also the matrix $\mathbf{P}$ is selected as the covariance matrix of a Lyapunov candidate function. The extra term in the objective function forces the system to reach a neighborhood Ω near the origin. If the region Ω is properly selected then the system will follow a trajectory that converges to the origin (Chen and Allgöwer, 1998a,b), (Mayne and Michalska, 1990) and (Findeisen and Allgöwer, 2002) resulting in asymptotic stability. It must be noted that the MPC methods with terminal condition also assume an exact knowledge of the system parameters. However, if the system parameters are not exactly known, the imposition of terminal conditions is not sufficient to ensure closed-loop stability. In a recent comprehensive review of nonlinear predictive control, (Findeisen and Allgöwer, 2002) identified the issue of robustness to uncertainty as one of the key challenges that remain for effective implementation of NMPC in industry. Robustness of NMPC is one of the key subjects of the current thesis.

2.2.1 Linear MPC

When the process is operated in a close neighborhood of a particular set of operating conditions, a linear model may be suitable to describe its behavior. Two types of linear models have been considered for predicting the value of $\hat{\mathbf{y}}$ within linear model-based predictive control strategies: step-response or state-space model. Based on those models two main linear MPC (LMPC) algorithms may be derived. The LMPC with step-response model has been used by (Cutler and Ramaker, 1980) whereas LMPC based on state-space model have been presented in (Maciejowski, 2002).

The main disadvantage of the step-response in linear model-based MPC is that the dimensions of the matrices required for obtaining the MPC controller are a function of p, m, n_y and n_u thus limiting its application to low dimension problems. For example, (Gao, 2004) have previously presented a gain-scheduled MPC strategy that is based on a set of linear MPC controllers where each of these was designed based on a linear step-response model. The use of step-response models was shown to seriously limit the capability of the controller since the memory and computational requirements were fairly high even for moderate prediction horizons of the order of 10. On the other hand, state-space models do not suffer from that disadvantage and therefore they were used in this thesis for the design of gain-scheduled controllers. The details of the construction of the state-space based LMPC are given in section 3.2. It will be shown that the state-space formulation has explicitly removed the limitations of prediction horizons previously encountered by (Gao, 2004). Since most chemical processes are inherently nonlinear the linear model-based formulations will not result in optimal performance thus motivating the use of nonlinear models for controller design as described in the following section.

2.2.2 Nonlinear MPC

When the process model is nonlinear the equations to predict the output take the form:

$$\begin{aligned}\dot{\mathbf{x}} =& f\left(\mathbf{x},\mathbf{u}\right)\\ \hat{\mathbf{y}} =& g\left(\mathbf{x},\mathbf{u}\right)\end{aligned} \tag{2.25}$$

where f and g are nonlinear vector fields that can be obtained from a first-principles or from a nonlinear empirical model of the process. In these cases equations (2.20) or (2.21) give origin to a nonlinear optimization problem.

One possibility to handle nonlinear systems consists in obtaining a linear model around a given operating point. Using that linear model an MPC controller can be designed. Local linearization is the basis for constructing a gain-scheduling MPC (GSMPC), as will be explained in the next chapter.

As opposed to the standard LMPC case, different NMPC controller formulations have been proposed in the literature. However, there are some common elements shared by these procedures as follows:

1. Obtain either a first-principles or an empirical model of the process studied.
2. Select a set of values for p, m, $\mathbf{Q}$ and $\mathbf{R}$.
3. Solve the nonlinear optimization problem of equations (2.20) or (2.21). Use the model obtained in step 1 to calculate the output predictions of equation (2.25).
4. Implement on the system the first value of the manipulated variable profile obtained in 3.
5. Obtain the new measurements of the system.
6. Go back to 3 and repeat the cycle.

Nonlinear MPC has been applied successfully to a number of processes described by either a first-principles or an empirical model. The key difficulty with first-principles models is that generally they are difficult to obtain for several reasons as follows:

- Unknown parameters due to biological, chemical or physical constants which are difficult to measure.
- Unknown microscopic interactions.
- Unknown interactions between several systems.
- Unknown representative model structure.

On the other hand, if a first-principles model is available, it may contain complex high-order ordinary and/or partial differential equations that are very time consuming to solve if used for on-line optimization within the MPC framework. Also, these models are, as explained above, not suitable for the formulation of analytical robustness tests. These are the main motivations for employing an empirical model within the MPC formulation. Accordingly, the next section presents a list of chronologically ordered studies where NMPC have been applied using empirical models.

(Hérnandez and Arkun, 1993) used a polynomial ARMA model of an open-loop unstable continuous stirred tank reactor (CSTR). The NMPC was used for set-point tracking and disturbance rejection; in both cases it showed good performance. To compensate for model mismatch when the system was operated outside the calibration region an estimator was used similar to the one proposed by (Sistu and Bequette, 1991). The ARMA coefficients were obtained from input-output data. No formal analysis of robustness was attempted in this work and the design of the controller was solely based on extensive simulations.

(Doyle III et al., 1995) used a second order Volterra series model of a CSTR. The work compared the performance of a NMPC and a LMPC for set-point tracking. The results showed that NMPC surpasses the performance of LMPC. The effect of parameter uncertainty was studied through extensive simulations; in this case the NMPC performance also surpassed that of LMPC. The Volterra series model coefficients were obtained by Carlemann linearization.

(Maner et al., 1996) used a second order Volterra series model of a CSTR in a MIMO configuration. The NMPC was compared to a LMPC for set-point tracking. It was shown that the NMPC performance was superior to that of LMPC. The Volterra series model coefficients were also obtained by Carlemann linearization.

(Fruzzetti et al., 1997) used Hammerstein models for representing a pH neutralization and a binary distillation process. The work compared the performance of a NMPC with a LMPC. In both cases the NMPC improvement was clearly superior to the LMPC. The NMPC showed no oscillations, less settling time and less overshoots and undershoots. The Hammerstein model coefficients were obtained from input-output data. Robustness issues were not considered and tuning was based on simulations.

(Norquay et al., 1999) used a Wiener model of a pH neutralization process. The work compared the performance of NMPC, LMPC and a proportional-integer (PI) controller for set-point tracking and disturbance rejection. It was shown by simulations that the performance achieved using the NMPC controller surpassed the performance obtained by using a LMPC or a PI controller.

(Gerkšič et al., 2000) used a Wiener model of a pH neutralization process. The work showed that the performance achieved using a NMPC was superior compared to a LMPC. It was shown that if there is a mismatch between the model and the process, a NMPC without state estimation can still be used to obtain acceptable performance.

(Jeong et al., 2001) used a Wiener model of a continuous polymerization reactor. The work compared the performance of a NMPC with a LMPC. In the region where the process had a linear behavior there were no differences between the performances achieved using a NMPC or a LMPC. However, when the process was operated in zones where the system exhibited nonlinear behavior the NMPC showed a clear improvement over a LMPC. Additionally the NMPC showed no oscillations and no steady-state bias.

2.3 Robust control

Whether a first-principles model or an empirical model is used, there will always be some degree of inaccuracy in the model. Effective control strategies need to consider this model variation to provide a controller that will not be affected by the model inaccuracy. If the control strategy does not consider this inaccuracy, the controller based on the model may result in poor closed-loop performance when it is applied to the real process. In extreme cases, the closed-loop may become unstable.

In robust control it is assumed that there is some degree of inaccuracy between the process model and the real plant. Therefore, a control strategy designed based on robust control methods is expected to achieve better results when compared with controllers that do not consider model error.

Model uncertainty is generally categorized into two types, unstructured and structured.

For unstructured uncertainty:

- Several sources of uncertainty are lumped together.
- Generally results in conservative controllers.

For structured uncertainty:

- The individual sources of uncertainty are identified and represented directly.
- Results in less conservative controllers, as compared to those obtained when the uncertainty is considered to be unstructured.

The uncertainty can also be real valued to represent variations in the model parameters or complex, to represent uncertain dynamic behavior in the frequency domain.

After a model is obtained robust control methods can be used to analyze it. The first step consists in identifying the sources of model uncertainty. The second step consists in using a test to check if the system has the following properties: nominal stability, robust stability, nominal performance and robust performance.

Nominal stability and performance tests refer to the stability and performance of the nominal model whereas robust stability and performance tests refer to the stability and performance of the family of models defined by the nominal model with their associated model uncertainty.

The robust control tools considered in this thesis to perform the stability and performance tests are based on LMI's or on the SSV. These methods are reviewed in the following subsections.

2.3.1 Linear matrix inequalities based test

The stability and performance problems can be formulated as a solution to a set of linear matrix inequalities. LMI's can also be used to account for input and output constraints (Kothare et al., 1996).

The LMI framework considers matrix inequalities of the following form:

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + \sum_{i=1}^{m} \mathbf{x}_i \mathbf{F}_i > 0 \tag{2.26}$$

where $\mathbf{F}_j, j = 0, \ldots, m$ are symmetric real matrices defined by the problem statement, $\mathbf{x} \in \Re^m$ is a vector of variables and $\mathbf{F} > 0$ is positive definite.

When the problem involves several LMI's the structure of the problem can be equivalently transformed to a single higher dimensional LMI. For example, if the system of LMI's has the following structure

$$\begin{gathered} \mathbf{F}^{(1)}(\mathbf{x}) > 0 \\ \vdots \\ \mathbf{F}^{(p)}(\mathbf{x}) > 0 \end{gathered} \tag{2.27}$$

the LMI system can be expressed as:

$$\operatorname{diag}\left(\mathbf{F}^{(1)}(\mathbf{x}), \ldots, \mathbf{F}^{(p)}(\mathbf{x})\right) > 0 \tag{2.28}$$

The generic LMI problem used in the current thesis is the feasibility problem. An LMI feasibility problem seeks a feasible solution, i.e., $\mathbf{x} = (x_1, \ldots, x_m)$ such that equation (2.27) is true.

Section 2.4.1 will present the application of LMI's to the model predictive control formulation. This leads to the design of a robust gain-scheduled MPC controller based on a LMI formulation.

2.3.2 Structured singular value

The structured singular value analysis referred also as to μ analysis considers a plant model that is subject to unstructured or structured uncertainty. It also considers that there is an interconnection between the model structure represented by $\mathbf{M}$ and its uncertainties represented by $\boldsymbol{\Delta}$ through a linear fractional transformation or LFT. Details of the construction of $\mathbf{M}$ and $\boldsymbol{\Delta}$ for control applications are given in section 5.3.

The analysis is based on the calculation of the structured singular value according to the uncertainty description $\boldsymbol{\Delta}$. The SSV is used to test the robust stability and robust performance properties of the system.

The definition of the SSV of a matrix $\mathbf{M} \in \mathbf{C}^{n \times n}$ with respect to an uncertainty structure $\boldsymbol{\Delta}$ is as follows:

$$\mu_{\boldsymbol{\Delta}}(\mathbf{M}) = \frac{1}{\min_{\Delta \in \boldsymbol{\Delta}} \left[\overline{\sigma}(\Delta) \mid \det(\mathbf{I} - \mathbf{M}\boldsymbol{\Delta}) = 0 \right]} \tag{2.29}$$

unless there is no $\Delta \in \boldsymbol{\Delta}$ which makes $\mathbf{I} - \mathbf{M}\boldsymbol{\Delta}$ singular, in which case $\mu_{\Delta}(\mathbf{M}) = 0$, (Packard and Doyle, 1993). The SSV is a measure of the size of the uncertainty that is required in order to destabilize the closed-loop system represented by $\mathbf{M}$. Consequently for each problem it is necessary to obtain the appropriate $\mathbf{M}$ and $\boldsymbol{\Delta}$ matrices to perform the analysis.

Because the calculation of the SSV makes explicit use of the structure of $\boldsymbol{\Delta}$, less conservative controllers result (Bates and Postlethwaite, 2002). Since the uncertainty matrix $\boldsymbol{\Delta}$ may be composed of both unstructured and structured uncertainty elements it is described by the following general form:

$$\begin{aligned}\mathbf{\Delta} = &\left[\operatorname{diag}\left[\delta_1 \mathbf{I}_{r1}, \ldots, \delta_S \mathbf{I}_{rS}, \mathbf{\Delta}_{S+1}, \ldots, \mathbf{\Delta}_{s+F}\right] :\right. \\ &\left.\delta_i \in \mathbf{C}, \mathbf{\Delta}_{S+j} \in \mathbf{C}^{m_j \times m_j}, 1 \leq i \leq S, 1 \leq j \leq F\right]\end{aligned} \tag{2.30}$$

where S and F are the number of repeated scalar blocks and the number of full block respectively.

2.4 Robust model predictive control

Robust model predictive control designs depend on a nominal model with its uncertainty description. The combination of the nominal model and an uncertainty description is used, for analysis purposes, to generate a family of models. Then, the main problem in robust MPC consists in analyzing each model within this family to test whether the system has nominal stability, robust stability, nominal performance and robust performance properties (Campo and Morari, 1987). Robust MPC designs are generally based on the worst model in the set. If the worst model has the four properties, then it can be inferred that any other model within the uncertainty set will also have these properties. The next subsections contain a review of robust LMPC and robust NMPC methods.

2.4.1 Robust LMPC

Based on the mathematical background that was developed for linear systems, the following research works used the available theories and tools to propose several robust LMPC formulations.

(Campo and Morari, 1987) proposed a min-max formulation that minimizes the MPC objective function for the worst plant. The model uncertainty is associated with the finite impulse response coefficients (FIR). The robustness tests consider the error infinity norm of the worst plant to check if it stays within specific bounds. The closed-loop system satisfies robust stability and robust performance if the error infinity norm is within pre-specified bounds. Because the FIR model is linear, the optimization problem can be formulated as a linear programming (LP) problem simplifying its solution.

(Zafiriou, 1990) proposed a robust MPC method based on the contraction principle. The method considers that the uncertainty is associated with the FIR coefficients used to model the process. The system requires obtaining an operator F that maps the system state at sampling instant k to the system state at sampling instant $k + 1$. Asymptotic stability for all the sets of FIR values considered is tested by checking whether the infinity norm of F is always decreasing. Due to the use of the contraction mapping concept (Zanovello and Budman, 1999) found this formulation to be very conservative.

(Kothare et al., 1996) developed a robustness test based on LMI's. The method requires a process model and its uncertainty description which can be represented by the following two models: polytopic or multi-model and structured feedback. The objective function solves a min-max problem where the cost function is maximized with respect to the worst plant and minimized with respect to the manipulated variables. Because the proposed formulation considers that $p = \infty$ it has guaranteed nominal stability (Muske and Rawlings, 1993a) and (Rawlings and Muske, 1993) provided the process is open-loop stable.

(Badgwell, 1997) used a linearized model of the plant and proposed a method that accounts for constraints. The constraints are formulated as terminal conditions that force each plant in the set to end up at steady-state within a certain region about the origin and by doing this robust stability of the closed-loop system was ensured.

(van den Boom, 1997) applied the LMI formulation of (Kothare et al., 1996) to nonlinear systems by using a feedback linearization. Then the resulting system of LMI's was solved for robust stability and robust performance to obtain the feedback law. This algorithm was found to be efficient in the vicinity of the operating point used for the linearization. The key disadvantage of this approach is that it requires a nonlinear transformation to obtain a linear model from the original nonlinear model which is not always available.

(Wan and Kothare, 2002) developed a procedure that adds a group of LMI's to the original formulation of (Kothare et al., 1996) when an observer is required to estimate the states.

(Wang and Rawlings, 2004a,b) modified the LMI problem formulation of (Kothare et al., 1996) to work with systems represented by a linear ARMAX model. This algorithm works by considering a finite number of models in a branching type algorithm. Therefore, as the number of branches and the settling time of the process increases, the complexity of the problem also increases.

The main disadvantage of all the previous methods is that the models used for output prediction are linear. The best approach so far to the problem of finding a robust MPC with nonlinear prediction model consists in linearizing the model and use one of the methods reviewed above.

2.4.2 Robust NMPC

Robustness tests often require that the model uncertainty can be mathematically factorized from the nominal model. This requirement rules out the possibility to formally address robustness of predictive controllers based on first-principles models since in these models the parameters often appear in expressions that are not amenable to the required factorization. For example, activation energy within an exponential Arrhenius term cannot be effectively separated into a nominal and uncertain part for the purpose of robustness analysis. This explains why there are a limited number of robust NMPC algorithms and the approach that is generally used to design them consists in solving a min-max problem. However, the main problem of the algorithms is that the min-max formulation is based on solving extensive simulations that are computationally intensive and time consuming.

The studies that have proposed a robust NMPC algorithm can be classified in two main groups: algorithms that are based on simulations and algorithms that propose a formal analytical test. Some examples of algorithms that consider both approaches are:

(Kawohl et al., 2007) proposed a methodology based on calculating the first and second statistical moments that describe the variation of the objective function. To calculate the moments the algorithm uses a Monte Carlo simulation approach. The NMPC controller is based on a first-principles model of the system, additionally it is considered that the controller is employed on a batch process.

(Magni and Scattolini, 2007) presented a review of methods that require calculating a Lyapunov candidate function for uncertain systems. Only the theoretical bases were presented but not specific examples were given. The use of Lyapunov functions generally leads to conservative designs.

(Diehl et al., 2008) proposed an algorithm that requires computing the derivatives of the objective function, constraints and uncertainty set. The algorithm assumes that the worst value of the objective function is achieved at the bounds of the uncertainty set. The methodology was tested on a batch process and it was assumed that the first-principles model of the process was available.

(Zavala and Biegler, 2009) proposed an algorithm that considers uncertainty in the evaluation of the cost function using nonlinear programming (NLP) sensitivity concepts. The methodology was applied to a CSTR and it was assumed that a first-principles model of the process was available.

The only analytical based approach to the problem proposed by (Ma and Braatz, 2001) and (Nagy and Braatz, 2003) uses an analytical algorithm based on the SSV to calculate the worst cost function at the end of a time horizon. The objective of this study was not control per se but rather optimization of the end point of a batch process. The cost function was based on first and second orders Taylor expansions of the nonlinear equations governing the process. The analytical test proposed by (Nagy and Braatz, 2003) was used in the current thesis to formulate a predictive controller for continuous processes.

The absence of a systematic approach for designing robust nonlinear predictive controllers based on first-principles models has motivated the use in this thesis of particular nonlinear empirical models that can be used to formulate analytical robustness tests.

Chapter 3

Robust state-affine model-based gain-scheduling MPC [*]

Overview

A methodology is proposed to design a robust gain-scheduled model predictive control (MPC) strategy and to quantify the relative advantages of this controller versus a linear MPC strategy. For the purpose of analysis and controller design the process is represented by a nonlinear state-affine model identified from input-output data. This model can be split in linear and nonlinear terms where the linear part is used for controller design and the nonlinear part is accounted for as model uncertainty. Then, robust stability and robust performance tests are formulated based on linear matrix inequalities where the manipulated variables weight of the controllers is tuned to maximize performance. The uncertainty bounds used for the robustness tests are obtained in an iterative fashion by using the frequency response of the manipulated variable with respect to the feedback error. The control strategy performance is quantified by the ratio between the error norm and the disturbance norm. Finally, a case study involving a multiple-input-multiple-output bioreactor is presented. The study is able to predict for which range of operation the gain-scheduling MPC surpasses the performance of the linear MPC.

[*]Adapted from Díaz-Mendoza, R., Gao, J. and Budman, H. Methodology for designing and comparing linear versus gain-scheduled model predictive controllers, *Industrial & Engineering Chemistry Research*, 2009, 48, 22, 9985–9998.

3.1 Introduction

Model predictive control (MPC) is a widely used control algorithm in the chemical industry. In most current industrial applications, linear MPC controllers are used that are based on a single linear model of the process. However, control systems that provide optimal performance for a particular linear model may perform poorly when implemented on a nonlinear system (Zheng and Morari, 1993). Due to the process nonlinearity, a system behaves differently when operated at different operating conditions. Therefore, controllers that are based on one single linear model have to be detuned to achieve robustness to model error that arises from the differences between the linear model used for controller design and the actual nonlinear process behavior.

The basic philosophy in the literature for optimizing the performance of MPC algorithms in the presence of plant-model mismatch is to modify the online minimization problem to a min-max problem, where the worst case value of the objective function is minimized over the set of plants that account for the nominal model and uncertainty (Campo and Morari, 1987) and (Zheng and Morari, 1993). Clearly, this approach is computationally more demanding than solving the optimization problem for a single nominal model, and also the resulting controllers tend to be conservative. Nonlinear model predictive control (NMPC) algorithms have been proposed to explicitly address the nonlinearity of the process and to improve the closed-loop performance (Findeisen et al., 2003). However, it is more difficult to guarantee robust stability and robust performance for these controllers as compared to linear MPC controllers, and a nonlinear mechanistic model of the process is required that is often difficult to obtain.

To avoid the conservatism of robust linear controllers or the added complexity of a nonlinear controller algorithm, a widely accepted approach in the chemical industry has been the use of gain-scheduled controllers. These controllers are typically designed on the basis of a set of linear models that result in a corresponding set of linear controllers for any particular process. At each interval of time, one of these linear controllers is activated based on the value that the scheduling variables achieve during closed-loop operation. Different criteria are used for selecting the scheduling variable; e.g., this variable should change significantly with changes in operating conditions and should efficiently capture the system nonlinear behavior (Bequette, 2003), (Rugh and Shamma, 2000) and (Shamma and Athans, 1990). In this work, the manipulated variable has been selected as the scheduling variable since the nonlinear terms of the nonlinear model used to represent the system are powers of the manipulated variable values. Although gain-scheduled controllers compensate better for the system's nonlinearity than linear controllers, it will be shown in this work that they must be tuned for robustness to model error. The case study included in this chapter will illustrate that the lack of robustness of the local controllers that form the gain-scheduling strategy may lead to bad performance or even instability. In particular (Rugh and Shamma, 2000) pointed out that the design of robust gain-scheduling controllers can be approached by using linear parameter varying (LPV) models, and the analysis can be conducted using linear matrix inequalities (LMI's). However, the specific problem of designing robust gain-scheduled MPC algorithms has not been studied in a systematic fashion. In particular, previous research has not provided computationally efficient tools to predict whether a gain-scheduled MPC controller will perform better that one single linear MPC controller for a wide range of operating conditions and external disturbances. The ability to quantify the performance of a linear MPC controller versus a gain-scheduled MPC controller could allow the practitioner to make an informed decision regarding the need for a gain-scheduled MPC strategy.

This chapter discusses a systematic approach for the design of a robust gain-scheduled MPC based on the quantification of the closed-loop performance in the presence of model error due to nonlinearity and model structure errors. The relationship between the root mean square (RMS) of the deviation of the controlled variables from the set-point by the RMS of the inputs affecting the process is used in this work for quantification and optimization of the closed-loop performance, and it is also used as a basis for comparison of the gain-scheduled MPC controllers with linear MPC controllers. For this work the RMS was calculated as:

$$s_{\text{rms}} = \sqrt{\frac{s(1) + s(2) + \ldots + s(n)}{n}} \tag{3.1}$$

where s is a signal. Since a main source of model error between the linear models used for control and the actual system is due to nonlinearity, the RMS index used in this work reflects the impact of the nonlinearity on the closed-loop operation. The quantification of nonlinearity and its impact on control have been studied by different researchers (Nikolaou and Misra, 2003) and (Sheweickhardt and Allgöwer, 2007). One way to quantify the effect of nonlinearity on the closed-loop performance is by a brute force nonlinear simulation-based search of the worst case scenario among the infinite possible combinations of disturbances and controller tuning parameters. However, this is computationally prohibitive. Instead, this work discusses a systematic approach to quantify the effect of closed-loop nonlinearity, and although it is limited, as shown later, to a particular type of empirical models, it provides a useful and easy to compute bound for comparison between linear and gain-scheduled MPC controllers.

The models used for designing robust MPC controllers can be based on first-principles equations, or they can be empirical. Although first-principles models provide a better representation of the process behavior, it is often difficult to obtain them, and they are generally too complex for the purpose of robustness analysis. Therefore, it is assumed in this work that a mechanistic model of the process to be controlled is not available to the designer, and therefore an empirical model has to be used instead which could be directly identified from experimental data. Nonlinear state-affine models have been proposed in the past as a general model structure for the representation of nonlinear systems, and they have been used for the study of observability and controllability of nonlinear processes (Sontag, 1979). These models are given by the following equations

$$\mathbf{x}\left(k+1\right)=\left[\mathbf{F}_0+\sum_{i=1}^{n_F-1}\sum_{j=1}^{n_u}\mathbf{F}_{i,j}\left(u_j\left(k\right)\right)^i\right]\mathbf{x}\left(k\right)+\left[\mathbf{G}_1+\sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\mathbf{G}_{i+1,j}\left(u_j\left(k\right)\right)^i\right]\mathbf{u}\left(k\right) \tag{3.2}$$

$$\mathbf{y}\left(k\right)=\mathbf{C}^{\mathrm{un}}\mathbf{x}\left(k\right)+\mathbf{W}_F\mathbf{d}\left(k\right) \tag{3.3}$$

where $\mathbf{u}\left(k\right)$ is the current manipulated variable vector; $\mathbf{y}\left(k\right)$ is the current controlled variable vector; $\mathbf{d}\left(k\right)$ is a normalized unmeasured disturbance; $\mathbf{W}_F$ is the magnitude of the normalized disturbance; and n_F and n_G are the number of $\mathbf{F}$ and $\mathbf{G}$ matrices, respectively, that are required to model the process. It has been shown that these models can be easily obtained from input-output process data (Sontag, 1979), in this work the parameters of the state-affine model were obtained from a straightforward least-squares calculation. Using this fact (Gao and Budman, 2004) proposed the use of these types of models for the purpose of design and robustness analysis of gain-scheduled proportional-integral-derivative (PID) controllers. They showed that a key advantage in using these models is that they permit a straightforward representation of the nonlinearity as model error terms that can be later used for robust control design. However, key limitations of the previous work (Gao and Budman, 2004) were: (i) the controller was limited to a simple PID and (ii) the bounds on the manipulated variables $\mathbf{u}(k)$ were assumed a priori based on input saturation limits, and this assumption resulted in conservative controllers. (Gao, 2004) also used the state-affine model to design and analyze a robust gain-scheduled step-response based MPC. However, the key limitations were: (i) the bounds on the manipulated variables $\mathbf{u}(k)$ were also assumed a priori and (ii) by using the step-response model to calculate the MPC the system dimensions are a function the prediction horizon p, number of inputs n_u, number of outputs n_y and n_F, n_G. In this work, a robust control design procedure will be given whereby the bounds of the manipulated variables during closed-loop operation with gain-scheduled MPC (GSMPC) controllers can be calculated from an iterative procedure based on the magnitude of the disturbances, additionally to decrease the system dimensions the procedure will be based in a state-space MPC. Table 3.1 shows the main differences between (Gao, 2004) and the current work.

This chapter is organized as follows: section 3.2.1 develops the closed-loop state-space formulation of the system that is obtained from the combination of the nominal linear model plus uncertainty and the MPC controller. Thus, the closed-loop equations are formulated as an affine parameter-dependent system. Section 3.3 provides LMI formulations of robust stability and robust performance tests for the closed-loop model obtained in section 3.2.3. Section 3.4 describes the main contributions of the work including the procedure to quantify the uncertainty bounds and the method for designing an optimal robust gain-scheduled MPC. In section 3.5 the proposed approach is applied to a multiple-input-multiple-output (MIMO) bioreactor process, and an extensive comparison of a linear MPC

Table 3.1: List of differences between (Gao, 2004) and the current work

	(Gao, 2004)	Current work
Model MPC controller	step-response	state-space
System dimension's	increase with p (memory limitations)	independent of p
Uncertainty bounds	prespecified (conservative)	obtained iteratively (less conservative)

controller versus a gain-scheduled MPC algorithm is conducted for this problem. Finally, results and conclusions are presented in section 3.6.

3.2 State-Space formulation of the closed-loop system

3.2.1 Model uncertainty and disturbances

When the process represented by equations (3.2) and (3.3) is operated within a small neighborhood of the origin, i.e., $\mathbf{u}(k)$ is very small, the process can be described by the linear part of the model as follows

$$\mathbf{x}^{\mathrm{L}}(k+1) = \mathbf{F}_0\mathbf{x}^{\mathrm{L}}(k) + \mathbf{G}_1\mathbf{u}(k) \tag{3.4}$$

$$\mathbf{y}^{\mathrm{L}}(k) = \mathbf{C}\mathbf{x}^{\mathrm{L}}(k) \tag{3.5}$$

where the subscript L denotes linear. Then, for the purpose of robust control analysis, the nonlinear terms consisting of the second and higher-order powers of $\mathbf{u}(k)$ in equation (3.2) can be accounted for as model errors or uncertainties between a nominal linear model, defined by the matrices $\mathbf{F}_0$ and $\mathbf{G}_1$, with respect to the full nonlinear model of the system given by equations (3.2) and (3.3). Accordingly a model uncertainty perturbation $\delta_{i,j,k}$ is defined as follows

$$\delta_{i,j,k} = \left(u_j\left(k\right)\right)^i,\ i = 1,\dots,\max\left(n_F, n_G\right),\ j = 1,\dots,n_u \tag{3.6}$$

In general, it is not trivial to quantify the uncertainty from mechanistic first-principles models, and often the solution of complex optimization problems is required to identify this uncertainty (Doyle III et al., 1989). Thus, a key advantage in considering the uncertainty terms to be equal to the powers of the manipulated variable values according to equation (3.6) is that from this description, the uncertainty can be easily accounted for. Lower and upper limits on the manipulated variables are either related to saturation limits, or may be obtained from analytical bounds as it will be shown in a later section in this chapter. These bounds are represented as follows

$$u_j\left(k\right) \in \left[u_j^{\text{lb}}, u_j^{\text{ub}}\right],\ j = 1,\dots,n_u \tag{3.7}$$

Since the nonlinear state-affine model given by equations (3.2) and (3.3) is only an empirical approximation of the actual process, model error is also expected between the nonlinear state-affine model and the actual process. To account for this modeling error an output uncertainty $\boldsymbol{\delta}^{\text{out}}$ is added to the process output equation as follows:

$$\mathbf{C}^{\text{un}} = \mathbf{C} + \boldsymbol{\delta}^{\text{out}}\mathbf{I} \tag{3.8}$$

where $\boldsymbol{\delta}^{\text{out}}$ is real and contained between a lower value $\boldsymbol{\delta}^{\text{out-lb}}$ and an upper value $\boldsymbol{\delta}^{\text{out-ub}}$

$$\boldsymbol{\delta}^{\text{out}} = \operatorname{diag}\left[\delta_1^{\text{out}},\dots,\delta_{n_y}^{\text{out}}\right] \tag{3.9}$$

where $\boldsymbol{\delta}^{\text{out}}$ is obtained from open-loop comparisons of the nonlinear state-affine model given by equations (3.2) and (3.3) and the actual nonlinear process as per the following maximization problem

$$\boldsymbol{\delta}^{\text{out}} = \max\left[\left|\mathbf{y}^{\text{process}} - \mathbf{y}^{\text{model}}\right|\right] \tag{3.10}$$

In practice, if equations (3.2) and (3.3) are identified to accurately describe the actual nonlinear process, the resulting $\boldsymbol{\delta}^{\text{out}}$ is expected to be very small. It is also assumed that external disturbances are affecting the process output. To limit the bandwidth of the disturbances entering the system, a first order representation of the disturbance was assumed as follows

$$\mathbf{d}\left(k+1\right) = \text{BW}\mathbf{d}\left(k\right) + \left(1-\text{BW}\right)\nu\left(k\right) \tag{3.11}$$

where BW is the filter parameter. This filtering is essential because it is impossible to satisfy robustness with respect to model uncertainty for a disturbance of infinite bandwidth.

For example, if $n_u = 2$ and the model consider up to the square powers of $\mathbf{u}$ and in the presence of truncation error and disturbances, equations (3.2) and (3.3) can be written as follows

$$\mathbf{x}\left(k+1\right) = \left[\mathbf{F}_0 + \mathbf{F}_{1,1}\delta_{1,1,k} + \mathbf{F}_{1,2}\delta_{1,2,k}\right]\mathbf{x}\left(k\right) + \left[\mathbf{G}_1 + \mathbf{G}_{2,1}\delta_{1,1,k} + \mathbf{G}_{2,2}\delta_{1,2,k}\right]\mathbf{u}\left(k\right) \tag{3.12}$$

$$\mathbf{y}\left(k\right) = \left[\mathbf{C} + \boldsymbol{\delta}^{\text{out}}\mathbf{I}\right]\mathbf{x}\left(k\right) + \mathbf{W}_F\mathbf{d}\left(k\right) \tag{3.13}$$

3.2.2 State-Space formulation of the MPC controller

Since the gain-scheduled MPC strategy is composed of a family of linear MPC controllers, the equations for a state-space version of a linear unconstrained MPC controller are summarized in this section. Consider a MIMO system with n_x states, n_u inputs, and n_y outputs that will be controlled by a MPC controller, with prediction horizon p and control horizon m. The model-based prediction equation can be written as follows (Maciejowski, 2002)

$$\begin{bmatrix} \hat{\mathbf{y}}\left(k+1|k\right) \\ \hat{\mathbf{y}}\left(k+2|k\right) \\ \vdots \\ \hat{\mathbf{y}}\left(k+p|k\right) \end{bmatrix} = \boldsymbol{\Psi}\mathbf{x}^{\text{L}}\left(k\right) + \boldsymbol{\Gamma}\mathbf{u}\left(k-1\right) + \boldsymbol{\Theta}\begin{bmatrix} \Delta\mathbf{u}\left(k\right) \\ \vdots \\ \Delta\mathbf{u}\left(k+m-1\right) \end{bmatrix} \tag{3.14}$$

where the vector $\Delta\mathbf{u}$ and the matrices $\mathbf{\Psi}$, $\mathbf{\Gamma}$ and $\mathbf{\Theta}$ are defined as follows

$$[\Delta\mathbf{u}(k)]_{n_u\times 1} = \mathbf{u}(k) - \mathbf{u}(k-1) \tag{3.15}$$

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{CF}_0 \\ \mathbf{CF}_0^2 \\ \vdots \\ \mathbf{CF}_0^p \end{bmatrix}_{(p\times n_y)\times n_x} \tag{3.16}$$

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{CG}_1 \\ \mathbf{CF}_0\mathbf{G}_1 + \mathbf{CG}_1 \\ \vdots \\ \sum_{i=0}^{p-1}\mathbf{CF}_0^i\mathbf{G}_1 \end{bmatrix}_{(p\times n_y)\times n_u} \tag{3.17}$$

$$\mathbf{\Theta} = \begin{bmatrix} \mathbf{CG}_1 & \dots & 0 \\ \mathbf{CF}_0\mathbf{G}_1 + \mathbf{CG}_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{m-1}\mathbf{CF}_0^i\mathbf{G}_1 & \dots & \mathbf{CG}_1 \\ \sum_{i=0}^{m}\mathbf{CF}_0^i\mathbf{G}_1 & \dots & \mathbf{CF}_0\mathbf{G}_1 + \mathbf{CG}_1 \\ \vdots & \vdots & \vdots \\ \sum_{i=0}^{p-1}\mathbf{CF}_0^i\mathbf{G}_1 & \dots & \sum_{i=0}^{p-m}\mathbf{CF}_0^i\mathbf{G}_1 \end{bmatrix}_{(p\times n_y)\times(m\times n_u)} \tag{3.18}$$

Equation (3.14) related to the prediction can be corrected by adding a feedback term that considers the difference between the process output and the predicted output at sampling instant k as follows

$$\begin{bmatrix} \hat{\mathbf{y}}(k+1|k) \\ \hat{\mathbf{y}}(k+2|k) \\ \vdots \\ \hat{\mathbf{y}}(k+p|k) \end{bmatrix} = \mathbf{\Psi}\mathbf{x}^{\mathrm{L}}(k) + \mathbf{\Gamma}\mathbf{u}(k-1) + \mathbf{\Theta}\begin{bmatrix} \Delta\mathbf{u}(k) \\ \vdots \\ \Delta\mathbf{u}(k+m-1) \end{bmatrix} + \mathbf{N}_2\left[\mathbf{y}^{\mathrm{process}}(k) - \hat{\mathbf{y}}(k)\right] \tag{3.19}$$

where $\hat{\mathbf{y}}(k)$ is calculated from equation (3.5) and $\mathbf{N}_2$ is defined as

$$\mathbf{N}_2 = \begin{bmatrix} \mathbf{I}_{n_y} \\ \vdots \\ \mathbf{I}_{n_y} \end{bmatrix}_{(p \times n_y) \times n_y} \tag{3.20}$$

The standard MPC algorithm cost function is used as follows

$$J = \min_{\Delta \mathbf{U}} \left([\mathbf{Y} - \mathbf{Y}^{\mathrm{sp}}]^{\mathrm{T}} \mathbf{Q} [\mathbf{Y} - \mathbf{Y}^{\mathrm{sp}}] + \Delta \mathbf{U}^{\mathrm{T}} \mathbf{R} \Delta \mathbf{U} \right) \tag{3.21}$$

The term $\Delta \mathbf{U}^{\mathrm{T}} \mathbf{R} \Delta \mathbf{U}$ is introduced in the formula to prevent an excessive movement of the manipulated variables. $\mathbf{Q}$ and $\mathbf{R}$ are positive-definite weighting matrices for the manipulated and controlled variables, respectively, and $\hat{\mathbf{Y}}$, $\mathbf{Y}^{\mathrm{sp}}$ and $\Delta \mathbf{U}$ are defined as

$$\hat{\mathbf{Y}} = [\hat{\mathbf{y}}(k+1), \ldots, \hat{\mathbf{y}}(k+p)]^{\mathrm{T}} \tag{3.22}$$

$$\mathbf{Y}^{\mathrm{sp}} = [\mathbf{y}^{\mathrm{sp}}(k+1), \ldots, \mathbf{y}^{\mathrm{sp}}(k+p)]^{\mathrm{T}} \tag{3.23}$$

$$\Delta \mathbf{U} = [\Delta \mathbf{u}(k), \ldots, \Delta \mathbf{u}(k+m-1)]^{\mathrm{T}} \tag{3.24}$$

The least-squares solution of the minimization problem is given by (Maciejowski, 2002)

$$\Delta \mathbf{u}(k) = \mathbf{K}_{\mathrm{MPC}} \left[\mathbf{Y}^{\mathrm{sp}} - \boldsymbol{\Psi} \mathbf{x}^{\mathrm{L}}(k) - \boldsymbol{\Gamma} \mathbf{u}(k-1) + \mathbf{N}_2 [-\mathbf{y}^{\mathrm{process}}(k) + \hat{\mathbf{y}}(k)] \right] \tag{3.25}$$

where $\mathbf{K}_{\mathrm{MPC}}$ is calculated as follows

$$\mathbf{K}_{\mathrm{MPC}} = \mathbf{N}_1 \left[\left[\boldsymbol{\Theta}^{\mathrm{T}} \mathbf{Q} \boldsymbol{\Theta} + \mathbf{R} \right]^{-1} \boldsymbol{\Theta}^{\mathrm{T}} \mathbf{Q} \right] \tag{3.26}$$

and $\mathbf{N}_1$ is defined as

$$\mathbf{N}_1 = \left[\mathbf{I}_{n_u}, \mathbf{0}_{n_u \times ((m-1) \times n_u)} \right]_{n_u \times (m \times n_u)} \tag{3.27}$$

Then, the value of the manipulated variable at sampling instant k can be obtained from equation (3.25)

$$\mathbf{u}(k) = \mathbf{u}(k-1) + \mathbf{K}_{\mathrm{MPC}}\left[\mathbf{Y}^{\mathrm{sp}} - \boldsymbol{\Psi}\mathbf{x}^{\mathrm{L}}(k) - \boldsymbol{\Gamma}\mathbf{u}(k-1) + \mathbf{N}_2\left[-\mathbf{y}^{\mathrm{process}}(k) + \hat{\mathbf{y}}(k)\right]\right] \tag{3.28}$$

The robust control stability and performance tests require a closed-loop formulation of the system. This formulation is obtained by combining the process equations given by the uncertain state-affine model and the controller equations as shown in the following section.

3.2.3 Closed-Loop state-space model for robustness analysis

The closed-loop formulation is obtained by combining the equations corresponding to the state-affine model equation (3.2), process output equation (3.3), state-space model equation (3.4), controlled variable equation (3.28), and disturbance equation (3.11). For the purpose of robust analysis, the actual feedback term in equation (3.28) is substituted by the difference between the model output from the nonlinear state-affine model obtained from equation (3.13) and the linear predicted output obtained from equation (3.5) as follows

$$\mathbf{y}^{\mathrm{process}}(k) - \hat{\mathbf{y}}(k) = \mathbf{C}^{\mathrm{un}}\mathbf{x}(k) + \mathbf{W}_F\mathbf{d}(k) - \mathbf{C}\mathbf{x}^{\mathrm{L}}(k) \tag{3.29}$$

After substituting the value of $\mathbf{u}(k)$ from equation (3.28) in equations (3.2) and (3.4), the following equations are obtained

$$\begin{aligned}\mathbf{x}^{\mathrm{L}}(k+1) = {} & [\mathbf{G}_1\mathbf{K}_{\mathrm{MPC}}]\,\mathbf{y}^{\mathrm{sp}} + [\mathbf{F}_0 + \mathbf{G}_1\mathbf{K}_{\mathrm{MPC}}(-\boldsymbol{\Psi} + \mathbf{N}_2\mathbf{C})]\,\mathbf{x}^{\mathrm{L}}(k) + \\ & [\mathbf{G}_1\mathbf{K}_{\mathrm{MPC}}(-\mathbf{N}_2\mathbf{C}^{\mathrm{un}})]\,\mathbf{x}(k) + [\mathbf{G}_1(\mathbf{I}_{n_u} - \mathbf{K}_{\mathrm{MPC}}\boldsymbol{\Gamma})]\,\mathbf{u}(k-1) + \\ & [\mathbf{G}_1\mathbf{K}_{\mathrm{MPC}}(-\mathbf{N}_2\mathbf{W}_F)]\,\mathbf{d}(k)\end{aligned} \tag{3.30}$$

$$
\begin{aligned}
\mathbf{x}(k+1) = & \left[\left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\mathbf{K}_{\mathrm{MPC}}\right]\mathbf{y}^{\mathrm{sp}} + \\
& \left[\left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\mathbf{K}_{\mathrm{MPC}}\left(-\boldsymbol{\Psi} + \mathbf{N}_2\mathbf{C}\right)\right]\mathbf{x}^{\mathrm{L}} + \\
& \left[\left(\mathbf{F}_0 + \sum_{i=1}^{n_F-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{F}_{i,j}\right)\right]\mathbf{x}(k) + \\
& \left[\left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\mathbf{K}_{\mathrm{MPC}}\left(-\mathbf{N}_2\mathbf{C}^{\mathrm{un}}\right)\right]\mathbf{x}(k) + \\
& \left[\left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j-1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\left(\mathbf{I}_{n_u} - \mathbf{K}_{\mathrm{MPC}}\boldsymbol{\Gamma}\right)\right]\mathbf{u}(k-1) + \\
& \left[\left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\mathbf{K}_{\mathrm{MPC}}\left(-\mathbf{N}_2\mathbf{W}_F\right)\right]\mathbf{d}(k)
\end{aligned}
\tag{3.31}
$$

Considering the disturbance rejection problem, i.e., $\mathbf{Y}^{\mathrm{sp}} = \mathbf{0}_{(n_y \times p)\times 1}$ equations (3.3), (3.11), (3.28), (3.30) and (3.31) can be written in matrix form as

$$
\left[\begin{array}{c}
\mathbf{x}^{\mathrm{L}}(k+1) \\
\mathbf{x}(k+1) \\
\mathbf{u}(k) \\
\mathbf{d}(k+1) \\ \hline
\mathbf{y}(k)
\end{array}\right]
=
\left[\begin{array}{cccc|c}
\mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14} & \mathbf{B}_{11} \\
\mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} & \mathbf{A}_{24} & \mathbf{B}_{21} \\
\mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} & \mathbf{A}_{34} & \mathbf{B}_{31} \\
\mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{A}_{44} & \mathbf{B}_{41} \\ \hline
\mathbf{C}_{11} & \mathbf{C}_{12} & \mathbf{C}_{13} & \mathbf{C}_{14} & \mathbf{D}_{11}
\end{array}\right]
\left[\begin{array}{c}
\mathbf{x}^{\mathrm{L}}(k) \\
\mathbf{x}(k) \\
\mathbf{u}(k-1) \\
\mathbf{d}(k) \\ \hline
\nu(k)
\end{array}\right]
\tag{3.32}
$$

where

$$[\mathbf{A}_{11}]_{n_x \times n_x} = \mathbf{F}_0 + \mathbf{G}_1\mathbf{K}_{\mathrm{MPC}}\left(-\boldsymbol{\Psi} + \mathbf{N}_2\mathbf{C}\right) \tag{3.33a}$$

$$[\mathbf{A}_{12}]_{n_x \times n_x} = -\mathbf{G}_1\mathbf{K}_{\mathrm{MPC}}\mathbf{N}_2\mathbf{C}^{\mathrm{un}} \tag{3.33b}$$

$$[\mathbf{A}_{13}]_{n_x \times n_u} = \mathbf{G}_1\left(\mathbf{I}_{n_u} - \mathbf{K}_{\mathrm{MPC}}\boldsymbol{\Gamma}\right) \tag{3.33c}$$

$$[\mathbf{A}_{14}]_{n_x \times n_d} = -\mathbf{G}_1\mathbf{K}_{\mathrm{MPC}}\mathbf{N}_2\mathbf{W}_F \tag{3.33d}$$

$$[\mathbf{A}_{21}]_{n_x \times n_x} = \left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\mathbf{K}_{\mathrm{MPC}}\left(-\boldsymbol{\Psi} + \mathbf{N}_2\mathbf{C}\right) \tag{3.34a}$$

$$[\mathbf{A}_{22}]_{n_x\times n_x} = \left(\mathbf{F}_0 + \sum_{i=1}^{n_F-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{F}_{i,j}\right) + \left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\mathbf{K}_{\text{MPC}}\left(-\mathbf{N}_2\mathbf{C}^{\text{un}}\right) \tag{3.34b}$$

$$[\mathbf{A}_{23}]_{n_x\times n_u} = \left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\left(\mathbf{I}_{n_u} - \mathbf{K}_{\text{MPC}}\mathbf{\Gamma}\right) \tag{3.34c}$$

$$[\mathbf{A}_{24}]_{n_x\times n_u} = -\left(\mathbf{G}_1 + \sum_{i=1}^{n_G-1}\sum_{j=1}^{n_u}\delta_{i,j,k}\mathbf{G}_{i+1,j}\right)\mathbf{K}_{\text{MPC}}\mathbf{N}_2\mathbf{W}_F \tag{3.34d}$$

$$[\mathbf{A}_{31}]_{n_u\times n_x} = \mathbf{K}_{\text{MPC}}\left(-\mathbf{\Psi} + \mathbf{N}_2\mathbf{C}\right) \tag{3.35a}$$

$$[\mathbf{A}_{32}]_{n_u\times n_x} = -\mathbf{K}_{\text{MPC}}\mathbf{N}_2\mathbf{C}^{\text{un}} \tag{3.35b}$$

$$[\mathbf{A}_{33}]_{n_u\times n_u} = \mathbf{I}_{n_u} - \mathbf{K}_{\text{MPC}}\mathbf{\Gamma} \tag{3.35c}$$

$$[\mathbf{A}_{33}]_{n_u\times n_d} = -\mathbf{K}_{\text{MPC}}\mathbf{N}_2\mathbf{W}_F \tag{3.35d}$$

$$[\mathbf{A}_{41}]_{n_d\times n_x} = \mathbf{0}_{n_d\times n_x} \tag{3.36a}$$

$$[\mathbf{A}_{42}]_{n_d\times n_x} = \mathbf{0}_{n_d\times n_x} \tag{3.36b}$$

$$[\mathbf{A}_{43}]_{n_d\times n_u} = \mathbf{0}_{n_d\times n_u} \tag{3.36c}$$

$$[\mathbf{A}_{44}]_{n_d\times n_d} = \text{BW} \tag{3.36d}$$

$$[\mathbf{B}_{11}]_{n_x\times n_d} = \mathbf{0}_{n_x\times n_d} \tag{3.37a}$$

$$[\mathbf{B}_{21}]_{n_x\times n_d} = \mathbf{0}_{n_x\times n_d} \tag{3.37b}$$

$$[\mathbf{B}_{31}]_{n_u\times n_d} = \mathbf{0}_{n_u\times n_d} \tag{3.37c}$$

$$[\mathbf{B}_{41}]_{n_d\times n_d} = (1 - \text{BW}) \tag{3.37d}$$

$$[\mathbf{C}_{11}]_{n_y\times n_x} = \mathbf{0}_{n_y\times n_x} \tag{3.38a}$$

$$[\mathbf{C}_{12}]_{n_y\times n_x} = \mathbf{C}_{\text{un}} \tag{3.38b}$$

$$[\mathbf{C}_{13}]_{n_y\times n_u} = \mathbf{0}_{n_y\times n_u} \tag{3.38c}$$

$$[\mathbf{C}_{14}]_{n_y\times n_d} = \mathbf{W}_F \tag{3.38d}$$

$$[\mathbf{D}_{11}]_{n_y\times n_d} = \mathbf{0}_{n_y\times n_d} \tag{3.39}$$

The state-space representation of equation (3.32) can be written in compact form as:

$$\begin{aligned}\begin{bmatrix}\boldsymbol{\eta}(k+1)\\ \mathbf{e}(k)\end{bmatrix} &= \begin{bmatrix}\mathbf{A}(\boldsymbol{\delta}_k) & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}\begin{bmatrix}\boldsymbol{\eta}(k)\\ \nu(k)\end{bmatrix}\\ \boldsymbol{\eta}(0) &= \boldsymbol{\eta}_0\end{aligned} \tag{3.40}$$

where

$$\boldsymbol{\eta}(k+1) = \left[\mathbf{x}^{\mathrm{L}}(k+1), \mathbf{x}(k+1), \mathbf{u}(k), \mathbf{d}(k+1)\right]^{\mathrm{T}} \tag{3.41}$$

$$\boldsymbol{\eta}(k) = \left[\mathbf{x}^{\mathrm{L}}(k), \mathbf{x}(k), \mathbf{u}(k-1), \mathbf{d}(k)\right]^{\mathrm{T}} \tag{3.42}$$

The term $\boldsymbol{\delta}_k$ is a vector of uncertain and time-varying real parameters that has the following elements

$$\begin{aligned}\boldsymbol{\delta}_k = &[\delta_{1,1,k}, \ldots, \delta_{1,n_u,k}, \delta_{2,1,k}, \ldots, \delta_{2,n_u,k}, \ldots,\\ &\delta_{\max(n_F,n_G),1,k}, \ldots, \delta_{\max(n_F,n_G),n_u,k}, \delta_1^{\mathrm{out}}, \ldots, \delta_{n_y}^{\mathrm{out}}\Big]\end{aligned} \tag{3.43}$$

with the following assumptions:

1. Each element of $\boldsymbol{\delta}_k$ is real and is know to be between a lower limit and an upper limit.

2. In the formulation of equation (3.40) the state matrix $\mathbf{A}(\boldsymbol{\delta}_k)$ depends affinely on the parameters as follows:

$$\begin{aligned}\mathbf{A}(\boldsymbol{\delta}_k) = &\mathbf{A}_0 + \mathbf{A}_{1,1}\delta_{1,1,k} + \ldots + \mathbf{A}_{\max(n_F,n_G),n_u}\delta_{\max(n_F,n_G),n_u,k} + \ldots\\ &\mathbf{A}_1^{\mathrm{out}}\delta_1^{\mathrm{out}} + \ldots + \mathbf{A}_{n_y}^{\mathrm{out}}\delta_{n_y}^{\mathrm{out}}\end{aligned} \tag{3.44}$$

   where $\mathbf{A}_0, \mathbf{A}_{1,1} \ldots, \mathbf{A}_{\max(n_F,n_G),n_u}, \mathbf{A}_1^{\mathrm{out}}, \ldots, \mathbf{A}_{n_y}^{\mathrm{out}}$ are known fixed matrices.

The first assumption means that the parameter vector $\boldsymbol{\delta}_k$ is valued in a hyper-rectangle reffered to as the parameter box. In the sequel, $\boldsymbol{\xi}$ denoted the vertices or corners of this parameter box that is defined as follows:

$$\begin{aligned}&\boldsymbol{\xi} = \left(\delta_{1,1,k}, \ldots, \delta_{\max(n_F,n_G),n_u,k}, \delta_1^{\mathrm{out}}, \ldots, \delta_{n_y}^{\mathrm{out}}\right)\\ &\delta_{i,j,k} \in \left[\delta_{i,j,k}^{\mathrm{lb}}, \delta_{i,j,k}^{\mathrm{ub}}\right], \, i = 1, \ldots, \max(n_F, n_G), \, j = 1, \ldots, n_u\\ &\delta_j^{\mathrm{out}} \in \left[\delta_j^{\mathrm{out-lb}}, \delta_j^{\mathrm{out-ub}}\right], \, j = 1, \ldots, n_y\end{aligned} \tag{3.45}$$

This closed-loop state-space representation is used to test robust stability and robust performance by using the tests described in the next section.

3.3 Robust stability and robust performance tests

This section presents the basic definitions and theorems that are required for the LMI based tests. The proofs for the robust stability and robust performance test summarized in this section can be found in (Gao and Budman, 2004) for discrete systems.

Definition 1 *Quadratic Lyapunov Stability (QLS). For systems defined by*

$$\begin{aligned}\boldsymbol{\eta}(k+1) =& \mathbf{A}(\boldsymbol{\delta}_k)\boldsymbol{\eta}(k)\\ \boldsymbol{\eta}(0) =& \boldsymbol{\eta}_0\end{aligned} \tag{3.46}$$

a sufficient condition for asymptotic stability is the existence of a positive-definite quadratic Lyapunov function $V(k) = \boldsymbol{\eta}(k)^{\mathrm{T}}\mathbf{P}\boldsymbol{\eta}$, $V(k) > 0$ *with* $\mathbf{P} > 0$, $\mathbf{P} = \mathbf{P}^{\mathrm{T}}$, *such that*

$$V(k+1) - V(k) < 0 \tag{3.47}$$

for all admissible uncertainties $\boldsymbol{\delta}_k$*and for all initial conditions* $\boldsymbol{\eta}_0$.

Theorem 1 *Let* $\boldsymbol{\delta}_k$ *as defined in equation (3.43) and* $\boldsymbol{\delta}_k \in \Re^{\max(n_F,n_G)+n_u+n_y}$ *be a vector of time-varying uncertain real parameters varying in the hyper-rectangle, and let* $\boldsymbol{\xi}$ *denote the* *set of vertices of this hyper-rectangle. Consider the time varying system in equation (3.32) where* $\mathbf{A}(\boldsymbol{\delta}_k)$ *depends affinely on* $\boldsymbol{\delta}_k$*. The system in equation (3.32) satisfies Quadratic Lyapunov Stability (QLS) (Gao and Budman, 2004) if there exists* $\mathbf{P} > 0$, $\mathbf{P} = \mathbf{P}^{\mathrm{T}}$ *such that*

$$\mathbf{A}(\boldsymbol{\xi})\,\mathbf{P}\mathbf{A}(\boldsymbol{\xi}) - \mathbf{P} < 0, \forall\boldsymbol{\xi} \tag{3.48}$$

In other words, it suffices that $\mathbf{P}$ be positive-definite and satisfy the resulting LMIs at each corner, $\boldsymbol{\xi}$, of the parameter box.

Definition 2 *Quadratic Lyapunov Performance (QLP). The system in equation (3.40) with zero initial state satisfies QLS and*

$$||\mathbf{e}||_{L2} < \gamma\,||\nu||_{L2} \tag{3.49}$$

for all L_2*-bounded input* ν *if there exists a positive-definite quadratic Lyapunov function* $V(k) = \boldsymbol{\eta}(k)^{\mathrm{T}}\mathbf{P}\boldsymbol{\eta}(k)$, $V(k) > 0$ *with* $\mathbf{P} > 0$ *and* $\mathbf{P} = \mathbf{P}^{\mathrm{T}}$ *such that*

$$V(k+1) - V(k) + [\mathbf{e}(k)]^{\mathrm{T}}\,\mathbf{e}(k) - \gamma^2\,[\nu(k)]^{\mathrm{T}}\,\nu(k) < 0 \tag{3.50}$$

for all admissible uncertainties $\boldsymbol{\delta}_k$ *and for zero initial conditions* $\boldsymbol{\eta}_0$*.*

Theorem 2 *Let* $\boldsymbol{\delta}_k$ *as defined in equation (3.43) and* $\boldsymbol{\delta}_k \in \Re^{\max(n_F,n_G)+n_u+n_y}$ *be a vector of time-varying uncertain real parameters varying in the hyper-rectangle, and let* $\boldsymbol{\xi}$ *denote the set of vertices of this hyper-rectangle. Consider the time varying system in equation (3.32) where* $\mathbf{A}(\boldsymbol{\delta}_k)$ *depends affinely on* $\boldsymbol{\delta}_k$*. The time-varying system in equation (3.32) satisfies quadratic Lyapunov performance (QLP) (Gao and Budman, 2004) if there exists* $\mathbf{P} > 0$, $\mathbf{P} = \mathbf{P}^{\mathrm{T}}$ *such that*

$$\begin{bmatrix} [\mathbf{A}(\boldsymbol{\xi})]^{\mathrm{T}}\,\mathbf{P}\mathbf{A}(\boldsymbol{\xi}) - \mathbf{P} & [\mathbf{A}(\boldsymbol{\xi})]^{\mathrm{T}}\,\mathbf{P}\mathbf{B} & \mathbf{C}^{\mathrm{T}} \\ \mathbf{B}^{\mathrm{T}}\mathbf{P}\mathbf{A}(\boldsymbol{\xi}) & \mathbf{B}^{\mathrm{T}}\mathbf{P}\mathbf{B} - \gamma^2\mathbf{I} & \mathbf{D}^{\mathrm{T}} \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < 0, \forall\boldsymbol{\xi} \tag{3.51}$$

The inequality in equation (3.51) can be solved to minimize the performance index γ for all possible models contained in $\boldsymbol{\xi}$ and for all disturbances that have a specific magnitude, e.g., $|\nu| < 1$.

$$\gamma_{\mathrm{P}} = \begin{cases} \min \gamma \\ \mathrm{wrt}\,\mathrm{P} \\ \quad \mathrm{st} \begin{cases} \mathbf{P} > 0,\ \mathbf{P} = \mathbf{P}^{\mathrm{T}} \\ \begin{bmatrix} [\mathbf{A}(\boldsymbol{\xi})]^{\mathrm{T}} \mathbf{P}\mathbf{A}(\boldsymbol{\xi}) - \mathbf{P} & [\mathbf{A}(\boldsymbol{\xi})]^{\mathrm{T}} \mathbf{P}\mathbf{B} & \mathbf{C}^{\mathrm{T}} \\ \mathbf{B}^{\mathrm{T}}\mathbf{P}\mathbf{A}(\boldsymbol{\xi}) & \mathbf{B}^{\mathrm{T}}\mathbf{P}\mathbf{B} - \gamma^2\mathbf{I} & \mathbf{D}^{\mathrm{T}} \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < 0,\ \forall \boldsymbol{\xi} \end{cases} \end{cases} \tag{3.52}$$

The minimization problem in equation (3.52) can be solved with the interface YALMIP (Löfberg, 2004) and a semi definite programming (SDP) solver such as SDPT3 (Toh et al., 1999).

3.4 Robust gain-scheduled MPC design

For open-loop stable plants, the closed-loop system performance and the uncertainty bounds calculation depend on the MPC design parameters, p, m, $\mathbf{Q}$ and $\mathbf{R}$. In principle all the design parameters, i.e., p, m, $\mathbf{Q}$ and $\mathbf{R}$ could be changed to achieve optimality. However, the effect of p and m will require the solution of a computationally demanding mixed integer nonlinear programming optimization. For simplicity, in this work the manipulated variables weight matrix $\mathbf{R}$ is the only parameter that is changed to achieve optimality, whereas the other parameters are assumed constant. In this work the choice of p and m was done according to practical guidelines reported in the literature (Agachi et al., 2007), i.e., m between 1 to 4 and p equal to approximately 3 times the dominant time constant of the input-output responses. Also, for each input variable $\mathbf{R}$ is maintained constant along the control horizon m, for a multiple input case $\mathbf{R}$ is as follows:

$$\mathbf{R} = \mathrm{diag}\left[\mathbf{R}_1, \ldots, \mathbf{R}_m\right] \tag{3.53}$$

where

$$\mathbf{R}_i = \operatorname{diag}\left[r_1, \ldots, r_{n_u}\right], i = 1, \ldots, m \tag{3.54}$$

After the MPC design parameters have been specified, the first step in the design of the robust controller is to obtain the uncertainty bounds, i.e., it is necessary to know the values of $\boldsymbol{\delta}_k$. The evaluation of $\boldsymbol{\delta}^{\text{out}}$ was discussed in the previous section. With respect to the evaluation of $\delta_{1,1,k}, \ldots, \delta_{\max(n_F,n_G),n_u,k}$, a simple option is to assume that these bounds correspond to the saturation limits of the manipulated variables. It has been shown previously (Gao and Budman, 2004) that this assumption results in highly conservative controllers since it requires considering unnecessarily large changes in the manipulated variables. Provided that the control loops are properly designed, the system will operate for the majority of the time away from the saturation limits. Another option is to calculate the saturation limits from simulations for particular disturbances and a specific set of tuning parameters. However, this strategy is very time consuming especially since the simulations have to be conducted for each set of tuning parameters assumed during the optimization of the controller. A much better option is to calculate the bounds analytically based on the uncertain model proposed in section 3.2.1. The key idea is based on the recognition, from equation (3.28), that the control action is equal to the product of the controller gain multiplied by a feedback correction term that is composed of the disturbances entering the system plus the model error occurring between the output of the linear model used for MPC control and the output from the nonlinear process.

According to equation (3.28) and assuming for the disturbance rejection problem that the set point $\mathbf{Y}^{\text{sp}} = \mathbf{0}$, the effect of the feedback on the manipulated variables is as follows

$$\mathbf{W}_{\text{feedback}} = \mathbf{K}_{\text{MPC}}\mathbf{N}_2\left[-\mathbf{C}^{\text{un}}\mathbf{x}(k) - \mathbf{W}_F\mathbf{d}(k) + \mathbf{C}\mathbf{x}^{\text{L}}(k)\right] \tag{3.55}$$

The key idea is to find a bound for this term by using the same RMS calculation proposed in section 3.3 for calculating the worst output error. The only difference with respect to the calculation in section 3.3 is that the output obtained from equation (3.32) used for calculating the RMS of the output is substituted by an output equation that is equal to the effect of the feedback on the manipulated variable 5

$$\mathbf{y}_n(k) = \mathbf{W}_{\text{feedback}}^{\text{worst}} \tag{3.56}$$

The effect of the feedback on the manipulated variable can be written as a vector where each element is the feedback effect associated with the i input as follows

$$\mathbf{W}_{\text{feedback}} = \left[\mathbf{W}_1^{\text{wa}}, \ldots, \mathbf{W}_{n_u}^{\text{wa}}\right]^{\text{T}} \tag{3.57}$$

each element $\mathbf{W}_i^{\text{wa}}$, $i = 1, \ldots, n_u$ is defined as

$$\mathbf{W}_i^{\text{wa}} = \boldsymbol{\alpha}_i \mathbf{K}_{\text{MPC}} \mathbf{N}_2 \left[-\mathbf{C}^{\text{un}}\mathbf{x}(k) - \mathbf{W}_F \mathbf{d}(k) + \mathbf{C}\mathbf{x}^{\text{L}}(x)\right] \tag{3.58}$$

the vector $\boldsymbol{\alpha}_i$ with dimensions $1 \times n_u$ is defined as

$$\begin{aligned} \boldsymbol{\alpha}_i &= [\alpha_{1,1}, \ldots, \alpha_{1,n_u}] \\ \text{with} &\begin{cases} \alpha_{1,j} = 0, \; j \neq i \\ \alpha_{1,j} = 1, \; j = i \end{cases} \end{aligned} \tag{3.59}$$

Considering that $\mathbf{Y}^{\text{sp}} = \mathbf{0}_{(n_y \times p)\times 1}$, equations (3.11), (3.28), (3.30), (3.31) and (3.58) can be formulated as follows

$$\left[\begin{array}{c} \mathbf{x}^{\text{L}}(k+1) \\ \mathbf{x}(k+1) \\ \mathbf{u}(k) \\ \mathbf{d}(k+1) \\ \hline \mathbf{W}_j^{\text{wa}}(k) \end{array}\right] = \left[\begin{array}{cccc|c} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14} & \mathbf{B}_{11} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} & \mathbf{A}_{24} & \mathbf{B}_{21} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} & \mathbf{A}_{34} & \mathbf{B}_{31} \\ \mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{A}_{44} & \mathbf{B}_{41} \\ \hline \mathbf{C}_{11} & \mathbf{C}_{12} & \mathbf{C}_{13} & \mathbf{C}_{14} & \mathbf{D}_{11} \end{array}\right] \left[\begin{array}{c} \mathbf{x}^{\text{L}}(k) \\ \mathbf{x}(k) \\ \mathbf{u}(k-1) \\ \mathbf{d}(k) \\ \hline \nu(k) \end{array}\right] \tag{3.60}$$

where the matrices **A**, **B** and **D** have a structure similar to equations (3.33) to (3.36), (3.37) and (3.39), respectively. The **C** matrix has a structure that is defined as

$$[\mathbf{C}_{11}]_{1\times n_y} = \boldsymbol{\alpha}_i \mathbf{K}_{\mathrm{MPC}} \mathbf{N}_2 \mathbf{C} \tag{3.61a}$$

$$[\mathbf{C}_{12}]_{1\times n_y} = -\boldsymbol{\alpha}_i \mathbf{K}_{\mathrm{MPC}} \mathbf{N}_2 \mathbf{C}^{\mathrm{un}} \tag{3.61b}$$

$$[\mathbf{C}_{13}]_{1\times n_u} = \mathbf{0}_{1\times n_u} \tag{3.61c}$$

$$[\mathbf{C}_{14}]_{1\times n_d} = -\boldsymbol{\alpha}_i \mathbf{K}_{\mathrm{MPC}} \mathbf{N}_2 \mathbf{W}_F \tag{3.61d}$$

For every element of the vector $\mathbf{W}_{\mathrm{feedback}}$, there will be a closed-loop system representation where the only elements that change are the terms $\mathbf{C}_{11}$, $\mathbf{C}_{12}$ and $\mathbf{C}_{14}$ of equation (3.61).

The minimization problem of equation (3.52) applied to the closed-loop system of equation (3.60) provides a bound over the standard deviation of $\mathbf{W}^{\mathrm{worst}}_{\mathrm{feedback}}$ (Ricardez et al., 2008). This bound can be used to calculate the worst $\Delta\mathbf{u}\,(k)$ and consequently the worst $\mathbf{u}\,(k)$ which are calculated according to the following formulas based on equations (3.25) and (3.28).

$$\Delta\mathbf{u}^{\mathrm{worst}}\,(k) = \mathbf{K}_{\mathrm{MPC}}\left[-\boldsymbol{\Psi}\mathbf{x}^{\mathrm{L}}\,(k) - \boldsymbol{\Gamma}\mathbf{u}^{\mathrm{worst}}\,(k-1)\right] + \mathbf{W}^{\mathrm{worst}}_{\mathrm{feedback}} \tag{3.62}$$

$$\mathbf{u}^{\mathrm{worst}}\,(k) = -\mathbf{K}_{\mathrm{MPC}}\boldsymbol{\Psi}\mathbf{x}^{\mathrm{L}}\,(k) + [\mathbf{I}_{n_u} - \mathbf{K}_{\mathrm{MPC}}\boldsymbol{\Gamma}]\,\mathbf{u}^{\mathrm{worst}}\,(k-1) + \mathbf{W}^{\mathrm{worst}}_{\mathrm{feedback}} \tag{3.63}$$

The value of $\mathbf{u}^{\mathrm{worst}}\,(k)$ from equation (3.63) can be substituted in equation (3.4) to obtain

$$\mathbf{x}^{\mathrm{L}}\,(k) = [\mathbf{F}_0 - \mathbf{G}_1\mathbf{K}_{\mathrm{MPC}}\boldsymbol{\Psi}]\,\mathbf{x}^{\mathrm{L}}\,(k) + \mathbf{G}_1\,[\mathbf{I}_{n_u} - \mathbf{K}_{\mathrm{MPC}}\boldsymbol{\Gamma}]\,\mathbf{u}^{\mathrm{worst}}\,(k-1) + \mathbf{G}_1\mathbf{W}^{\mathrm{worst}}_{\mathrm{feedback}} \tag{3.64}$$

Then, equations (3.63) and (3.64) can be put into state-space form, where the state-update equation is as follows

$$\begin{bmatrix} \mathbf{x}^{\mathrm{L}}\,(k+1) \\ \mathbf{u}^{\mathrm{worst}}\,(k) \end{bmatrix} = \begin{bmatrix} \mathbf{A}^{\mathrm{wa}}_{11} & \mathbf{A}^{\mathrm{wa}}_{12} \\ \mathbf{A}^{\mathrm{wa}}_{21} & \mathbf{A}^{\mathrm{wa}}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\mathrm{L}}\,(k) \\ \mathbf{u}^{\mathrm{worst}}\,(k-1) \end{bmatrix} + \begin{bmatrix} \mathbf{B}^{\mathrm{wa}}_{11} \\ \mathbf{B}^{\mathrm{wa}}_{21} \end{bmatrix} \tag{3.65}$$

Since it is desired to calculate a bound for each element of the $\mathbf{u}^{\text{worst}}(k)$ vector, i.e., the value of $u_i^{\text{worst}}(k)$, $i = 1, \ldots, n_u$, the corresponding state-space output equations is used as follows

$$u_i^{\text{worst}} = \begin{bmatrix} \mathbf{C}_{11}^{\text{wa}} & \mathbf{C}_{12}^{\text{wa}} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{\text{L}}(k) \\ \mathbf{u}^{\text{worst}}(k-1) \end{bmatrix} + \mathbf{D}_{11}^{\text{wa}} \tag{3.66}$$

where the matrices $\mathbf{A}^{\text{wa}}$, $\mathbf{B}^{\text{wa}}$, $\mathbf{C}^{\text{wa}}$ and $\mathbf{D}^{\text{wa}}$ of equations (3.65) and (3.66) are defined as

$$\mathbf{A}_{11}^{\text{wa}} = \mathbf{F}_0 - \mathbf{G}_1\mathbf{K}_{\text{MPC}}\boldsymbol{\Psi} \tag{3.67a}$$

$$\mathbf{A}_{12}^{\text{wa}} = \mathbf{G}_1\left[\mathbf{I}_{n_u} - \mathbf{K}_{\text{MPC}}\boldsymbol{\Gamma}\right] \tag{3.67b}$$

$$\mathbf{A}_{21}^{\text{wa}} = -\mathbf{K}_{\text{MPC}}\boldsymbol{\Psi} \tag{3.67c}$$

$$\mathbf{A}_{22}^{\text{wa}} = \mathbf{I}_{n_u} - \mathbf{K}_{\text{MPC}}\boldsymbol{\Gamma} \tag{3.67d}$$

$$\mathbf{B}_{11}^{\text{wa}} = \mathbf{G}_1\mathbf{W}_{\text{feedback}}^{\text{worst}} \tag{3.68a}$$

$$\mathbf{B}_{21}^{\text{wa}} = \mathbf{W}_{\text{feedback}}^{\text{worst}} \tag{3.68b}$$

$$\mathbf{C}_{11}^{\text{wa}} = \boldsymbol{\alpha}_i\left[-\mathbf{K}_{\text{MPC}}\boldsymbol{\Psi}\right] \tag{3.69a}$$

$$\mathbf{C}_{12}^{\text{wa}} = \boldsymbol{\alpha}_i\left[\mathbf{I}_{n_u} - \mathbf{K}_{\text{MPC}}\boldsymbol{\Gamma}\right] \tag{3.69b}$$

$$\mathbf{D}_{11}^{\text{wa}} = \boldsymbol{\alpha}_i\mathbf{W}_{\text{feedback}}^{\text{worst}} \tag{3.70}$$

Since the closed-loop system given by equation (3.65) is linear, the calculation of the bound on the manipulated variable can be done either in the time domain or in the frequency domain. It was decided to perform the calculation in the frequency domain to reduce conservatism by allowing for the possibility of limiting the analysis to a finite frequency bandwidth corresponding to the bandwidth of the disturbances entering the system; i.e., it is considered that $\mathbf{d}^{\text{lb}} < \mathbf{d} < \mathbf{d}^{\text{ub}}$. Thus, the bound of $\mathbf{u}^{\text{worst}}$ can be obtained from (Morari and Zafiriou, 1989)

$$u_{i\ \max} = \sup_{\omega^{\text{lb}} < \omega < \omega^{\text{ub}}} \left|\mathbf{H}_i(\omega)\right|, i = 1, \ldots, n_u \tag{3.71}$$

where $\mathbf{H}_i$ is the transfer function for the state-space system defined by equations (3.65) and (3.66), whereas ω^{lb} and ω^{ub} are the lower and upper frequencies, respectively, of the disturbance $\mathbf{d}$ that enters the system. Since $\mathbf{u}^{\text{worst}}$ was calculated based on the bound over $\mathbf{W}^{\text{worst}}_{\text{feedback}}$ that corresponds to a RMS bound, $\mathbf{u}^{\text{worst}}$ represents a bound over the standard deviation of the vector $\mathbf{u}^{\text{worst}}$. Therefore, if one assumes that the disturbances are normally distributed, it can be expected that uncertainty bounds corresponding to all possible disturbances entering the system can be calculated with a 93% confidence level by considering $2\mathbf{u}^{\text{worst}}$.

The procedure to calculate the uncertainty bounds can be summarized as per Procedure 1.

Procedure 1: Calculation of the uncertainty bound $\delta_{1,j,k}$, $j = 1, \ldots, n_u$

1. Obtain the state-affine matrices $\mathbf{F}$ and $\mathbf{G}$, normalized disturbance magnitude $\mathbf{W}_F$, output uncertainty $\boldsymbol{\delta}^{\text{out}}$, and disturbance frequency limits ω^{lb} and ω^{ub}.

2. For a specific set of values of prediction horizon p, control horizon m and controlled variables weight matrix $\mathbf{Q}$:

   2.1. For the current value of the weight matrix $\mathbf{R}$:

      2.1.1. Calculate the controller $\mathbf{K}_{\text{MPC}}$ according to equation (3.26).

   2.2. Provide an initial estimate for every $\delta_{1,j,k}$, $j = 1, \ldots, n_u$

   2.3. Obtain the matrices specified in equations (3.33) to (3.37), (3.39) and (3.61).

      2.3.1. Build the closed-loop system given by equation (3.60)

   2.4. Calculate γ_{P} according to the minimization problem of equation (3.52).

      2.4.1. Update the value of $\mathbf{W}^{\text{worst}}_{\text{feedback}}$ according to the following relationship $\mathbf{W}^{\text{worst}}_{\text{feedback}} = \gamma_{\text{P}}$.

   2.5. For every $\delta_{1,j,k}$, $j = 1, \ldots, n_u$:

      2.5.1. Obtain $\boldsymbol{\alpha}_j$ according to equation (3.59).

      2.5.2. Obtain the matrices specified in equations (3.67) to (3.70).

2.5.3. Build the closed-loop system given by equations (3.65) and (3.66) and obtain its transfer matrix $\mathbf{H}_j$.

2.5.4. Obtain $u_{j\ \max}$ from equation (3.71).

2.5.5. If $\exists\, \delta_{1,j,k},\, :\, |\delta_{1,j,k} - 2u_{j\ \max}| > \epsilon,\ j = 1, \ldots, n_u$.

2.5.5.1. Calculate a new estimate for the uncertainty bound as $\delta_{1,j,k} = 2u_{j\ \max}$.

2.5.5.2. Go to step 2.3 and repeat the process.

2.5.6. If $|\delta_{1,j,k} - 2u_{j\max}| \leq \epsilon,\ j = 1, \ldots, n_u$

2.5.6.1. Accept the uncertainty bound vector $\boldsymbol{\delta}_k$.

After the uncertainty bounds have been obtained according to Procedure 1, the value of the output variability γ for the closed-loop system of equation (3.32) can be calculated. This value is a measure of the effect of the disturbance on the output RMS of the system, i.e., the effect that the disturbance $\nu\,(k)$ has over $\mathbf{y}\,(k)$, i.e.,

$$\gamma = \frac{||\mathbf{e}||_2}{||\boldsymbol{\nu}||_2} = \frac{||\mathbf{y}||_2}{||\boldsymbol{\nu}||_2} \tag{3.72}$$

Thus, a small value of γ indicates that $\mathbf{y}\,(k)$ is barely affected by $\nu\,(k)$, whereas a larger value of γ indicates that $\mathbf{y}\,(k)$ is greatly affected by $\nu\,(k)$. Since the purpose of the controller is to keep the effect of $\nu\,(k)$ over $\mathbf{y}\,(k)$ to a minimum, then the controller must be tuned by varying the weight matrix $\mathbf{R}$ so that γ is reduced to its minimum.

Accordingly, a procedure is proposed to design a robust MPC controller as per Procedure 2.

Procedure 2: Calculation of a robust MPC for minimization of the output RMS.

1. Obtain the state-affine matrices $\mathbf{F}$ and $\mathbf{G}$, normalized disturbance magnitude $\mathbf{W}_F$, output uncertainty $\boldsymbol{\delta}^{\text{out}}$, and disturbance frequency limits ω^{lb} and ω^{ub}.

2. For a specific set of values of prediction horizon p, control horizon m and controlled variables weight matrix $\mathbf{Q}$:

3. Perform a grid search to provide an initial estimate for the minimization of the output RMS according to the following steps:

   3.1. Select a lower and upper bound for the manipulated variables weight matrix, i.e., $\mathbf{R} \in \left[\mathbf{R}^{\text{lb}}, \mathbf{R}^{\text{ub}}\right]$ where $\mathbf{R}^{\text{lb}} = \text{diag}\left[\mathbf{R}_1^{\text{lb}}, \ldots, \mathbf{R}_m^{\text{lb}}\right]$, $\mathbf{R}_j^{\text{lb}} = \text{diag}\left[r_1^{\text{lb}}, \ldots, r_{n_u}^{\text{lb}}\right]$, $i = 1, \ldots, m$ and $\mathbf{R}^{\text{ub}} = \text{diag}\left[\mathbf{R}_1^{\text{ub}}, \ldots, \mathbf{R}_m^{\text{ub}}\right]$, $\mathbf{R}_j^{\text{ub}} = \text{diag}\left[r_1^{\text{ub}}, \ldots, r_{n_u}^{\text{ub}}\right]$, $i = 1, \ldots, m$.

   3.2. Subdivide the interval $\left[\mathbf{R}^{\text{lb}}, \mathbf{R}^{\text{ub}}\right]$ in q_i subintervals.

   3.3. For every q_i subinterval create a weight matrix $\mathbf{R}_{q_i}$ where $\mathbf{R}_{q_i} = \text{diag}\left[\ \mathbf{R}_{1,q_i}, \ldots, \mathbf{R}_{m,q_i}\ \right]$ and $\mathbf{R}_{j,q_i} = \text{diag}\left[r_{1,q_i}, \ldots, r_{n_u,q_i}\right]$, $j = 1, \ldots, m$.

   3.4. For every $\mathbf{R}_i$, $i = 1, \ldots, q_i$:

      3.4.1. Calculate the controller $\mathbf{K}_{\text{MPC}}$ according to equation (3.26).

      3.4.2. Obtain the uncertainty bounds vector $\boldsymbol{\delta}_k$ as per Procedure 1.

      3.4.3. Obtain the matrices specified in equations (3.33) to (3.39).

         3.4.3.1. Build the closed-loop system given by equation (3.32).

      3.4.4. Calculate γ_{P} according to the minimization problem of equation (3.52).

      3.4.5. If a feasible solution exists for the minimization problem of equation (3.52), accept the optimized performance index γ_{P}.

4. The weight matrix $\mathbf{R}_i$ that achieved the minimum γ_{P} is used as an initial estimate for the following minimization problem

$$\gamma^{\text{optimal}} = \begin{cases} \min \gamma_{\text{P}} \\ \text{wrt}\, \mathbf{R} \end{cases}$$

where

$$\gamma_{\text{P}} = \begin{cases} \min \gamma \\ \text{wrt}\, \text{P} \\ \quad \text{st} \begin{cases} \mathbf{P} > 0,\ \mathbf{P} = \mathbf{P}^{\text{T}} \\ \begin{bmatrix} [\mathbf{A}(\boldsymbol{\xi})]^{\text{T}} \mathbf{PA}(\boldsymbol{\xi}) - \mathbf{P} & [\mathbf{A}(\boldsymbol{\xi})]^{\text{T}} \mathbf{PB} & \mathbf{C}^{\text{T}} \\ \mathbf{B}^{\text{T}}\mathbf{PA}(\boldsymbol{\xi}) & \mathbf{B}^{\text{T}}\mathbf{PB} - \gamma^2 \mathbf{I} & \mathbf{D}^{\text{T}} \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < 0,\ \forall \boldsymbol{\xi} \end{cases} \end{cases} \tag{3.73}$$

5. Solve the optimization problem of equation (3.73). To calculate γ_{P}, follow steps 3.4.1 to 3.4.5.

6. The minimal analytical γ will be referred heretofore as γ^{optimal}, and its corresponding $\mathbf{R}$ matrix as $\mathbf{R}^{\text{optimal}}$.

Although the inner optimization in equation (3.73) is convex, i.e., the function γ_{P} is convex with respect to $\mathbf{P}$, the outer optimization problem that searches for the optimal performance index, γ^{optimal}, is non-convex with respect to the controller weight matrix $\mathbf{R}$. Thus, the optimization procedure outlined above may find a local optimum instead of a global optimum. To address this problem, the outer optimization was conducted for a large set of initial values. The procedure for the design of a robust MPC controller discussed above can be used to design a gain-scheduled MPC strategy that is made of a family of linear MPC controllers. To design such strategy the designer must select a priori n_R operating regions around which the system is expected to operate for long periods of time. Typically, each operating condition may be defined based on the average value of a disturbance entering the process, e.g., a production rate or a feed concentration value. Then, a nonlinear state-affine model of the type given by equations (3.2) and (3.3) can be identified around each of the n_R selected operating conditions. Finally, the procedure discussed above is used to design n_R MPC controllers for each of the predefined n_R regions where each controller is activated in their corresponding region. Since the disturbances that were used to determine the operating regions may not be measurable online, the scheduling of the controller can be done based on the manipulated variables $\mathbf{u}$ calculated by the controller in response to the disturbances entering the process.

3.5 Case study results

To illustrate the design technique a two-input-two-output bioreactor example was taken from the literature (Hoo and Kantor, 1986). This example has been used before as a benchmark problem for testing different nonlinear strategies (Imsland et al., 2003). The mechanistic model is described by the following set of equations

$$\frac{\mathrm{d}c_1}{\mathrm{d}t} = \left[\frac{\mu_1 S}{K+S} - D\right] c_1 \tag{3.74}$$

$$\frac{\mathrm{d}c_2}{\mathrm{d}t} = \left[\frac{\mu_2 S}{K+S}\,\frac{K_I}{K_I + c_3} - D\right] c_2 \tag{3.75}$$

$$\frac{\mathrm{d}c_3}{\mathrm{d}t} = P_r c_1 c_3 + D\left(c_{3,f} - c_3\right) \tag{3.76}$$

$$S = S_{\text{in}} - \frac{c_1}{Y_{c,1}} - \frac{c_2}{Y_{c,2}} \tag{3.77}$$

The value of the parameters can be found in Table 3.2. The model describes the dynamics of two cell strains that differ in their sensitivity to an external growth inhibiting agent, where c_1 [g/L] is the concentration of the inhibitor resistant cells; c_2 [g/L] is the concentration of the inhibitor sensitive cells; c_3 [g/L] is the concentration of the inhibitor in the medium; and S [g/L] is the concentration of the rate-limiting substrate.

From input-output information, collected from simulations of the mechanistic model given above, a state-affine model representation as given by equations (3.2) and (3.3) with sampling time of 5 h was obtained to apply the proposed methodology. Since the time constant of this process was found to be of the order of 1000 h or larger, a sampling time of 5 h was considered to be suitable as the sampling interval to be used for control. The model was identified using a least-squares regression where the system was excited by a multilevel pseudorandom binary sequence (PRBS) that is commonly used as an excitation signal for

Table 3.2: List of parameter values

Parameter	Definition	Value
μ_1	Maximum specific growth rate of species 1	0.4 hr^{-1}
μ_2	Maximum specific growth rate of species 2	0.5 hr^{-1}
K	Substrate saturation constant	0.05 (g/L)
K_I	Inhibitor saturation constant	0.02 (g/L)
$Y_{c,1}$	Yield coefficient for species 1	0.2
Y_{c_2}	Yield coefficient for species 2	0.15
P_r	Rate constant for the deactivation of the inhibitor	0.5 (g/L h)

the identification of nonlinear models (Nowak and Van Veen, 1994). The concentrations c_1 and c_2 were selected as controlled variables, whereas the dilution rate D and feed rate Dc_{3f} were selected as manipulated variables. The controlled and manipulated variables were normalized according to the following formulas

$$x_1 = \frac{(c_1 - 0.16)}{0.05} \tag{3.78}$$

$$x_2 = \frac{(c_2 - 0.06)}{0.05} \tag{3.79}$$

$$u_1 = \frac{D - D^{\text{ss}}}{8.2559 \times 10^{-3}} \tag{3.80}$$

$$u_2 = \frac{Dc_{3f} - (Dc_{3f})^{\text{ss}}}{4.1279 \times 10^{-5}} \tag{3.81}$$

where D^{ss} and $(Dc_{3f})^{\text{ss}}$ are the steady-state values corresponding to the operating condition around which a linear model is identified for the purpose of designing a local MPC controller.

In the simulation studies, a gain-scheduled MPC is compared to a linear MPC (LMPC). The value of the substrate's feed concentration S_{in} will be considered as the main disturbance affecting the process. Therefore, the objective of both control strategies consists in rejecting the disturbance and keeping the control variables within their predefined set-point ($c_1^{\text{sp}} = 0.16$ g/L and $c_2^{\text{sp}} = 0.06$ g/L).

A scenario where GSMPC is expected to perform better than LMPC is when the process is operated for long periods of time around operating conditions corresponding to different steady-state values of the feed concentration S_{in}. This is based on the assumption that the disturbances in S_{in} consist of high frequency oscillations superimposed on very infrequent changes of the average value of this variable. Gain-Scheduled algorithms are especially suited for these situations since the controller is designed based on a set of "local" models identified for each one of these regions. This also corresponds to the typical mode of operation of gain-scheduled control in the chemical industry where processes are typically operated around different specific steady-states for long periods of time. These steady-states may correspond to different product grades, types of feed, production rates etc. Then, a model is identified around each steady-state and a corresponding MPC controller is designed based on that model.

To test the performance of both controllers, two different case studies were considered. Case study A considers that the system is operated around three steady-state values $S_{\text{in}}^{\text{ss}} = 2.0$ g/L, $S_{\text{in}}^{\text{ss}} = 2.5$ g/L and $S_{\text{in}}^{\text{ss}} = 3.0$ g/L, where as case study B considers a larger windows of operation that include steady-states corresponding to $S_{\text{in}}^{\text{ss}} = 2.0$ g/L $S_{\text{in}}^{\text{ss}} = 3.0$ g/L and $S_{\text{in}}^{\text{ss}} = 4.0$ g/L. It should be noticed that the feed concentration is not measured, and the manipulated variables are the only ones used for scheduling. For every steady-state value, a state-affine model with the following structure was identified

$$\begin{aligned}\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = &\left[\mathbf{F}_0 + \mathbf{F}_{1,1}\delta_{1,1,k} + \mathbf{F}_{1,2}\delta_{1,2,k}\right] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \\ &\left[\mathbf{G}_1 + \mathbf{G}_{2,1}\delta_{1,1,k} + \mathbf{G}_{2,2}\delta_{1,2,k}\right] \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}\end{aligned} \tag{3.82}$$

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \left[\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} \delta_1^{\text{out}} & 0 \\ 0 & \delta_2^{\text{out}} \end{bmatrix}\right] \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \mathbf{W}_F \mathbf{d}(k) \tag{3.83}$$

where $\delta_1^{\text{out}} \in [-0.1, 0.1]$ and $\delta_2^{\text{out}} \in [-0.1, 0.1]$.

The matrices of the models identified for the different operating conditions specified above can be found in Appendix A. The designed GSMPC is composed of three MPC controllers, each designed for a model identified around a particular value of $S_{\text{in}}^{\text{ss}}$ following procedures 1 and 2 outlined in the previous section. The LMPC to be compared with the GSMPC, is designed based on the minimization of the following average:

$$\gamma_{\text{LMPC}}^{\text{optimal}} = \frac{\sum_{i=1}^{n_R} \gamma_{\text{LMPC}}^{\text{Region}\,i}}{n_R} \tag{3.84}$$

where each $\gamma_{\text{LMPC}}^{\text{Region}\,i}$ is calculated according to the procedure described in the previous section by using, to represent the process, the state-affine model and uncertainty bounds identified for the j region. The LMPC controller, used to minimize the objective function in equation (3.84), uses as internal model the state-space model corresponding to the middle value of the inlet concentration, i.e., the model identified for region two, and it uses the same input weight matrix for the three regions. This corresponds to the standard approach for implementation of linear MPC controllers where the controller is designed based on a model identified around nominal operating conditions, and then it is tuned for robustness to account for situations where the system is operated away from these nominal conditions.

The parameters of the GSMPC and LMPC controllers used in all the case studies are $p = 100$, $m = 4$. Since the output variables are assigned equal importance $\mathbf{Q}$ was selected as the identity, i.e.,

$$\mathbf{Q} = \text{diag}\left[\mathbf{Q}_1, \ldots, \mathbf{Q}_p\right] \tag{3.85}$$

$$\mathbf{Q}_i = \text{diag}\left[1, 1\right], i = 1, \ldots, p \tag{3.86}$$

A preliminary test conducted to assess whether using a GSMPC can improve the performance compared to a LMPC consists of comparing the GSMPC and LMPC designed based on the nominal models, i.e., when the uncertainty is zero

$$\begin{aligned} &\boldsymbol{\xi} = \left(\delta_{1,1,k}, \ldots, \delta_{\max(n_F,n_G),n_u,k}, \delta_1^{\text{out}}, \ldots, \delta_{n_y}^{\text{out}}\right) \\ &\delta_{i,j,k} = 0,\ i = 1, \ldots, \max(n_F, n_G),\ j = 1, \ldots, n_u \\ &\delta_j^{\text{out}} = 0,\ j = 1, \ldots, n_y \end{aligned} \tag{3.87}$$

Since the main source of the uncertainty are related to the deviations in the manipulated variables $\mathbf{u}$, and since these deviations are linearly related to $\mathbf{W}^{\text{worst}}_{\text{feedback}}$, the nominal case corresponds to a situation were $\mathbf{W}^{\text{worst}}_{\text{feedback}} = 0$ or equivalently, to the hypothetical case with disturbances of zero magnitude. Thus, when the disturbances are zero, the uncertainty bounds associated to the manipulated variables are correspondingly equal to zero. In addition, for the nominal case, the model error referred to as $\boldsymbol{\delta}^{\text{out}}$ is also assumed to be zero. The optimal values of γ calculated following the procedure outlined in the previous section are shown in Table 3.3 and Table 3.4 for case studies A and B, respectively. It is clear from these tables that GSMPC is superior to LMPC as indicated by the smaller γ values obtained with GSMPC. It is also clear that the most significant differences between the two controllers occur around the lower value of inlet concentration, $S^{\text{ss}}_{\text{in}} = 2.0$ g/L, and the difference around this value is, as expected, significantly larger for case study B than for case study A since for case study B a larger range of inlet concentration values is considered for control design requiring further detuning of the LMPC controller for case study B to achieve robustness. On the bases of the calculated rations of γ between the two controllers, the improvement in performance to be achieved with the GSMPC controller around $S^{\text{ss}}_{\text{in}} = 2.0$ g/L is 25% for case study A and up to 132% for case study B. However, these improvements are only theoretical since they correspond to zero magnitude disturbances.

Table 3.3: Optimal nominal GSMPC and LMPC γ values for case study A

S_{in}	GSMPC	GSMPC	LMPC	LMPC	ratio of
g/L	$\gamma^{\text{op-nom}}$	$\mathbf{R}_i^{\text{op-nom}}$	$\gamma^{\text{op-nom}}$	$\mathbf{R}_i^{\text{op-nom}}$	$\gamma^{\text{op-nom}}$
2.0	0.5328	$\text{diag}\,[1\text{E}-4, 1\text{E}-8]$	0.7065	$\text{diag}\,[1\text{E}-8, 1\text{E}-6]$	1.33
2.5	0.5206	$\text{diag}\,[1\text{E}-8, 1\text{E}-8]$	0.5206	$\text{diag}\,[1\text{E}-8, 1\text{E}-6]$	1.00
3.0	0.3799	$\text{diag}\,[1, 1\text{E}-8]$	0.3822	$\text{diag}\,[1\text{E}-8, 1\text{E}-6]$	1.01

When the uncertainty is considered, i.e., the disturbance magnitude is not zero, the optimal GSMPC γ value referred to as γ^{op} and corresponding GSMPC manipulated variables weight matrix $\mathbf{R}^{\text{op}}$ were calculated according to the procedures outlined in section 3.4. The results are shown in Table 3.5 and Table 3.6 for case studies A and B respectively. The

Table 3.4: Optimal nominal GSMPC and LMPC γ values for case study B

S_{in}	GSMPC	GSMPC	LMPC	LMPC	ratio of
g/L	$\gamma^{\text{op-nom}}$	$\mathbf{R}_i^{\text{op-nom}}$	$\gamma^{\text{op-nom}}$	$\mathbf{R}_i^{\text{op-nom}}$	$\gamma^{\text{op-nom}}$
2.0	0.5328	$\text{diag}\,[1\text{E}-4, 1\text{E}-8]$	1.2366	$\text{diag}\,[1\text{E}-8, 1.6\text{E}-5]$	2.32
3.0	0.3799	$\text{diag}\,[1, 1\text{E}-8]$	0.3799	$\text{diag}\,[1\text{E}-8, 1.6\text{E}-5]$	1.00
4.0	0.1792	$\text{diag}\,[5, 1\text{E}-8]$	0.1880	$\text{diag}\,[1\text{E}-8, 1.6\text{E}-5]$	1.05

calculations for the LMPC controllers designed with uncertainty are summarized in Table 3.7 and Table 3.8. For comparison purposes, the optimal manipulated variable weights $\mathbf{R}^{\text{op}}$ obtained through the optimization procedure given in section 3.5 and the manipulated variable bounds obtained iteratively according to the procedure outlined in that section are also listed in Table 3.5 to Table 3.8. It is clear from the comparison of Table 3.3 and Table 3.4 with the values in the first columns of Table 3.5 to Table 3.8 that there is a consistent deterioration in performance for the cases with uncertainty as compared to their counterparts without uncertainty. This is indicated by a consistent increase of the γ values calculated with uncertainty and without uncertainty that result from detuning the controllers for robustness with respect to this model uncertainty.

Table 3.5: Optimal GSMPC γ and $\mathbf{R}$ values for case study A

S_{in}	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0 [g/L]	1.6171	$\text{diag}\,[900, 32644]$	± 1.01	± 0.15
2.5 [g/L]	1.1802	$\text{diag}\,[1110, 326]$	± 0.91	± 0.64
3.0 [g/L]	0.9474	$\text{diag}\,[8944, 512]$	± 0.15	± 0.31

Table 3.6: Optimal GSMPC γ and $\mathbf{R}$ values for case study B

S_{in}	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0 [g/L]	1.6171	$\text{diag}\,[900, 32644]$	± 1.01	± 0.15
3.0 [g/L]	0.9474	$\text{diag}\,[8944, 512]$	± 0.15	± 0.31
4.0 [g/L]	0.2674	$\text{diag}\,[528, 517904]$	± 0.05	± 0.07

Table 3.7: Optimal LMPC γ and $\mathbf{R}$ values for case study A

S_{in}	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0 [g/L]	2.0108	$\text{diag}\,[7560, 1064]$	± 0.72	± 0.56
2.5 [g/L]	1.3863	$\text{diag}\,[7560, 1064]$	± 0.23	± 0.20
3.0 [g/L]	1.0853	$\text{diag}\,[7560, 1064]$	± 0.12	± 0.21

Table 3.8: Optimal LMPC γ and $\mathbf{R}$ values for case study B

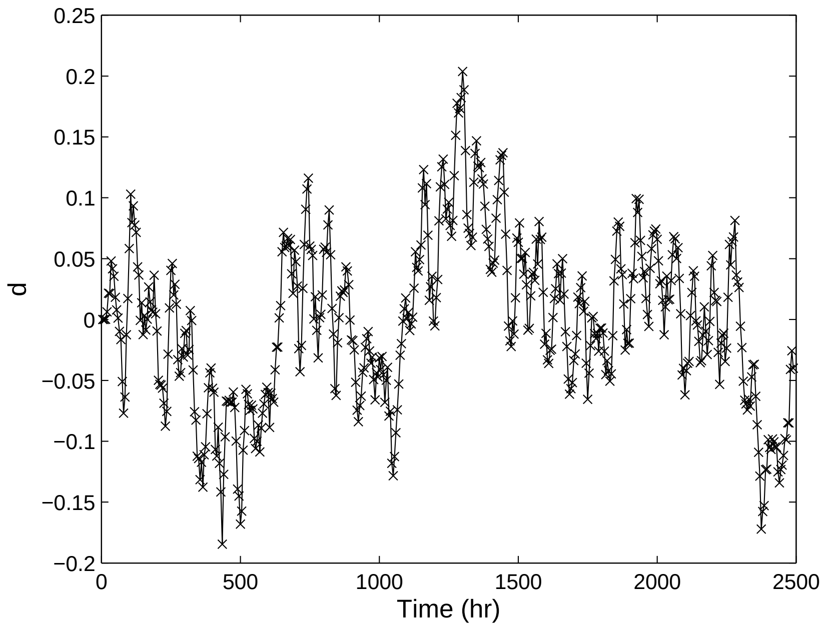
S_{in}	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0 [g/L]	1.9701	$\text{diag}\,[2868048, 17232]$	± 0.30	± 0.32
3.0 [g/L]	0.9786	$\text{diag}\,[2868048, 17232]$	± 0.09	± 0.09
4.0 [g/L]	0.2763	$\text{diag}\,[2868048, 17232]$	± 0.07	± 0.07

For the purpose of comparing the overall performance of both controllers, an average performance index was also calculated for the GSMPC by averaging the three γ^{op} values shown in Table 3.5 and Table 3.6 as done for the LMPC as per equation (3.84). Using this definition, the resulting values obtained with the GSMPC were $\gamma_{\text{GSMPC}}^{\text{op-av}} = 1.2482$ for case study A and $\gamma_{\text{GSMPC}}^{\text{op-av}} = 0.9440$ for case study B. For the LMPC the resulting γ^{op} and $\mathbf{R}_i^{\text{op}}$ for every region are shown in Table 3.7 and Table 3.8 for case studies A and B, respectively. The average performance indexes, calculated from equation (3.84) were $\gamma_{\text{LMPC}}^{\text{op-av}} = 1.4941$ for case study A and $\gamma_{\text{LMPC}}^{\text{op-av}} = 1.0705$ for case study B. On the basis of these averages, it is clear that the GSMPC outperforms the LMPC controllers for both scenarios resulting in an improvement of 19% for case study A and 13% for case study B. Although these averaged percentages of improvement achieved by the GSMPC are moderate, the improvements achieved for particular regions of operation are significant. For instance, it is evident from comparison of the first columns of Table 3.5 and Table 3.7 for case study A and Table 3.6 and Table 3.8 for case study B that the largest differences between the performances of the two controllers are still obtained around an inlet concentration of $S_{\text{in}}^{\text{ss}} = 2.0$ g/L as for the nominal case. The improvement of GSMPC over LMPC around this concentration is approximately 25% for case study A and 22% for case study B. It is also clear that the differences between the two controllers are smaller for the cases with uncertainty than for the nominal cases due to the detuning of the controllers required for achieving robustness to model uncertainty. Thus, the differences in performance between the two controllers are expected to increase as the magnitude of the disturbance becomes smaller. For smaller magnitudes of disturbance, the uncertainty bounds corresponding to the expected maximal deviations in manipulated variables are also expected to become smaller with the result that the index γ^{op} will approach the nominal values given in Tables 3.3 and 3.4. To verify this point, γ^{op} was calculated around an inlet concentration of $S_{\mathrm{in}}^{\mathrm{ss}} = 2.0$ g/L for a disturbance that is 10 times smaller than the disturbance magnitude used for the calculations shown in Tables 3.5 and 3.6. The resulting γ^{op} was 0.6948 that is considerable smaller than the value of 1.6171 in Table 3.5 and much closer, as expected, to the nominal value 0.5328 shown in Table 3.3. Thus the values listed in Table 3.3 and Table 3.4 represent the largest differences that can be obtained between the two controllers since for these tables uncertainties are not accounted for in the controllers' design.

Another important observation for Table 3.6 is that the input weights required by the GSMPC, when operating around an inlet concentration of $S_{\mathrm{in}}^{\mathrm{ss}} = 4.0$ g/L, are significantly larger than the weights calculated for other operating conditions. This has also the consequence that the LMPC controller designed for case study B, that includes the operating point corresponding to $S_{\mathrm{in}}^{\mathrm{ss}} = 4.0$ g/L and that it has to provide control for the whole region based on one single model, requires very large input weights for control as shown in Table 3.8. The use of large input weights results in highly detuned controllers that are expected to provide, as shown later in this section, sluggish closed-loop responses. A careful examination of the closed-loop system given by equation (3.32) reveals that the matrix $\mathbf{A}$ for $S_{\mathrm{in}}^{\mathrm{ss}} = 4.0$ g/L for both controllers has an eigenvalue that is very close to 1, and consequently, the closed-loop system is very close to its stability limit. Then, the optimization described in section 3.4, that requires compliance with robust stability, forces the input weights to be very large to achieve stability.

To corroborate the results of the analysis presented above, an extensive simulation study was conducted that consisted of simulating the controllers calculated in Table 3.5 to Table 3.8 for a large number of disturbances. Since the γ values calculated in the tables correspond to worst case scenarios, disturbances that lead to the worst simulated γ values were sought. Obviously, there is no systematic way to find the exact worst disturbance other than to do a brute force search of all possible disturbances and to compare the resulting values of γ. The simulations consisted in perturbing the value of the disturbance $\mathbf{d}(k)$ around the different S_{in} values corresponding to the three different regions defined in case studies A and B respectively. The code used to generate the disturbance can be found in Appendix I. Figure 3.1 shows a particular disturbance that was found to result in a large value of γ and that was consequently used for the simulation studies.

Figure 3.1: Disturbance used for case studies A and B



As mentioned in section 3.1, in existent gain-scheduled industrial applications, each one of the linear controllers that compose the gain-scheduling strategy is generally designed based on a local step-response identification of the process (Bequette, 2003). In that case, the local controller is not formally tuned for robustness with respect to nonlinear effects with the expectation that the scheduling would take care of the nonlinear behavior of the process. To assess the importance of the robust tuning proposed in this work a series of additional optimizations and simulations were conducted as follows: (i) the input weight of the gain-scheduled MPC strategy were optimized around each operating condition used for scheduling for the case where uncertainty, due to high-order nonlinear terms, is ignored, (ii) the weights obtained from the optimization above were used to simulate the closed-loop system around the different operating conditions. The weights obtained from these optimizations and the theoretical γ values, shown in Table 3.3 and Table 3.4, are as expected very small since uncertainty has been ignored. The simulated performance using these weights, shown in Figure 3.2 and Figure 3.3 for the controlled and manipulated variables when the system is operated around an inlet concentration of $S_{\mathrm{in}}^{\mathrm{ss}} = 2.0$ g/L and the same disturbances as in case studies A and B, is unacceptable since the bioreactor was driven to biomass washout. This same behavior was observed around all three operating conditions. Thus, the consideration of nonlinear effects in the tuning of the controllers is essential for acceptable operation of the bioreactor.

Figure 3.4 to Figure 3.7 show the two controlled variables and the two manipulated variables as a function of time for case study A when the system is operated around steady-states corresponding to the two extreme inlet concentration values, i.e., $S_{\mathrm{in}}^{\mathrm{ss}} = 2.0$ g/L and $S_{\mathrm{in}}^{\mathrm{ss}} = 3.0$ g/L. It is evident from Figure 3.4 and Figure 3.5 that the GSMPC reduces significantly the output variability as compared to the LMPC. The differences are especially significant for output y_2 around $S_{\mathrm{in}}^{\mathrm{ss}} = 2.0$ g/L as predicted by the analysis. Figure 3.8 to Figure 3.11 show the two controlled variables and the two manipulated variables as a function of time for case study B for the two extreme inlet concentration values used for that case study, i.e., $S_{\mathrm{in}}^{\mathrm{ss}} = 2.0$ g/L and $S_{\mathrm{in}}^{\mathrm{ss}} = 4.0$ g/L. The simulations clearly corroborate that the GSMPC provides significantly better performance than the LMPC for case study B.

Figure 3.2: Controlled variables, nominal LMPC versus GSMPC for case study A when the process is operating around $S_{\text{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC)

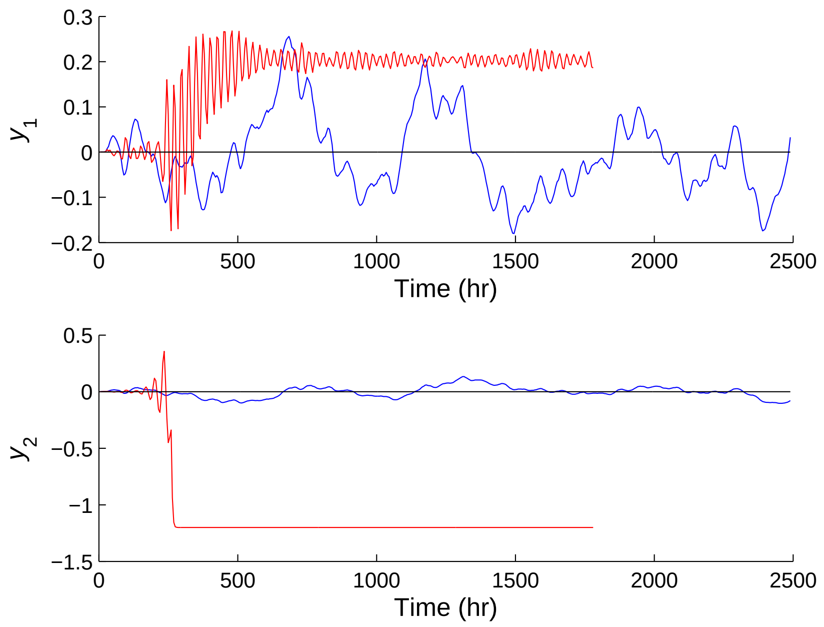


Figure 3.3: Manipulated variables, nominal LMPC versus GSMPC for case study B when the process is operating around $S_{\text{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC)

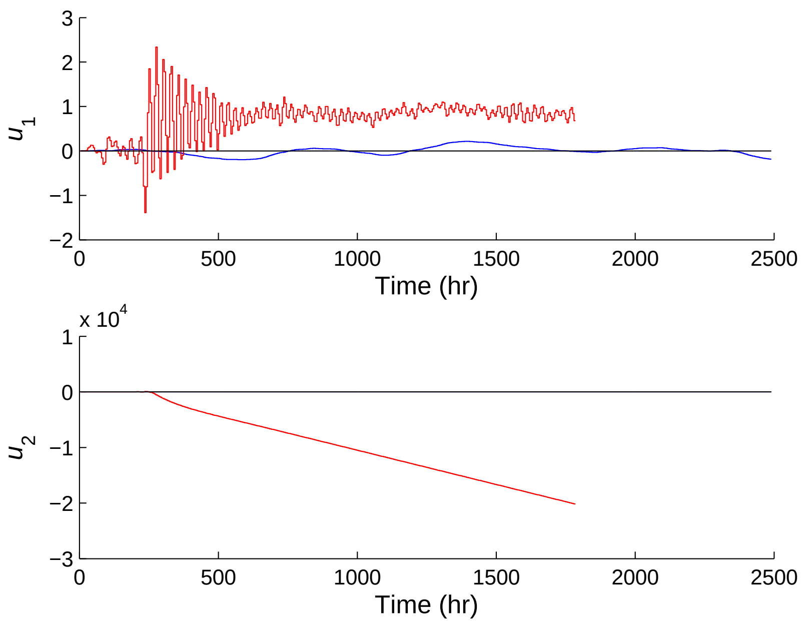


Figure 3.4: Controlled variables results for case study A when the process is operating around $S_{\text{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC)

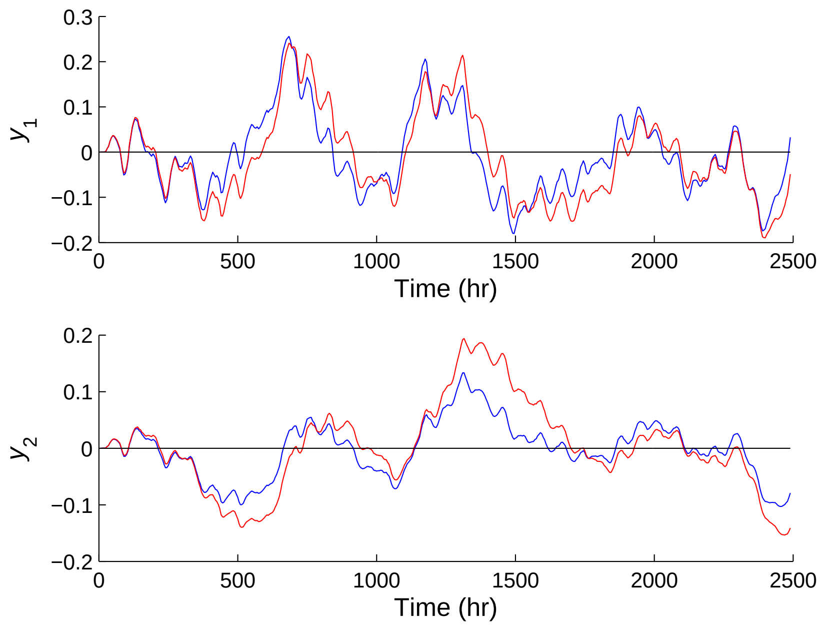


Figure 3.5: Manipulated variables results for case study A when the process is operating around $S_{\text{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC)

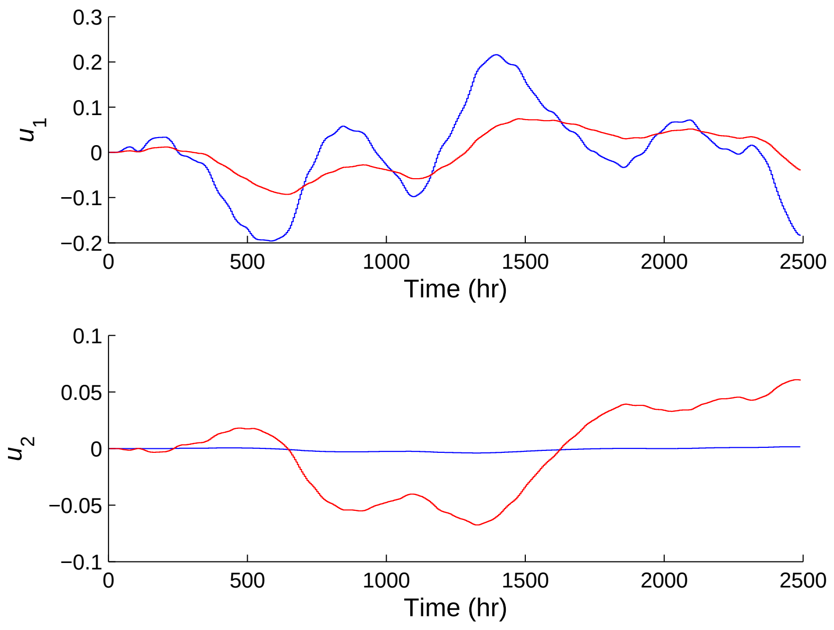
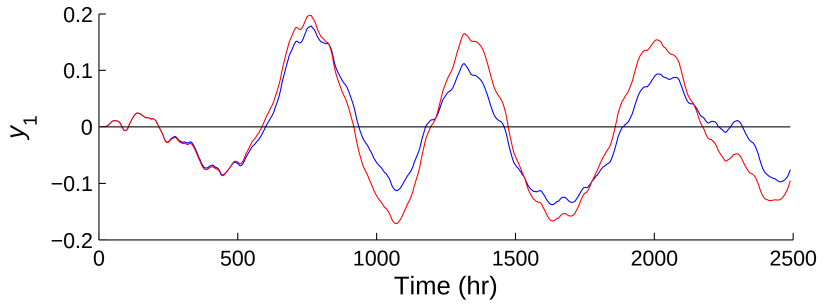


Figure 3.6: Controlled variables results for case study A when the process is operating around $S_{\text{in}} = 3.0$ g/L, (blue=GSMPC), (red=LMPC)



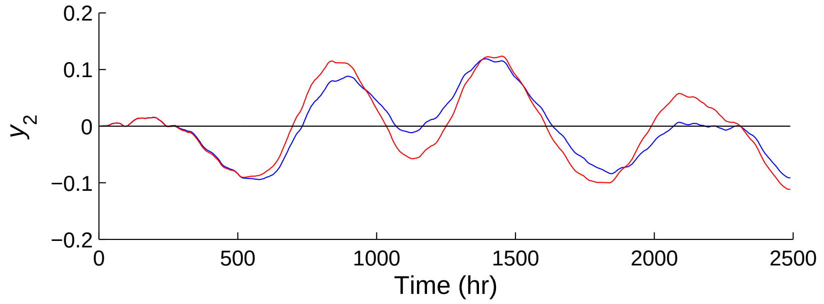
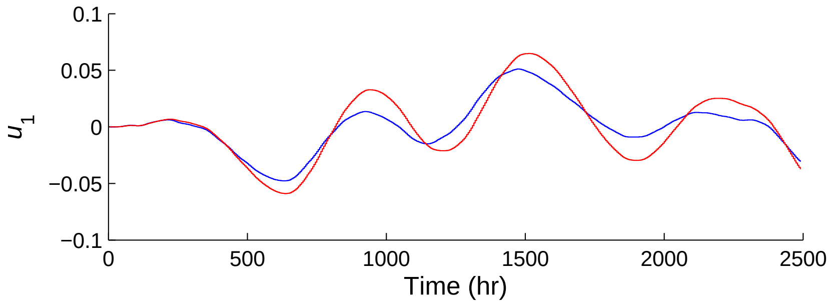


Figure 3.7: Manipulated variables results for case study A when the process is operating around $S_{\text{in}} = 3.0$ g/L, (blue=GSMPC), (red=LMPC)



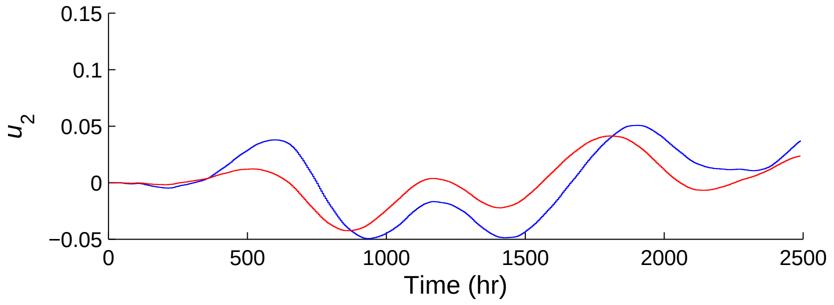


Figure 3.8: Controlled variables results for case study B when the process is operating around $S_{\text{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC)

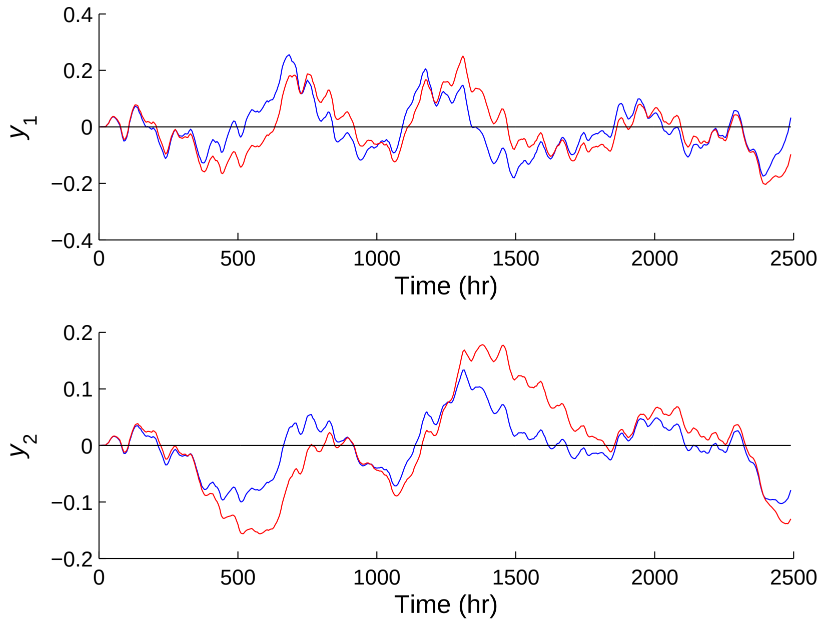


Figure 3.9: Manipulated variables results for case study B when the process is operating around $S_{\text{in}} = 2.0$ g/L, (blue=GSMPC), (red=LMPC)

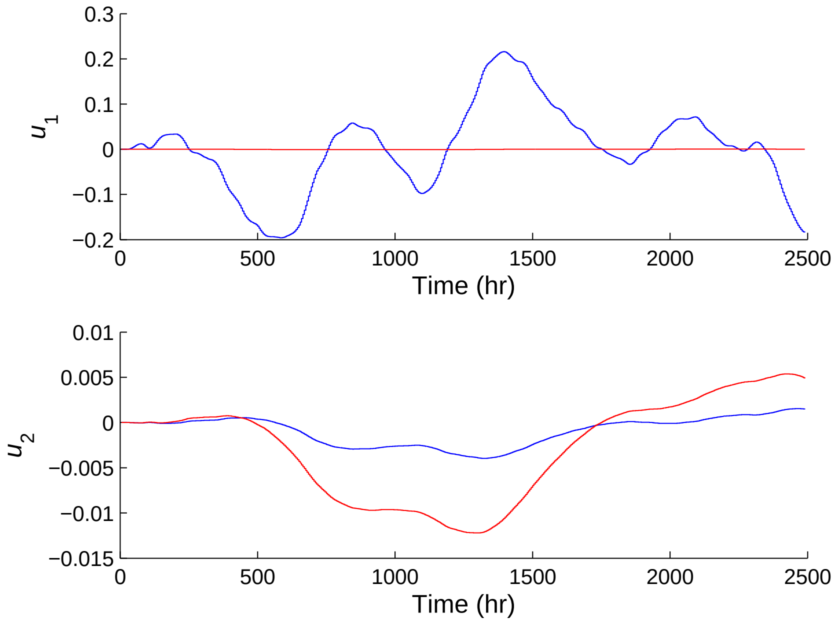


Figure 3.10: Controlled variables results for case study B when the process is operating around $S_{\text{in}} = 4.0$ g/L, (blue=GSMPC), (red=LMPC)

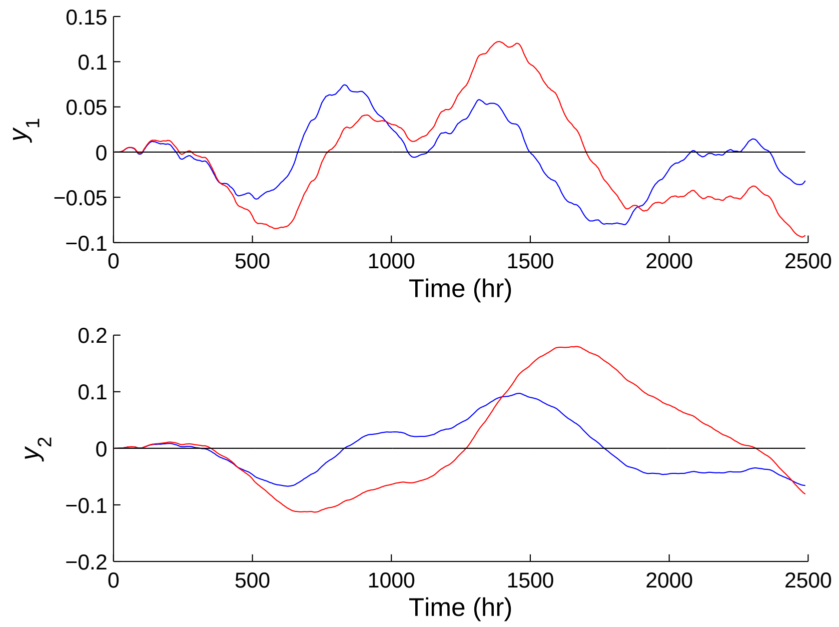
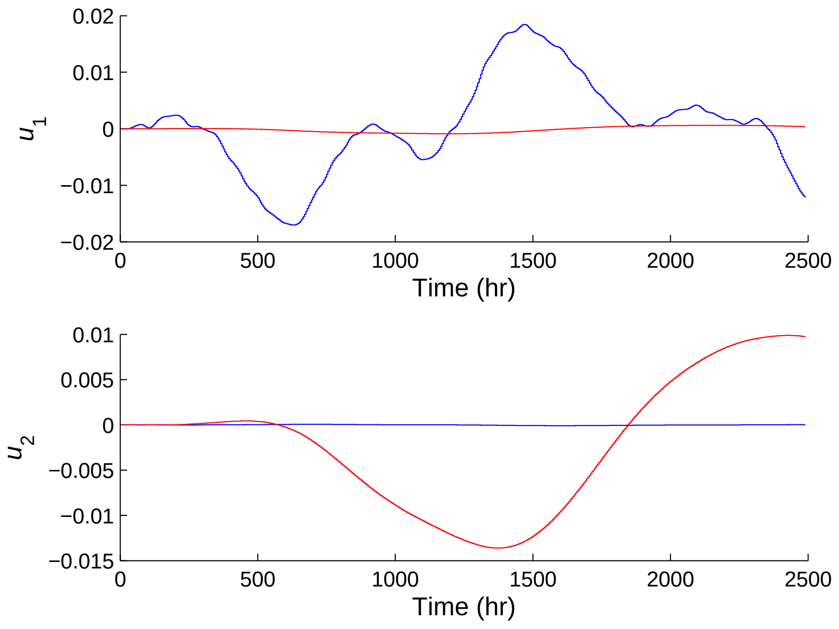


Figure 3.11: Manipulated variables results for case study B when the process is operating around $S_{\text{in}} = 4.0$ g/L, (blue=GSMPC), (red=LMPC)



To better quantify the differences in variability obtained in the simulations for the two controllers and for both case studies, γ values were calculated based on simulation results according to the following equation:

$$\gamma^{\text{sim}} = \frac{||\mathbf{y}^{\text{real process}}||_2}{||\boldsymbol{\nu}||_2} \tag{3.88}$$

The calculated values of γ^{sim} are presented in Table 3.9 and Table 3.10 for case studies A and B respectively. The calculations were done for the extreme values of inlet concentration considered in each one of the two case studies. The γ^{sim} results in Table 3.9 and Table 3.10 show that in all cases the GSMPC controller performance surpasses that of the linear MPC as predicted by the analysis. It is also clear from comparisons of the γ^{sim} values in Table 3.9 and Table 3.10 to the γ^{op} in Table 3.5 to Table 3.8 that the analysis is conservative since the simulated values γ^{sim} are lower than the analytical values γ^{op}. It should be remembered that the simulated values were obtained for a particular disturbance that was found, by trial and error, to result in a large γ^{sim} but a different disturbance may give a larger value of γ^{sim} closer to the analytical bound. However, despite the fact that the analysis is conservative as compared to the simulations, the analysis has correctly and consistently predicted that the GSMPC controller performs better than the LMPC. The improvements achieved with the GSMPC controller over the LMPC, based on the γ^{sim} results presented in Table 3.9 and Table 3.10, range from 23% to 73% as shown in these tables. It is clear from Table 3.10 that the performance improvement achieved with the GSMPC versus the LMPC is very significant, 73%, when the system is operated around the steady-state corresponding to $S_{\text{in}} = 4.0$ g/L. The key difference between the two controllers around this inlet concentration is that, as mentioned earlier in this section, the optimization resulted in very large input weights for the LMPC that were needed to satisfy the robust stability condition. These large input weights caused the LMPC controller to be very sluggish resulting in a very large variability as compared to GSMPC.

An additional reason for conducting the simulation study was to test whether the manipulated variables bounds, used as uncertainty in the analysis and calculated by the iterative procedure in section 3.4, were not violated in simulations. All the simulations

Table 3.9: Simulation results for case study A

S_{in}	$\gamma^{\text{sim}}_{\text{GSMPC}}$	$\gamma^{\text{op}}_{\text{GSMPC}}$	$\gamma^{\text{sim}}_{\text{LMPC}}$	$\gamma^{\text{op}}_{\text{LMPC}}$	$\frac{\gamma^{\text{sim}}_{\text{LMPC}}}{\gamma^{\text{sim}}_{\text{GSMPC}}}$
2.0 [g/L]	0.2316	1.6171	0.2856	2.0108	1.23
3.0 [g/L]	0.2203	0.9474	0.2741	1.0853	1.24

Table 3.10: Simulation results for case study B

S_{in}	$\gamma^{\text{sim}}_{\text{GSMPC}}$	$\gamma^{\text{op}}_{\text{GSMPC}}$	$\gamma^{\text{sim}}_{\text{LMPC}}$	$\gamma^{\text{op}}_{\text{LMPC}}$	$\frac{\gamma^{\text{sim}}_{\text{LMPC}}}{\gamma^{\text{sim}}_{\text{GSMPC}}}$
2.0 [g/L]	0.2316	1.6171	0.2856	1.9701	1.23
4.0 [g/L]	0.1385	0.2674	0.2403	0.2763	1.73

including the ones shown in Figure 3.5, Figure 3.7, Figure 3.9 and Figure 3.11 verified the bounds calculated by the procedure described in section 3.4.

Finally, the gain-scheduled controllers presented above, were designed for operation around specific operating points as defined by the inlet concentration average values. However, to complete the control strategy, it is also desirable to control the system during transitions between different average inlet concentration values. Such situation will arise when large changes occur in the average inlet concentration as shown in Figure 3.12. For such large shifts in inlet concentration it was found that the corresponding manipulated variable moves were large and therefore the uncertainty bounds are very large. For such large uncertainty it was found that the performances of the linear and gain-scheduled controllers were very similar. Therefore, a simple LMPC was used to control the system during the large transitions between different operating conditions. This LMPC was based on the model corresponding to the inlet concentration $S^{\text{ss}}_{\text{in}} = 2.5$ g/L and it was designed to satisfy two conditions: (i) robust stability for all the vertices considered in the LMI formulation and (ii) manipulated variable constraints given by $|\mathbf{u}| \leq 1$ in deviation variables. The resulting controller that satisfies both these conditions is given by an input weight of $\mathbf{R}_i = \text{diag}\,[480, 258]$. The response of the system during the transition defined by the change in inlet concentration is shown in Figure 3.13. From this figure it is clear that the closed-loop

Figure 3.12: Disturbance used for transition between S_{in} studies

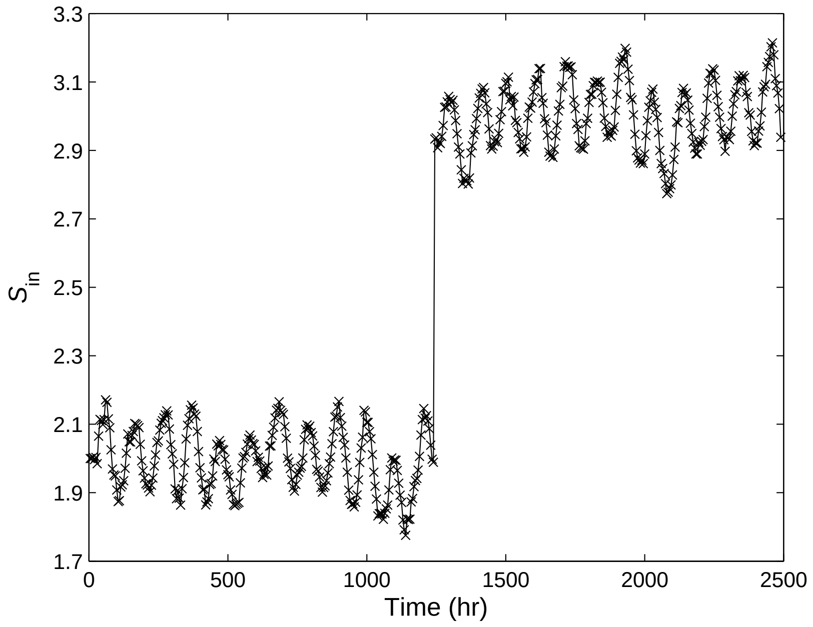
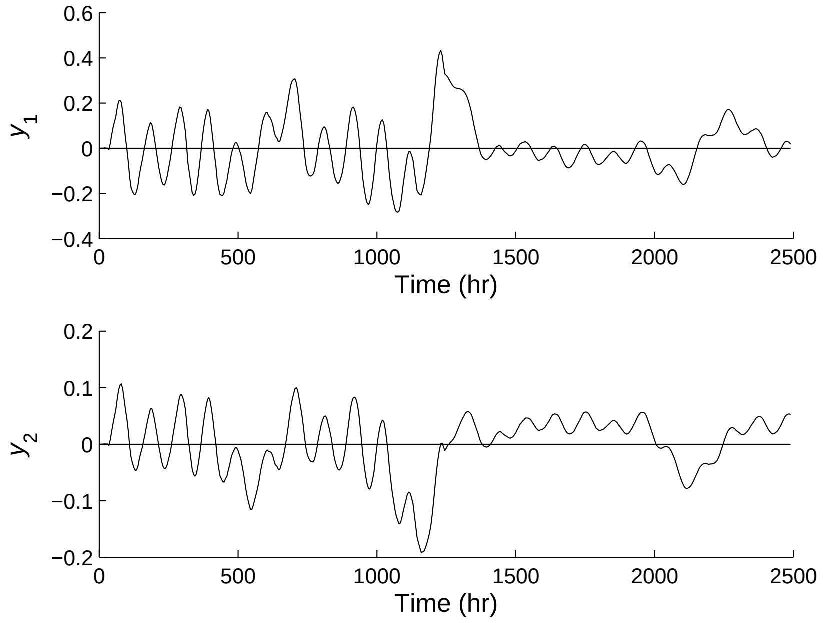


Figure 3.13: Controlled variables for transition between S_{in} studies



performance during the shift between two different average values of inlet concentration is acceptable. The simulation study during the transition was performed by using first the controller corresponding to inlet concentration $S_{\text{in}} = 2.0$ g/L followed by the LMPC during the transition and using finally the controller corresponding to inlet concentration of $S_{\text{in}} = 3.0$ g/L.

3.6 Conclusions

This chapter proposed a methodology for the design and testing of gain-scheduled model predictive controllers. A key idea of the work is to model the process by an empirical nonlinear state-affine model that can be identified from input-output data. This facilitates the practical application of the method since it does not require the use of a detailed mechanistic model of the process for control analysis and design. The nonlinear state-affine model can be split into a nominal model that considers just the linear terms and an uncertainty model that accounts for the nonlinear terms of the state-affine model. The linear part of this model is used for the design of an MPC controller.

After the model has been obtained robust stability and robust performance tests can be formulated as a finite number of linear matrix inequalities. Since it is required for the tests to know the limits of the uncertainty that the model can tolerate, an analytical iterative strategy was developed to obtain the value of these uncertainty bounds.

The proposed GSMPC is composed of a set of MPC controllers where each becomes active in different regions of the manipulated variable space. The input weight matrix of each MPC is calculated to minimize a norm of the feedback error normalized by the norm of the disturbances. For the two scenarios tested, the analytical results show that the performance of the GSMPC is consistently superior to the performance of a LMPC. However, the analysis also predicts that the closed-loop performance of both the GSMPC and LMPC will become increasingly similar as the magnitude of the disturbance is increased. Such magnitude increase was shown to result in an increase in manipulated variable action with a resulting increase of nonlinear effects according to the structure of the state-affine model used in the work. Then, when the LMPC and GSMPC controllers are detuned to achieve robustness with respect to the uncertainty related to the system nonlinearity, their closed performance becomes increasingly similar. The simulation results confirm the analytical results indicating that for the process studied a GSMPC controller performs significantly better than a LMPC.

Finally, the method proposed in this chapter can be effectively used by the practitioner to decide whether a gain-scheduled strategy is needed to improve closed-loop performance. The decision, as shown by the case studies, depends both on the range of operating conditions considered for operation and the magnitude of the disturbances entering the process since this magnitude is related, through the control actions, to the nonlinearity around a given operating point.

Although the procedures presented in this chapter require a significant computational effort, they are a viable alternative to a brute force simulation-based search of a worst case scenario among the infinite possible combinations of disturbances and input weights.

Chapter 4

Additional work for the gain-scheduling MPC

As mentioned in section 2.2 the objective function of a MPC algorithm is comprised of two terms: one term that penalizes the deviations of the controlled variables form its predefined set-point and another term that penalizes the movement of the manipulated variables. The MPC controller described in equation (3.26) considers both terms. For simplicity, the methodology to design a robust gain-scheduled controller presented in the previous chapter was initially designed to minimize a performance index γ that has the following structure

$$\gamma = \frac{||\mathbf{e}||_2}{||\nu||_2} \tag{4.1}$$

The methodology can be further improved by adding in the calculation of the performance index γ the effect of the manipulated variables. In that way it will be possible to assess whether the controller leads to less output variability but not at the cost of an exaggerated increase in control action. Accordingly, the new index γ can be represented by the following equation

$$\gamma = \frac{||\mathbf{e}||_2 + c_e\,||\mathbf{u}||_2}{||\nu||_2} \tag{4.2}$$

where the term c_e can be viewed as a weighting term that accounts for the contribution of $\mathbf{u}$ on γ. The idea behind this addition is to avoid excessive movements of the manipulated variables that may result in hardware wear and/or failure.

The first step to design a robust gain-scheduled controller based on the minimization of the γ index defined in equation (4.2) consists in modifying the closed-loop system representation given by equation (3.32) to include the effect of the controlled and manipulated variables as per the following equation:

$$\left[\begin{array}{c} \mathbf{x}^{\mathrm{L}}(k+1) \\ \mathbf{x}(k+1) \\ \mathbf{u}(k) \\ \mathbf{d}(k+1) \\ \hline \mathbf{y}(k) \\ \mathbf{u}(k) \end{array}\right] = \left[\begin{array}{cccc|c} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} & \mathbf{A}_{14} & \mathbf{B}_{11} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} & \mathbf{A}_{24} & \mathbf{B}_{21} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} & \mathbf{A}_{34} & \mathbf{B}_{31} \\ \mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{A}_{44} & \mathbf{B}_{41} \\ \hline \mathbf{C}_{11} & \mathbf{C}_{12} & \mathbf{C}_{13} & \mathbf{C}_{14} & \mathbf{D}_{11} \\ \mathbf{C}_{21} & \mathbf{C}_{22} & \mathbf{C}_{23} & \mathbf{C}_{24} & \mathbf{D}_{21} \end{array}\right] \left[\begin{array}{c} \mathbf{x}^{\mathrm{L}}(k) \\ \mathbf{x}(k) \\ \mathbf{u}(k-1) \\ \mathbf{d}(k) \\ \hline \nu(k) \end{array}\right] \tag{4.3}$$

$\mathbf{A}_{1j}$, $j = 1, \ldots, 4$ is calculated as in equation (3.33), $\mathbf{A}_{2j}$, $j = 1, \ldots, 4$ is calculated as in equation (3.34), $\mathbf{A}_{3j}$, $j = 1, \ldots, 4$ is calculated as in equation (3.35), $\mathbf{A}_{4j}$, $j = 1, \ldots, 4$ is calculated as in equation (3.36), $\mathbf{B}_{j1}$, $j = 1, \ldots, 4$ is calculated as in equation (3.37), $\mathbf{C}_{1j}$, $j = 1, \ldots, 4$ is calculated as in equation (3.38) and $\mathbf{D}_{11}$ is calculated as in equation (3.39). $\mathbf{C}_{2j}$, $j = 1, \ldots, 4$ is calculated as follows:

$$[\mathbf{C}_{21}]_{n_u \times n_x} = c_e \mathbf{K}_{\mathrm{MPC}} \left(-\mathbf{\Psi} + \mathbf{N}_2 \mathbf{C}\right) \tag{4.4a}$$

$$[\mathbf{C}_{22}]_{n_u \times n_x} = -c_e \mathbf{K}_{\mathrm{MPC}} \mathbf{N}_2 \mathbf{C}^{\mathrm{un}} \tag{4.4b}$$

$$[\mathbf{C}_{23}]_{n_u \times n_u} = c_e \mathbf{I}_{n_u} - \mathbf{K}_{\mathrm{MPC}} \mathbf{\Gamma} \tag{4.4c}$$

$$[\mathbf{C}_{24}]_{n_u \times n_d} = -c_e \mathbf{K}_{\mathrm{MPC}} \mathbf{N}_2 \mathbf{W}_F \tag{4.4d}$$

and $\mathbf{D}_{21}$ is calculated as

$$[\mathbf{D}_{21}]_{n_u \times n_d} = \mathbf{0}_{n_u \times n_d} \tag{4.5}$$

The design of a robust state-affine based gain-scheduled MPC controller consists of two steps. The first step calculates the uncertainty bounds according to procedure 1 presented in section 3.4. The second step optimizes the manipulated variables weight matrix $\mathbf{R}$ according to procedure 2 presented in section 3.4. In order to account for the effect of both the controlled and manipulated variables the following changes to the procedure presented in the previous chapter are made:

1. Step 2 of procedures 1 and 2 should include as a design parameter the value of c_e.
2. In step 3.4.3 of procedure 2 the matrices that must be calculated are those of equations (3.33) to (3.39) and (4.4) to (4.5).
3. Step 3.4.3.1 of procedure 2 uses the closed-loop model given by equation (4.3).

To show the effect that the term c_e has on the design and performance the next section presents a case study where the performance index γ used to design the robust controller considers both the contributions of the controlled and the manipulated variables.

4.1 Case study

The case study is the bioreactor that was previously used in section 3.5, the only significant modification is that a new input sequence was used for identification purposes. One of the consequences of using a new input sequence was that the error between the actual and predicted values of the controlled variables decreases compared to the values obtained for the identification of the previous chapter. Accordingly, because of the new input sequences the normalization formulas were changed to:

$$x_i = \frac{c_i - c_i^{\text{sp}}}{c_i^{\text{dv}}},\ i = 1, 2 \tag{4.6}$$

$$u_1 = \frac{D - D^{\text{ss}}}{D^{\text{dv}}} \tag{4.7}$$

$$u_2 = \frac{Dc_{3f} - (Dc_{3f})^{\text{ss}}}{(Dc_{3f})^{\text{dv}}} \tag{4.8}$$

where D^{ss} and $(Dc_{3f})^{\text{ss}}$ are the steady-state values corresponding to the operating condition around which a linear model is identified for the purpose of designing a local MPC controller. Similar to the previous case study the value of the substrate's feed concentration S_{in} will be considered as the main disturbance affecting the process. Therefore, the objective of both control strategies consists in rejecting the disturbance and keeping the controlled variables within their predefined set-point ($c_1^{\text{sp}} = 0.16$ g/L and $c_2^{\text{sp}} = 0.06$ g/L).

The selected operating regions for the controller are as before $S_{\text{in}} = 2.0$ g/L, $S_{\text{in}} = 2.5$ g/L and $S_{\text{in}} = 3.0$ g/L.

In order to keep the value of the controlled and manipulated variables for the three regions between -1 and $+1$, the values of c_i^{dv} in equation (4.6) were selected as $c_1^{\text{dv}} = 0.075$ and $c_2^{\text{dv}} = 0.0375$, the value of D^{dv} in equation (4.7) was selected as 2.4440×10^{-3}, the value of $(Dc_{3f})^{dv}$ in equation (4.8) was selected as 1.1220×10^{-5}, the values of D^{dv} and $\left(Dc_{3f}^{\text{ss}}\right)$ required by equations (4.7) and (4.8) can be found in Table 4.1.

Table 4.1: Values for normalization

	D^{ss}	$(Dc_{3f})^{\text{ss}}$
operating condition around $S_{\text{in}} = 2.0$ g/L	0.3764	2.2823×10^{-3}
operating condition around $S_{\text{in}} = 2.5$ g/L	0.3851	2.3259×10^{-3}
operating condition around $S_{\text{in}} = 3.0$ g/L	0.3891	2.3459×10^{-3}

The parameters of the GSMPC and LMPC controllers used for this case study are $p = 200$ and all the other parameters are the same as in the previous chapter.

Similar to the approach followed in the previous chapter, a preliminary study was conducted to assess whether using a GSMPC can provide better performance as compared to an LMPC. This was done by comparing the γ values for the GSMPC and LMPC designed based on the nominal models, i.e., when the uncertainty is zero. For comparison purposes all the initial simulations were done with $c_e = 0$, i.e., no penalty of the manipulated variables. The optimal values of γ calculated following the procedure outlined in the previous section are shown in Table 4.2 and Table 4.3.

The results in these tables clearly indicate that GSMPC is superior to LMPC as indicated by the smaller γ values obtained with the GSMPC. On the basis of the calculated ratios of γ between the two controllers, the improvement in performance to be achieved with the GSMPC controller around $S_{\text{in}} = 2.0$ g/L is 15% and around $S_{\text{in}} = 3.0$ g/L is 18%. A similar result, i.e., smaller γ for the GSMPC controller was also found for the case

Table 4.2: Optimal nominal GSMPC γ and manipulated variables weight matrix for $c_e = 0$

S_{in} [g/L]	GSMPC $\gamma^{\text{op}-\text{nom}}$	$\mathbf{R}_i^{\text{op}-\text{nom}}$
2.0	0.0864	$\text{diag}\,[1.18\text{E}-1, 1.41\text{E}-8]$
2.5	0.0857	$\text{diag}\,[2.11\text{E}2, 9.06\text{E}-9]$
3.0	0.0881	$\text{diag}\,[1.89\text{E}3, 2.62\text{E}-8]$

Table 4.3: Optimal nominal LMPC γ and manipulated variables weight matrix for $c_e = 0$

S_{in} [g/L]	GSMPC $\gamma^{\text{op}-\text{nom}}$	$\mathbf{R}_i^{\text{op}-\text{nom}}$
2.0	0.0997	$\text{diag}\,[1.48\text{E}-4, 6.21\text{E}-7]$
2.5	0.0978	$\text{diag}\,[1.48\text{E}-4, 6.21\text{E}-7]$
3.0	0.1047	$\text{diag}\,[1.48\text{E}-4, 6.21\text{E}-7]$

study of the previous chapter. However, the relative improvements of the GSMPC with respect to the LMPC were distributed differently as a function of the inlet concentration S_{in}. More specifically, the new input sequence and normalization had the following effects: (i) the value of $\gamma^{\text{op}-\text{nom}}$ decreased, (ii) the percentage improvement for the operating zone $S_{\text{in}} = 2.0$ g/L decreased from 33% to 15%, (iii) the percentage improvement for the operating zone $S_{\text{in}} = 2.5$ g/L decreased to 14% and (iii) the percentage improvement for the operating zone $S_{\text{in}} = 3.0$ g/L increased from 1% to 18%. If one sums up the error percentages the differences are similar than in the case study presented in chapter 3, i.e., approximately 10% - 15% at each operating condition making a total of 40% when the errors are added. These effects can be explained by the fact that for the newly identified models the errors are distributed more uniformly between the three operating regions.

When uncertainty is considered, the optimal GSMPC γ value and corresponding manipulated variable weight matrix were calculated according to the procedure presented in section 3.4 with the proper modifications described in this chapter. The results are shown in Table 4.4. The calculation for the LMPC controller designed with uncertainty is presented in Table 4.5.

For comparison purposes, the optimal manipulated variable weight matrix $\mathbf{R}$ and the manipulated variables bounds are also listed in Table 4.4 and Table 4.5. It is clear from

Table 4.4: Optimal GSMPC γ and manipulated variables weight matrix for $c_e = 0$

S_{in} [g/L]	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0	0.4099	$\text{diag}\,[74576, 37648]$	± 0.2179	± 0.1542
2.5	0.4303	$\text{diag}\,[74768, 45840]$	± 0.5861	± 0.1397
3.0	0.4700	$\text{diag}\,[74576, 71504]$	± 0.6432	± 0.0672

Table 4.5: Optimal LMPC γ and manipulated variables weight matrix for $c_e = 0$

S_{in} [g/L]	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0	0.4270	$\text{diag}\,[62288, 11144]$	± 1.7176	± 1.2804
2.5	0.4350	$\text{diag}\,[62288, 11144]$	± 0.6735	± 0.4981
3.0	0.4885	$\text{diag}\,[62288, 11144]$	± 0.6058	± 0.4542

the comparison of Table 4.2 and Table 4.3 against Table 4.4 and Table 4.5 that there is a consistent deterioration in performance for the cases with uncertainty as compared with the cases without uncertainty. This is indicated by the increase of the γ values obtained when uncertainty is considered.

As stated above, the effect of the manipulated variables have not been included in the objective function of the controllers obtained in Table 4.2 to Table 4.5, i.e., $c_e = 0$. The effect that the manipulated variables movements have on γ can be accounted for with the term c_e. When this weight is different than zero, the new γ value is obtained from:

$$\frac{||\mathbf{y}_1 + \mathbf{y}_2 + c_e\,(\mathbf{u}_1\mathbf{u}_2)||_2}{||\boldsymbol{\nu}||_2} = \gamma \tag{4.9}$$

For comparison purposes both a GSMPC and a LMPC controller were designed with $c_e > 0$. For two different values of c_e, the corresponding γ and manipulated variables weight matrix $\mathbf{R}$ for the GSMPC controller are presented in Table 4.6 and Table 4.7 and for the LMPC controller are presented in Table 4.8 and Table 4.9.

For the purpose of comparing the overall performance of both controllers, an average performance index was also calculated for the GSMPC and LMPC by averaging the three γ values shown in Table 4.4 to Table 4.9. The average γ values are shown in Table 4.10.

Table 4.6: Optimal GSMPC γ and manipulated variables weight matrix for $c_e = 0.1$

S_{in} [g/L]	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0	0.4102	diag $[74576, 50704]$	± 0.2179	± 0.1195
2.5	0.4314	diag $[72528, 16328]$	± 0.5941	± 0.3480
3.0	0.4740	diag $[72528, 29200]$	± 0.6420	± 0.1563

Table 4.7: Optimal GSMPC γ and manipulated variables weight matrix for $c_e = 0.25$

S_{in} [g/L]	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0	0.4615	diag $[50960, 42768]$	± 0.3075	± 0.1377
2.5	0.4340	diag $[66320, 59152]$	± 0.6508	± 0.1058
3.0	0.4757	diag $[52048, 10000]$	± 0.6693	± 0.3432

Table 4.8: Optimal LMPC γ and manipulated variables weight matrix for $c_e = 0.1$

S_{in} [g/L]	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0	0.4295	diag $[64336, 6664]$	± 1.6308	± 2.0820
2.5	0.4364	diag $[64336, 6664]$	± 0.6537	± 0.8322
3.0	0.4911	diag $[64336, 6664]$	± 0.5905	± 0.7586

Table 4.9: Optimal LMPC γ and manipulated variables weight matrix for $c_e = 0.25$

S_{in} [g/L]	γ^{op}	$\mathbf{R}_i^{\text{op}}$	u_1 bound	u_2 bound
2.0	0.5697	diag $[67408, 6024]$	± 1.5580	± 2.3032
2.5	0.4393	diag $[67408, 6024]$	± 0.6287	± 0.9306
3.0	0.5187	diag $[67408, 6024]$	± 0.5678	± 0.8462

Table 4.10: Comparison of γ optimal average for several values of c_e

c_e	GSMPC $\gamma^{\text{op-av}}$	LMPC $\gamma^{\text{op-av}}$
0	0.4367	0.4502
0.1	0.4385	0.4523
0.25	0.4571	0.5090

On the basis of these averages, it is clear that the GSMPC outperforms the LMPC controllers for all the values of c_e tested. This means that even when the controllers are compared on a basis of a combination of output error and control effort the GSMPC is superior to the LMPC controller. To corroborate the results of the analysis presented above, an extensive simulation study was conducted that consisted of simulating the controllers calculated in Table 4.4 to Table 4.9 for a large number of disturbances. The simulations consisted in perturbing the value of the disturbance $d(k)$ around the different S_{in} values corresponding to the three different operating regions. The code used to generate the disturbance can be found in Appendix I. Figure 4.1 shows a particular disturbance that was found to result in a large value of γ and that was consequently used for the simulation studies.

To better quantify the differences in variability obtained in the simulations for the two controllers and for those cases when $c_e = 0$, $c_e = 0.1$ and $c_e = 0.25$, γ values were calculated based on simulation results according to equation (4.2) and are referred heretofore as γ^{sim}. The calculated values of γ^{sim} are presented in Table 4.11 to Table 4.13.

Figure 4.1: Disturbance used for simulation studies, (a) System operating around $S_{\text{in}} = 2.0$ g/L, (b) System operating around $S_{\text{in}} = 3.0$ g/L

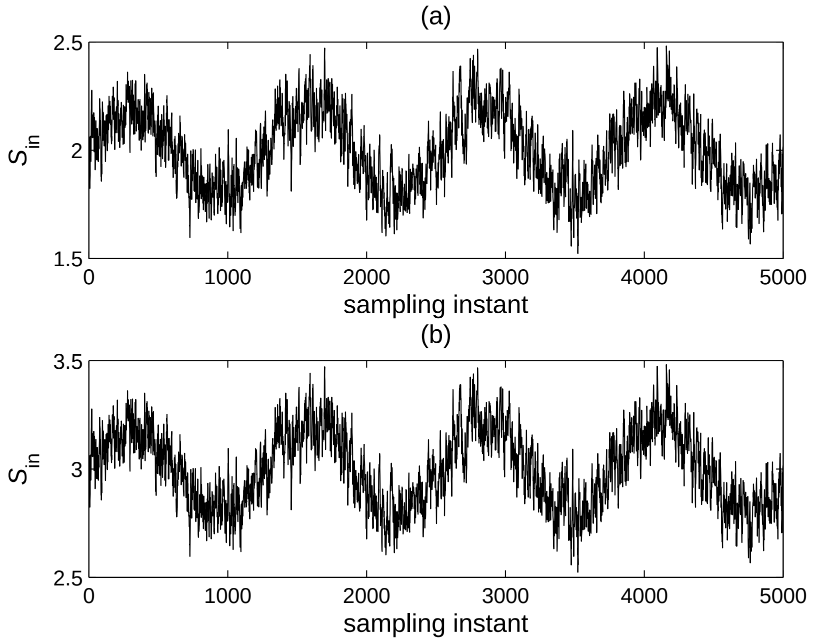


Table 4.11: Simulation results for $c_e = 0$

S_{in} [g/L]	GSMPC γ^{sim}	LMPC γ^{sim}	LMPC γ^{sim} / GSMPC γ^{sim}
2.0	0.2071	0.2078	1.00
3.0	0.1701	0.1889	1.11

Table 4.12: Simulation results for $c_e = 0.1$

S_{in} [g/L]	GSMPC γ^{sim}	LMPC γ^{sim}	LMPC γ^{sim} / GSMPC γ^{sim}
2.0	0.2073	0.2116	1.02
3.0	0.1709	0.1920	1.12

Table 4.13: Simulation results for $c_e = 0.25$

S_{in} [g/L]	GSMPC γ^{sim}	LMPC γ^{sim}	LMPC γ^{sim} / GSMPC γ^{sim}
2.0	0.2082	0.2179	1.04
3.0	0.1586	0.1989	1.25

The calculations were done for the extreme values of inlet concentration considered in each one of the two case studies. The γ^{sim} results in Table 4.11 to Table 4.13 show that in all cases the GSMPC controller performance surpasses that of the LMPC as predicted by the analysis. It is also clear from comparisons of the γ^{sim} values of Table 4.11 to Table 4.13 to the γ^{op} of Table 4.4 to Table 4.9 that the analysis is conservative since the simulated values γ^{sim} are lower than the analytical values γ^{op}. However, it was found that the new models together with the new normalizations used in this chapter, i.e., equation (4.6) to equation (4.8) led to less conservative results to those obtained in the previous chapter. It should be remembered that the simulated values were obtained for a particular disturbance that was found, by trial and error, to result in a large γ^{sim} but a different disturbance may give a larger value of γ^{sim} closer to the analytical bound. However, despite the fact that the analysis is conservative as compared to the simulations, the analysis has correctly and consistently predicted that the GSMPC outperforms the LMPC.

An additional reason for conducting the simulation study was to test whether the manipulated variable bounds, used as uncertainty in the analysis and calculated by the iterative procedure in section 3.4, were not violated in simulations. For comparison purposes the maximum absolute value of the manipulated variables are presented in Table 4.14 and Table 4.15 for the GSMPC and LMPC, respectively. The simulations also corroborated that the analytical bounds on manipulated variables were not violated.

Table 4.14: Maximum absolute value of the manipulated variables achieved during simulation for GSMPC

S_{in} [g/L]	c_e	max(abs($\mathbf{u}_1$))	$\mathbf{u}_1$ bound	max(abs($\mathbf{u}_2$))	$\mathbf{u}_2 bound$
2.0	0	0.14	±0.21	0.10	± 0.15
3.0	0	0.44	±0.64	0.04	± 0.07
2.0	0.1	0.15	±0.22	0.08	±0.12
3.0	0.1	0.45	±0.64	0.10	±0.16
2.0	0.25	0.21	±0.31	0.09	±0.14
3.0	0.25	0.52	±0.67	0.23	±0.34

Table 4.15: Maximum absolute value of the manipulated variables achieved during simulation for LMPC

S_{in} [g/L]	c_e	max(abs($\mathbf{u}_1$))	$\mathbf{u}_1$ bound	max(abs($\mathbf{u}_2$))	$\mathbf{u}_2 bound$
2.0	0	0.45	±1.72	0.33	± 1.28
3.0	0	0.32	±0.61	0.23	± 0.45
2.0	0.1	0.43	±1.63	0.53	±2.08
3.0	0.1	0.31	±0.59	0.39	±0.76
2.0	0.25	0.41	±1.56	0.59	±2.30
3.0	0.25	0.30	±0.57	0.44	±0.85

4.2 Conclusions

This chapter presented a specific modification to the methodology presented in chapter 3 that involves the penalization of the manipulated variables in the performance index used for optimization. The main conclusion is that even on the basis of this new index that includes a penalty on control action, GSMPC still outperforms the LMPC algorithm indicating that GSMPC achieves a better overall performance both in terms of error minimization and control effort.

Chapter 5

Structured singular valued based robust nonlinear model predictive controller using Volterra series models[*]

Overview

A methodology is proposed for designing a robust non-linear model predictive controller based on a Volterra series model with uncertain coefficients. A key benefit of using the Volterra series model is that it can be split into a nominal and an uncertainty model thus permitting the application of robust analysis tools. The controller is based on the on-line solution of a robust performance test based on a structured singular value calculation. The cost function of the controller can be formulated to account for manipulated variable movement weighting, manipulated variable constraints and a terminal condition. Finally, the proposed methodology is applied to a single-input-single-output continuous stirred tank reactor problem and to a multiple-input-multiple-output pH neutralization process.

[*]Adapted from Díaz-Mendoza, R. and Budman, H. Structured Singular Valued based robust nonlinear model predictive controller using Volterra series models, *Journal of Process Control*, 2010, 20, 5, 653–663. See Appendix C for further information

5.1 Introduction

Model predictive control (MPC) is currently one of the most used control strategies in the process industry. The MPC algorithm minimizes at each sampling instant a cost function with respect to the manipulated variables. The most common type of predictive controller applied in industry is based on linear models. On the other hand, current research is concentrating on nonlinear predictive control due to the reported advantages when controlling nonlinear processes (Findeisen and Allgöwer, 2002). Accordingly, the current chapter focuses on nonlinear model based predictive control.

Since the main objective of a predictive control strategy is to keep the value of the controlled variables close to its set-points, the cost function is generally based on a norm that penalizes the deviation of the predicted outputs from its predefined set-points. There are two types of models that can be used for output prediction: first-principles and empirical. First-principles models are based on the mass, energy and momentum balances of the process. Generally they are difficult to obtain and are complex for control analysis and design because they consist of high order nonlinear differential equations. On the other hand empirical models can be easily obtained from input-output data and, as shown in this chapter, they are often advantageous for control analysis and design. Both types of models have been used in previous nonlinear model predictive control (NMPC) algorithms. First-principles model based NMPC has been addressed by (Wright and Edgar, 1994), (Santos et al., 2001) and (Nygaard and Nævdal, 2006). There are many empirical models that have been used for designing a NMPC controller such as: polynomial autoregressive moving average models (Hérnandez and Arkun, 1993), Hammerstein models (Fruzzetti et al., 1997), Volterra models (Doyle III et al., 1995), (Maner et al., 1996) and (Maner and Doyle III, 1997), Volterra-Laguerre models (Parker and Doyle III, 2001) and Wiener models (Norquay et al., 1999), (Gerkšič et al., 2000) and (Jeong et al., 2001) to name a few. Since the accuracy of these predictions will affect the performance it is important to use a model that accurately describes the process behavior.

Considering that $\hat{y}$ is the predicted output and y^{sp} is the set-point, a vector accounting for the deviations of $\hat{y}$ from y^{sp} can be written as follows:

$$\mathbf{Y} = \begin{bmatrix} y_1^{\text{sp}}(k) - \hat{y}_1(k) \\ \vdots \\ y_1^{\text{sp}}(k+p) - \hat{y}_1(k+p) \\ \vdots \\ y_{n_y}^{\text{sp}}(k) - \hat{y}_{n_y}(k) \\ \vdots \\ y_{n_y}^{\text{sp}}(k+p) - \hat{y}_{n_y}(k+p) \end{bmatrix} \tag{5.1}$$

where p is the prediction horizon and n_y is the number of outputs. Without loss of generality, the objective function of the controller proposed in the current work minimizes the maximum absolute value of the elements of the $\mathbf{Y}$ vector with respect to the manipulated variables vector $\mathbf{U}$ according to the following formula

$$J = \min_{\text{wrt}\,\mathbf{U}} \|\mathbf{Y}\|_\infty \tag{5.2}$$

where the manipulated variables vector $\mathbf{U}$ is defined as

$$\mathbf{U} = \left[u_1(k), \ldots, u_1(k+m), \ldots, u_{n_u}(k), \ldots, u_{n_u}(k+m)\right]^{\text{T}} \tag{5.3}$$

m is the control horizon and n_u is the number of inputs. In principle, the methodology to be presented in this chapter can also be applied when a 2-norm is considered for the optimization problem in equation (5.2). However, the use of the 2-norm will lead to an increase of the dimensions of the algebraic formulation resulting in a corresponding increase in the computational effort. Therefore, for simplicity, a 2-norm has not been used in the current study.

The value of $\hat{y}$ is obtained from a model of the process. For any mathematical model there will always exist a discrepancy between the process output and the model output. If this discrepancy is ignored the resulting closed-loop performance may be poor and in extreme cases closed-loop instability may occur. Thus, it is imperative to consider the effect of mismatch between the model and the process by designing a controller that will be robust with respect to this mismatch. Although a significant amount of research has been conducted on robustness for MPC controllers that are based on linear models, the robustness of nonlinear model predictive controllers has been identified as an area of research that needs further investigation (Findeisen and Allgöwer, 2002). Some researchers have addressed the impact of model error on NMPC performance through simulation studies (Nagy and Braatz, 2003), (Kawohl et al., 2007), (Magni and Scattolini, 2007), (Diehl et al., 2008) and (Zavala and Biegler, 2009). Linear matrix inequality (LMI) based robustness tests have been developed for nonlinear systems that can be represented by specific empirical models (Peng et al., 2007). Thus, the key contribution of this chapter is the formulation of a nonlinear predictive control methodology based on empirical Volterra series model where robustness to model errors is addressed with a systematic theoretical approach.

The current chapter investigates the design of a robust NMPC based on an empirical Volterra series model. It is shown that the structure of the Volterra series model permits the formulation of the robust NMPC problem as a μ-structured singular value (SSV) test that can be used to calculate on-line the optimal control actions, i.e., the manipulated variables' vector $\mathbf{U}$. The μ-SSV (Doyle, 1982) based test is obtained, at each sampling instant, to bound the worst case predictions or constraint violations along predefined prediction and control horizons in the presence of disturbances and uncertainty in the parameters of the Volterra series model. To assess the merits of the proposed algorithm, two case studies involving a single-input-single-output (SISO) and multiple-input-multiple-output (MIMO) chemical processes were considered. In these case studies the proposed controller was compared with a non-robust linear MPC and with a non-robust NMPC that did not consider model error. This last comparison was conducted to demonstrate the need for considering model error within the controller design.

This chapter is organized as follows. In section 5.2 Volterra series models are reviewed. The SSV-based robustness test is presented in section 5.3. Section 5.3 also shows how to include, within the formulation, manipulated variable movement weighting, manipulated variables constraints and a terminal condition. Section 5.4 presents two case studies and conclusions are presented in section 5.5.

5.2 Volterra series models

Volterra series models have been shown to effectively capture nonlinear behavior (Schetzen, 1980) and (Parker et al., 2001). A specific advantage in the use of this type of model for this work is that it has a structure that can be readily split into a nominal part and another part that accounts for the uncertainty in the model parameters and accordingly, it can be used to formulate a mathematical robust performance test.

A Volterra series model relates the output of a process to its previous inputs. For a SISO process the relationship between the output and input can be written as follows:

$$\hat{y}(k) = \hat{y}_1(k) + \hat{y}_2(k) + \dots \tag{5.4}$$

where

$$\hat{y}_i(k) = \sum_{\sigma_1=1}^{\infty} \dots \sum_{\sigma_i=1}^{\infty} h_i(\sigma_1, \dots, \sigma_i)\, u(k-\sigma_1), \dots, u(k-\sigma_i) \tag{5.5}$$

In equation (5.5) the $h_i{}'s$ are the coefficients of the Volterra kernels. For practical purposes the series is truncated to include a finite number of terms resulting in the following equation:

$$\hat{y}(k) = h_0 + \sum_{\nu=1}^{N} \sum_{i_1=0}^{M-1} \dots \sum_{i_\nu=0}^{M-1} h_\nu(i_1, \dots, i_\nu)\, u(k-i_1) \dots u(k-i_\nu) \tag{5.6}$$

For example if $h_0 = 0$ and $N = 2$ it follows from equation (5.6):

$$\hat{y}(k) = \sum_{n=0}^{M-1} h_n u(k-n) + \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} h_{i,j} u(k-i)\, u(k-j) \tag{5.7}$$

where M is referred to as the memory of the system and it is generally chosen to correspond to the settling time of the process being modeled. The model can be generalized for a MIMO case as follows (Schetzen, 1980) and (Boyd and Chua, 1985):

$$\begin{aligned}\hat{y}_{\chi}(k) = &\left(\sum_{n=0}^{M-1} h_{n(\chi,1)} u_1(k-n) + \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} h_{i,j(\chi,1)} u_1(k-i)\, u_1(k-j)\right) + \ldots + \\ &\left(\sum_{n=0}^{M-1} h_{n(\chi,n_u)} u_1(k-n) + \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} h_{i,j(\chi,n_u)} u_{n_u}(k-i)\, u_{n_u}(k-j)\right)\end{aligned} \tag{5.8}$$

where $\chi = 1, \ldots, n_y$. The Volterra series coefficients can be obtained by least square regression or nonlinear optimization using process input-output data by imposing an appropriate input sequence. It has been shown (Nowak and Van Veen, 1994) that to identify the coefficients for a system with polynomial degree N, it is necessary to use a $N+1$ level pseudorandom multilevel sequence (PRMS). The coefficients can also be obtained by decomposing the model structure into a linear, diagonal and off-diagonal part (Parker et al., 2001) and (Soni and Parker, 2007). This decomposition allows imposing a plant friendly sequence that excites selected parts of the model simplifying the identification process. In this work the approach proposed in (Nowak and Van Veen, 1994) was used to identify the parameters.

In some cases a more compact Volterra model can be used by including autoregressive terms as per the following structure (Schetzen, 1980) and (Boyd and Chua, 1985):

$$\begin{aligned}\hat{y}_{\chi}(k) = &\sum_{q=1}^{n_{\mathrm{ARX}}} h_{q_{\chi}} \hat{y}_{\chi}(k-q) + \\ &\left(\sum_{n=0}^{M-1} h_{n(\chi,1)} u_1(k-n) + \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} h_{i,j(\chi,1)} u_1(k-i)\, u_1(k-j)\right) + \ldots + \\ &\left(\sum_{n=0}^{M-1} h_{n(\chi,n_u)} u_{n_u}(k-n) + \sum_{i=0}^{M-1}\sum_{j=i}^{M-1} h_{i,j(\chi,n_u)} u_{n_u}(k-i)\, u_{n_u}(k-j)\right)\end{aligned} \tag{5.9}$$

where n_{ARX} is the number of autoregressive terms. Because the number of coefficients that must be identified is significantly lower compared to its non autoregressive counterpart, autoregressive Volterra series models are simpler to use and have smaller sensitivity to noise in the data used for identification. Some early examples of NMPC with Volterra series models can be found in (Doyle III et al., 1995), (Maner et al., 1996) and (Maner and Doyle III, 1997). The formulation proposed in this work considers an autoregressive Volterra series model since it requires less parameters to identify and it results in an algebraic formulation of lower dimensions.

Regardless of the number of terms included in the model, the output of a Volterra series model will always be different from the actual process output. This difference is expected to occur due to model truncation errors and round-off errors. Thus, when using a Volterra series model for model-based control it is expected that performance will deteriorate due to the presence of this model error. The current chapter presents a method for assessing the effect of model error on closed-loop performance and for designing a controller that is robust with respect to this error.

5.3 Calculation of the worst case in the presence of model error

To account for the effect of modeling error, a model that is composed of a nominal part and an uncertainty description is used for model-based controller design. The combination of the model and its accompanying uncertainty description represents a large family of models that is used to represent the actual system to be controlled. A robust NMPC algorithm can then be designed based on the model that results in the worst closed-loop performance on the assumption that if the worst model in the set satisfies a specific performance index then all of the other models, within the family of models considered for robust design, will also satisfy that index. The analysis of the worst model that is directly associated with the worst value of the elements of the $\mathbf{Y}$ vector for the controller in equation (5.2) can be formulated in terms of a SSV norm (Braatz et al., 1994), (Ma and Braatz, 2001) and (Nagy and Braatz, 2003). Following the above, the first step for designing a robust NMPC consists in modifying equation (5.9) to include coefficient uncertainty as follows:

$$
\begin{aligned}
\hat{y}_{\chi}(k) = & \sum_{q=1}^{n_{\mathrm{ARX}}} h_{q_{\chi}} \hat{y}_{\chi}(k-q) + \\
& \left(\sum_{n=0}^{M-1} \left(h_{n(\chi,1)} \pm \delta h_{n(\chi,1)} \right) u_1(k-n) + \right. \\
& \left. \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} \left(h_{i,j(\chi,1)} \pm \delta h_{i,j(\chi,1)} \right) u_1(k-i)\, u_1(k-j) \right) + \ldots + \\
& \left(\sum_{n=0}^{M-1} \left(h_{n(\chi,n_u)} \pm \delta h_{n(\chi,n_u)} \right) u_{n_u}(k-n) + \right. \\
& \left. \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} \left(h_{i,j(\chi,n_u)} \pm \delta h_{i,j(\chi,n_u)} \right) u_{n_u}(k-i)\, u_{n_u}(k-j) \right) + \\
& ic_{\chi}(k) + w_{\chi}(k) + d_{\chi}(k)
\end{aligned}
\tag{5.10}
$$

where δh_n, $n = 0, \ldots, (M-1)$ and $\delta h_{i,j}$, $i = 0, \ldots, (M-1)$, $j = i, \ldots, (M-1)$ are the uncertainties in the coefficients of the linear and nonlinear Volterra series terms, respectively, $ic_{\chi}(k)$, $\chi = 1, \ldots, n_y$ is the effect of the initial conditions, w_{χ}, $\chi = 1, \ldots, n_y$ is a feedback term that considers the effect of unmeasured disturbances and $d_{\chi}(k)$, $\chi = 1, \ldots, n_y$ is the effect of measured disturbances. The feedback term $w\chi(k)$ is calculated according to the following equation

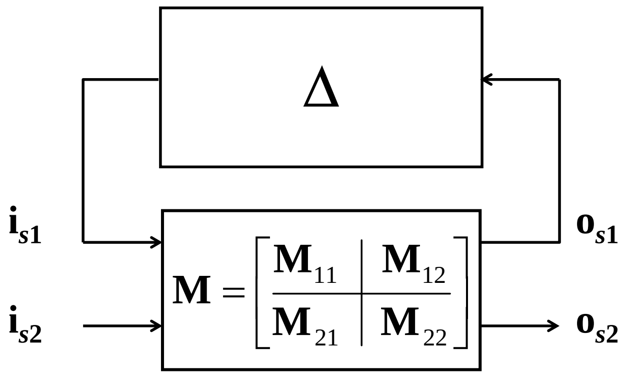
$$
w_{\chi}(k) = y_{\chi}^{\mathrm{real}}(k-1) - \hat{y}_{\chi}(k-1) \tag{5.11}
$$

For the purpose of prediction, the feedback term in equation (5.11) is assumed to be equal for all time intervals along the prediction horizon as is done on other MPC formulations (Cutler, 1983).

The formulation of a SSV-based robustness test requires two elements: (i) an appropriate interconnection matrix $\mathbf{M}$ that relates the nominal model with its uncertainty description and (ii) an uncertainty block structure $\boldsymbol{\Delta}$ accounting for the uncertainty in the model parameters. Appendix D provides details on how the prediction vector, comprised of elements calculated according to (5.10), can be represented by a corresponding $\mathbf{M} - \boldsymbol{\Delta}$ interconnection matrix. The interconnection is schematically shown in Figure 5.1. As shown in this figure, the inputs to $\mathbf{M}$ are $\mathbf{i}_{s1}$ and $\mathbf{i}_{s2}$ and the outputs are $\mathbf{o}_{s1}$ and $\mathbf{o}_{s2}$. If

$\mathbf{M}$ is partitioned into a structure that is compatible with the structure of the uncertainty matrix $\mathbf{\Delta}$, the equations describing the system of Figure 5.1 are:

Figure 5.1: Uncertainty description for an upper LFT



Δ
$\mathbf{i}_{s1}$
$\mathbf{i}_{s2}$
$\mathbf{o}_{s1}$
$\mathbf{o}_{s2}$
$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \hline \mathbf{M}_{21} & \mathbf{M}_{22} \end{array}\right]$

$$\begin{bmatrix} \mathbf{o}_{s1} \\ \mathbf{o}_{s2} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{i}_{s1} \\ \mathbf{i}_{s2} \end{bmatrix} \tag{5.12}$$

$$\mathbf{i}_{s1} = \mathbf{\Delta} \mathbf{o}_{s1} \tag{5.13}$$

The relationship between the exogenous signals $\mathbf{o}_{s2}$ and $\mathbf{i}_{s2}$ is given by

$$\mathbf{o}_{s2} = \left[\mathbf{M}_{21}\mathbf{\Delta}\left[\mathbf{I} - \mathbf{M}_{11}\mathbf{\Delta}\right]^{-1}\mathbf{M}_{12} + \mathbf{M}_{22}\right]\mathbf{i}_{s2} \tag{5.14}$$

where $\mathbf{o}_{s2}$ is related to $\mathbf{Y}$ and $\mathbf{i}_{s2}$ is related to $w_\chi(k)$ and $d_\chi(k)$. Details on the construction of this interconnection are given in Appendix D. Based on these interconnected matrices, the worst value of the elements of the $\mathbf{Y}$ vector, i.e., the worst $||\mathbf{Y}||_\infty$, can be calculated by the following SSV test (Braatz et al., 1994):

$$\max_{\text{wrt}\,\mathbf{H}^{\text{L}},\mathbf{H}^{\text{NL}}} \|\mathbf{Y}\|_\infty \geq k_{ssv} \Leftrightarrow \mu_{\mathbf{\Delta}}(\mathbf{M}) \geq k_{ssv} \tag{5.15}$$

where the maximization is carried out with respect to the uncertainty in the Volterra series coefficients defined by the following vectors:

$$\begin{aligned} \mathbf{H}^{\text{L}} &= \left[\mathbf{H}^{\text{L}}_1, \mathbf{H}^{\text{L}}_2, \ldots, \mathbf{H}^{\text{L}}_{n_y}\right] \\ \mathbf{H}^{\text{NL}} &= \left[\mathbf{H}^{\text{NL}}_1, \mathbf{H}^{\text{NL}}_2, \ldots, \mathbf{H}^{\text{NL}}_{n_y}\right] \end{aligned} \tag{5.16}$$

with

$$
\begin{aligned}
\mathbf{H}_i^{\mathrm{L}} &= \left[\delta h_{0\,(i,1)}, \ldots, \delta h_{0\,(i,n_u)}, \ldots, \delta h_{M-1\,(i,1)}, \ldots, h_{M-1\,(i,n_u)}\right] \\
\mathbf{H}_i^{\mathrm{NL}} &= \left[\delta h_{0,0\,(i,1)}, \ldots, \delta h_{0,0\,(i,n_u)}, \ldots, \delta h_{M-1,M-1\,(i,1)}, \ldots, h_{M-1,M-1\,(i,n_u)}\right]
\end{aligned}
\tag{5.17}
$$

The robustness test proposed in equation (5.15) is used to obtain a bound on the worst deviation of $\|\mathbf{Y}\|_\infty$ for the family of models defined by equations (5.10) and (5.16). The test in equation (5.15) can be reformulated by the following constrained optimization problem (Braatz et al., 1994):

$$
\max_{\mathrm{wrt}\,\mathbf{H}^{\mathrm{L}},\mathbf{H}^{\mathrm{NL}}} \|\mathbf{Y}\|_\infty = \max_{\substack{\mathrm{wrt}\,k_{ssv} \\ \mathrm{st}\,\mu_{\boldsymbol{\Delta}}(\mathbf{M})\geq k_{ssv}}} (k_{ssv}) \tag{5.18}
$$

The constrained optimization in equation (5.18), referred in the literature as a skew μ problem, has been shown to be convex (Braatz et al., 1994). It was mentioned in section 5.1 that the infinity norm was used instead of the 2-norm because the resulting dimensions of the interconnection matrix $\mathbf{M}$ are smaller. Since the calculation of the SSV is a non-polynomial hard problem the time required to solve equation (5.18) grows significantly with the dimensions of the interconnection matrix $\mathbf{M}$ and also, in extreme cases, memory limitations were encountered within the Matlab solving environment. These factors motivated the selection of the infinity norm for the present work.

As shown in Appendix D, $\mathbf{M}_{11}$ and $\mathbf{M}_{22}$ are zero matrices, $\mathbf{M}_{12}$ is a function of the scalar k_{ssv} and the elements of the $\mathbf{U}$ vector, i.e., $\mathbf{M}_{12} = (k_{ssv}, \mathbf{U})$ and $\mathbf{M}_{21}$ is a function of k_{ssv} and the elements of the $\mathbf{Y}$ vector, i.e., $\mathbf{M}_{21} = (k_{ssv}, \mathbf{Y})$.

The methodology used for finding a bound for the worst output along the prediction horizon by means of a skew μ problem can be extended to account for input and output constraints as follows. The key idea for the inclusion of constraints within the formulation is to append the values that should be kept within bounds to the prediction vector given by equation (5.1). Following this rationale it is possible to include: (i) constraints on manipulated variable changes $\Delta\mathbf{U}$ to prevent an excessive movement of the manipulated variables; (ii) constraints on the values of the manipulated variables u_{limits} to account for actuator limits and (iii) a terminal value condition tc to ensure convergence at steady-state.

This last constraint commonly referred to in the literature as a terminal condition (Chen and Allgöwer, 1998b), has been often used in previous NMPC algorithms for ensuring steady-state convergence to a neighborhood of the origin. The resulting vector with the appended variables related to manipulated variable changes' values, manipulated variables value constraints and to the steady-state output prediction can then be used within the following optimization problem:

$$\max_{\text{wrt}\,\mathbf{H}^{\mathrm{L}},\mathbf{H}^{\mathrm{NL}}} \left\| \begin{array}{c} \mathbf{Y} \\ \Delta\mathbf{U} \\ u_{\text{limits}} \\ tc \end{array} \right\|_{\infty} \geq k_{ssv} \Leftrightarrow \mu_{\mathbf{\Delta}}\left(\mathbf{M}\right) \geq k_{ssv} \tag{5.19}$$

Then, a skew μ problem is solved where a bound for the augmented vector of variables is solved as follows:

$$\max_{\text{wrt}\,\mathbf{H}^{\mathrm{L}},\mathbf{H}^{\mathrm{NL}}} \left\| \begin{array}{c} \mathbf{Y} \\ \Delta\mathbf{U} \\ u_{\text{limits}} \\ tc \end{array} \right\|_{\infty} = \max_{\substack{\text{wrt}\,k_{ssv} \\ \text{st}\,\mu_{\mathbf{\Delta}}(\mathbf{M})\geq k_{ssv}}} \left(k_{ssv}\right) \tag{5.20}$$

where the interconnection matrix $\mathbf{M}$ has a structure similar to equation (5.12), but with the difference that $\mathbf{M}_{21}$ is a function of k_{ssv} and the elements of the vectors $\mathbf{Y}$, $\Delta\mathbf{U}$, u_{limits} and tc i.e., $\mathbf{M}_{21} = (k_{\text{ssv}}, \mathbf{Y}, \Delta\mathbf{U}, u_{\text{limits}}, tc)$. The following subsections provide further details on how to formulate the: $\Delta\mathbf{U}$, u_{limits} and tc terms appearing in equation (5.20).

5.3.1 Manipulated variables movement penalization

This term is usually added to tune the speed of the closed-loop response and to prevent excessive wear of the actuators. To include the penalization of manipulated variable moves a vector referred to as $\Delta\mathbf{U}$ is defined as follows:

$$\Delta\mathbf{U} = \begin{bmatrix} W_1^{\Delta u_1}\left[u_1\left(k\right) - u_1\left(k-1\right)\right] \\ \vdots \\ W_m^{\Delta u_1}\left[u_1\left(k+m\right) - u_1\left(k+m-1\right)\right] \\ \vdots \\ W_1^{\Delta u_{n_u}}\left[u_{n_u}\left(k\right) - u_{n_u}\left(k-1\right)\right] \\ \vdots \\ W_m^{\Delta u_{n_u}}\left[u_{n_u}\left(k+m\right) - u_{n_u}\left(k+m-1\right)\right] \end{bmatrix} \tag{5.21}$$

where W_i^j, $i = 1, \ldots, m$, $j = \Delta u_1, \ldots, \Delta u_{n_u}$ is the weight associated to the movement of the j input from sampling time $(k+i-1)$ to $(k+i)$. Since this vector is bounded as per the problem in equation (5.20) then it follows that for all time intervals along the control horizon the manipulated variable moves are bounded as follows:

$$\begin{gathered} \max\left|W_1^{\Delta u_1}\left[u_1\left(k\right) - u_1\left(k-1\right)\right]\right| \leq k_{ssv} \\ \vdots \\ \max\left|W_m^{\Delta u_1}\left[u_1\left(k+m\right) - u_1\left(k+m-1\right)\right]\right| \leq k_{ssv} \\ \vdots \\ \max\left|W_1^{\Delta u_{n_u}}\left[u_{n_u}\left(k\right) - u_{n_u}\left(k-1\right)\right]\right| \leq k_{ssv} \\ \vdots \\ \max\left|W_m^{\Delta u_{n_u}}\left[u_{n_u}\left(k+m\right) - u_{n_u}\left(k+m-1\right)\right]\right| \leq k_{ssv} \end{gathered} \tag{5.22}$$

The weight values assigned to each manipulated variable move can be chosen to penalize the moves to different degrees in a similar fashion as the suppression factor, commonly used in conventional MPC control.

5.3.2 Manipulated variables constraints

Constraints in the manipulated variables arise due to actuator limits. To account for these limits a vector u_{limits} is defined as follows:

$$u_{\text{limits}} = \begin{bmatrix} k_{ssv}\frac{u_1(k)}{u_1^{\text{bound}}(k)} \\ \vdots \\ k_{ssv}\frac{u_1(k+m)}{u_1^{\text{bound}}(k+m)} \\ \vdots \\ k_{ssv}\frac{u_{n_u}(k)}{u_{n_u}^{\text{bound}}(k)} \\ \vdots \\ k_{ssv}\frac{u_{n_u}(k+m)}{u_{n_u}^{\text{bound}}(k+m)} \end{bmatrix} \tag{5.23}$$

The problem in equation (5.20) ensures that all of the elements of the vector in equation (5.23) are bounded as follows:

$$\begin{gathered} \max\left|k_{ssv}\frac{u_1(k)}{u_1^{\text{bound}}(k)}\right| \leq k_{ssv} \\ \vdots \\ \max\left|k_{ssv}\frac{u_1(k+m)}{u_1^{\text{bound}}(k+m)}\right| \leq k_{ssv} \\ \vdots \\ \max\left|k_{ssv}\frac{u_{n_u}(k)}{u_{n_u}^{\text{bound}}(k)}\right| \leq k_{ssv} \\ \vdots \\ \max\left|k_{ssv}\frac{u_{n_u}(k+m)}{u_{n_u}^{\text{bound}}(k+m)}\right| \leq k_{ssv} \end{gathered} \tag{5.24}$$

Then, after cancellation of k_{ssv} from both sides of the inequalities it trivially follows from equation (5.24):

$$\begin{gathered} \max\left|u_1(k)\right| \leq u_1^{\text{bound}}(k) \\ \vdots \\ \max\left|u_1(k+m)\right| \leq u_1^{\text{bound}}(k+m) \\ \vdots \\ \max\left|u_{n_u}(k)\right| \leq u_{n_u}^{\text{bound}}(k) \\ \vdots \\ \max\left|u_{n_u}(k+m)\right| \leq u_{n_u}^{\text{bound}}(k+m) \end{gathered} \tag{5.25}$$

Thus, the manipulated variables are bounded at each sampling instant by $u_i^{\text{bound}}(j)$, $i = 1, \ldots, n_u$, $j = k, \ldots, (k+m)$.

5.3.3 Terminal condition

A terminal condition is used to ensure that the difference between the set-point and the predicted steady-state output value corresponding to the last calculated manipulated variable move in the control horizon stays within a neighbourhood ϵ near the origin (Chen and Allgöwer, 1998b). Infeasibilities may arise when output constraints have to be satisfied in the presence of input constraints. In the current study, output constraints are only imposed at steady state through the terminal condition. Thus, the value of ϵ is pre-specified by the user with the only requirement that the region defined by ϵ can be reached at steady state considering the imposed limits on the manipulated variables. To enforce this condition within the maximization problem defined by equation (5.20) it is necessary to define a vector tc as follows:

$$tc = \begin{bmatrix} tc_1 \\ \vdots \\ tc_\chi \end{bmatrix} \tag{5.26}$$

$$tc_\chi = \frac{k_{ssv}}{\epsilon} \left(y_\chi^{\mathrm{sp}} (k+p) - \hat{y}_\chi (k+p) \right) \tag{5.27}$$

where $\hat{y}_\chi (k+p)$ is calculated from equation (5.10). If the cost function considers the vector tc within the cost function given in the left hand side of equation (5.20) the bound to be found by the solution of the corresponding skew μ problem will trivially ensure:

$$\begin{gathered} \max |tc_1| \leq k_{ssv} \\ \vdots \\ \max \left|tc_{n_\chi}\right| \leq k_{ssv} \end{gathered} \tag{5.28}$$

which after canceling out k_{ssv} from both sides of equation (5.28) can be simplified following equation (5.27) to the following inequalities:

$$\begin{gathered} \max \left|y_1^{\mathrm{sp}} (k+p) - \hat{y}_1 (k+p)\right| \leq \epsilon \\ \vdots \\ \max \left|y_\chi^{\mathrm{sp}} (k+p) - \hat{y}_\chi (k+p)\right| \leq \epsilon \end{gathered} \tag{5.29}$$

Thus the difference between the set-point and the predicted output at the end of the prediction horizon is bounded by ϵ.

The procedure to obtain the interconnection matrix $\mathbf{M}$ corresponding to the problem defined by equation (5.20) and the associated uncertainty block structure $\boldsymbol{\Delta}$ is detailed in Appendix D.

5.3.4 Robust NMPC control law

The robust NMPC control law is based on the evaluation of the cost function for the worst case that results when uncertainty is considered in the Volterra series coefficients. This can be obtained by solving the SSV-based robustness test proposed in equation (5.20). Then, the objective of the proposed controller is to minimize this worst-case cost function value with respect to the manipulated variables vector $\mathbf{U}$ at each time interval. Accordingly, the control calculation at each sampling instant consists of the solution of the following problem:

$$\min_{\text{wrt}\,\mathbf{U}} \left(\max_{\text{wrt}\,\mathbf{H}^{\mathrm{L}}\mathbf{H}^{\mathrm{NL}}} \left\| \begin{matrix} \mathbf{Y} \\ \Delta\mathbf{U} \\ u_{\text{limits}} \\ tc \end{matrix} \right\|_{\infty} \right) = \min_{\text{wrt}\,\mathbf{U}} \left(\max_{\substack{\text{wrt}\,k_{ssv} \\ \text{st}\,\mu_{\boldsymbol{\Delta}}(\mathbf{M}) \geq k_{ssv}}} (k_{ssv}) \right) \tag{5.30}$$

where, as per the previous subsections, the resulting controller has the following properties: (i) it considers uncertainty in the Volterra series coefficients through the vector $\mathbf{Y}$; (ii) it considers manipulated variable movement weighting through the term $\Delta\mathbf{U}$; (iii) it considers manipulated variables constraints through u_{limits} and (iv) it considers the effect of a terminal condition through tc. The inner maximization in equation (5.30) accounts for all the constraints, i.e., manipulated variable, terminal condition and μ inequality. This inner maximization is solved by proposing increasing values of the bound k_{ssv} until the equality $\mu = k_{ssv}$ is satisfied. The outer minimization is not subject to constraints and therefore, is solved with an unconstrained algorithm based on the Simplex method. When the inner maximization is unfeasible the outer minimization is restarted with a new set of initial guesses.

One disadvantage of the controller formulation given by equation (5.30) is that it does not have integral action in the presence of model error. The reason for the lack of integral action is as follows. Defining $\hat{y}_{\Delta}$ as the worst predicted output obtained from a series of models the elements of the feedback vector $\mathbf{E}$ are:

$$\mathbf{E} = \begin{bmatrix} y_1^{\text{sp}}(k) - \hat{y}_{\Delta 1}(k) \\ \vdots \\ y_1^{\text{sp}}(k+p) - \hat{y}_{\Delta 1}(k+p) \\ \vdots \\ y_{n_y}^{\text{sp}}(k) - \hat{y}_{\Delta n_y}(k) \\ \vdots \\ y_{n_y}^{\text{sp}}(k+p) - \hat{y}_{\Delta n_y}(k+p) \end{bmatrix} \tag{5.31}$$

That after correction with the feedback term in equation (5.11) results in the following prediction vector:

$$\mathbf{E}' = \begin{bmatrix} y_1^{\text{sp}}(k) - \hat{y}_{\Delta 1}(k) - y_1^{\text{real}}(k-1) + \hat{y}_1(k-1) \\ \vdots \\ y_1^{\text{sp}}(k+p) - \hat{y}_{\Delta 1}(k+p) - y_1^{\text{real}}(k-1) + \hat{y}_1(k-1) \\ \vdots \\ y_{n_y}^{\text{sp}}(k) - \hat{y}_{\Delta n_y}(k) - y_{n_y}^{\text{real}}(k-1) + \hat{y}_{n_y}(k-1) \\ \vdots \\ y_{n_\chi}^{\text{sp}}(k+p) - \hat{y}_{\Delta n_\chi}(k+p) - y_{n_y}^{\text{real}}(k-1) + \hat{y}_{n_y}(k-1) \end{bmatrix} \tag{5.32}$$

The best case that can be obtained from the minimization in equation (5.30) and in the absence of constraints is that each one of the elements in $\mathbf{E}'$ will be zero. In that case, since generally $\hat{y}_{\Delta i} \neq \hat{y}_i$ according to equation (5.32), $y_i^{\text{real}} \neq y_i^{\text{sp}}$ thus resulting in offset.

To correct for this situation a correction to the controller is proposed whereby when $||\mathbf{E}'||_\infty < \epsilon'$ then the prediction for the calculation of the worst case is done without considering the uncertainty. In that case $\hat{y}_{\Delta i} = \hat{y}_i$ and consequently $y_i^{\text{real}} = y_i^{\text{sp}}$ resulting in zero offset. This proposed correction results in a two-mode controller where, for $||\mathbf{E}'||_\infty \geq \epsilon'$ the Volterra model with uncertainty is used to calculate the worst case whereas for $||\mathbf{E}'||_\infty < \epsilon'$ the Volterra model without uncertainty is used for that calculation.

5.4 Case studies

Two case studies are presented to illustrate the properties of the proposed algorithm. The first case study is a SISO exothermic reactor and the second is a MIMO pH neutralization process. For both case studies comparisons are made between the proposed robust nonlinear controller with a non-robust linear model based MPC (LMPC).

5.4.1 SISO system

A first order, exothermic, irreversible reaction A $\rightarrow$ B is taking place in a continuously stirred tank reactor (CSTR). Assuming perfect mixing the dimensionless material and energy balances are as follows (Uppal et al., 1974) and (Doyle III et al., 1989):

$$\frac{\mathrm{d}x_1}{\mathrm{d}t} = x_1 + Da\,(1 - x_1)\exp\frac{x_2}{1 + x_2/\gamma} \tag{5.33}$$

$$\frac{\mathrm{d}x_2}{\mathrm{d}t} = -x_2 + BDa\,(1 - x_1)\exp\left(\frac{x_2}{1 + x_2/\gamma}\right) - \beta\,(x_2 - x_{2c}) \tag{5.34}$$

where B is the dimensionless heat of reaction, Da Damökhler number, x_1 dimensionless reactant concentration, x_2 dimensionless temperature, x_{2c} dimensionless coolant temperature, β dimensionless cooling rate and γ dimensionless activation energy. The control problem consists in maintaining x_1 at its predefined set-point by manipulating the value of x_{2c}, in the presence of perturbations in the value of β. A change in the value of β may be related to changes in temperature or flow rate. The following parameters are chosen $B = 1.0$, $Da = 0.072$, $\beta = 0.3$, $\gamma = 20.0$ and $x_{2c} = 14$. For these values one single steady-state is obtained at $x_1^{\text{ss}} = 0.622$ and $x_2^{\text{ss}} = 3.7092$.

An autoregressive Volterra model was identified for prediction within the NMPC strategy. It was found by trial and error that a choice of $M = 3$ and $n_{\mathrm{ARX}} = 1$ ensures that the autoregressive Volterra series model gives a reasonable fitting with a maximal error of 10% while at the same time keeping the interconnection matrix small. Selecting $M > 3$ and $n_{\mathrm{ARX}} > 1$ resulted in slightly better fitting but also led to increases in the dimensions of the $\mathbf{M}$ matrix and a corresponding increase in the on-line calculations of the SSV norm. To obtain the nominal and uncertain value of the auto-regressive Volterra series coefficients 10 different PRMS were applied to the system. The sampling time for identification and simulation purposes was 0.75 time units. For each PRMS a set of Volterra series coefficients were obtained by nonlinear optimization. The set of coefficients was divided into two subsets corresponding to the coefficients of the linear and nonlinear terms. Accordingly, $\mathbf{VC}$ defines the set of Volterra series kernels as follows:

$$\mathbf{VC} = \left[\mathbf{VC}_1^{\mathrm{L}}, \ldots, \mathbf{VC}_{n_y}^{\mathrm{L}}, \mathbf{VC}_1^{\mathrm{NL}}, \ldots, \mathbf{VC}_{n_y}^{\mathrm{NL}}\right] \tag{5.35}$$

$$\begin{aligned} \mathbf{VC}_i^{\mathrm{L}} &= \left[h_{0\,(i,1)}, \ldots, h_{0\,(i,n_u)}, \ldots, h_{M-1\,(i,1)}, \ldots, h_{M-1\,(i,n_u)}\right] \\ \mathbf{VC}_i^{\mathrm{NL}} &= \left[h_{0,0\,(i,1)}, \ldots, h_{0,0\,(i,n_u)}, \ldots, h_{M-1,M-1\,(i,1)}, \ldots, h_{M-1,M-1\,(i,n_u)}\right] \end{aligned} \tag{5.36}$$

for the j-th PRMS the identified Volterra series coefficients are:

$$\begin{aligned} \mathbf{VCid}^{\mathrm{L}} &= \left[\mathbf{VCid}_1^{\mathrm{L}}, \ldots, \mathbf{VCid}_{n_{seq}}^{\mathrm{L}}\right] \\ \mathbf{VCid}^{\mathrm{NL}} &= \left[\mathbf{VCid}_1^{\mathrm{NL}}, \ldots, \mathbf{VCid}_{n_{seq}}^{\mathrm{NL}}\right] \end{aligned} \tag{5.37}$$

$$\begin{aligned} \mathbf{VCid}_j^{\mathrm{L}} &= \left[\mathbf{VCid}_{j,1}^{\mathrm{L}}, \ldots, \mathbf{VCid}_{j,n_y}^{\mathrm{L}}\right] \\ \mathbf{VCid}_j^{\mathrm{NL}} &= \left[\mathbf{VCid}_{j,1}^{\mathrm{NL}}, \ldots, \mathbf{VCid}_{j,n_y}^{\mathrm{NL}}\right] \end{aligned} \tag{5.38}$$

$$\begin{aligned} \mathbf{VCid}_{j,i}^{\mathrm{L}} &= \left[h_{0\,(i,1),j}, \ldots, h_{0\,(i,n_u),j}, \ldots, h_{M-1\,(i,1),j}, \ldots, h_{M-1\,(i,n_u),j}\right] \\ \mathbf{VCid}_{j,i}^{\mathrm{NL}} &= \left[h_{0,0\,(i,1),j}, \ldots, h_{0,0\,(i,n_u),j}, \ldots, h_{M-1,M-1\,(i,1),j}, \ldots, h_{M-1,M-1\,(i,n_u),j}\right] \end{aligned} \tag{5.39}$$

The nominal value of the coefficients that were used for the construction of the interconnection matrix $\mathbf{M}$ were calculated as follows:

$$
\begin{aligned}
\mathbf{VC}^{\mathrm{L}} &= \sum_{j=1}^{n_{seq}} \frac{\mathbf{VCid}_j^{\mathrm{L}}}{n_{seq}} \\
\mathbf{VC}^{\mathrm{NL}} &= \sum_{j=1}^{n_{seq}} \frac{\mathbf{VCid}_j^{\mathrm{NL}}}{n_{seq}}
\end{aligned}
\tag{5.40}
$$

where n_{seq} is the number of PRMS sequences used. The elements of the vectors $\mathbf{VC}^{\mathrm{L}}$ and $\mathbf{VC}^{\mathrm{NL}}$ are equal to the means of $[\mathbf{VCid}_1^{\mathrm{L}}, \ldots, \mathbf{VCid}_{n_{seq}}^{\mathrm{L}}]$ and $[\mathbf{VCid}_1^{\mathrm{NL}}, \ldots, \mathbf{VCid}_{n_{seq}}^{\mathrm{NL}}]$, respectively.

Following the central limit theorem a 95% probability that the real value of the Volterra series coefficients is contained within two standard deviation of the mean was assumed. Therefore, the associated uncertainty matrix used for the construction of the interconnection matrix $\mathbf{M}$ is calculated as:

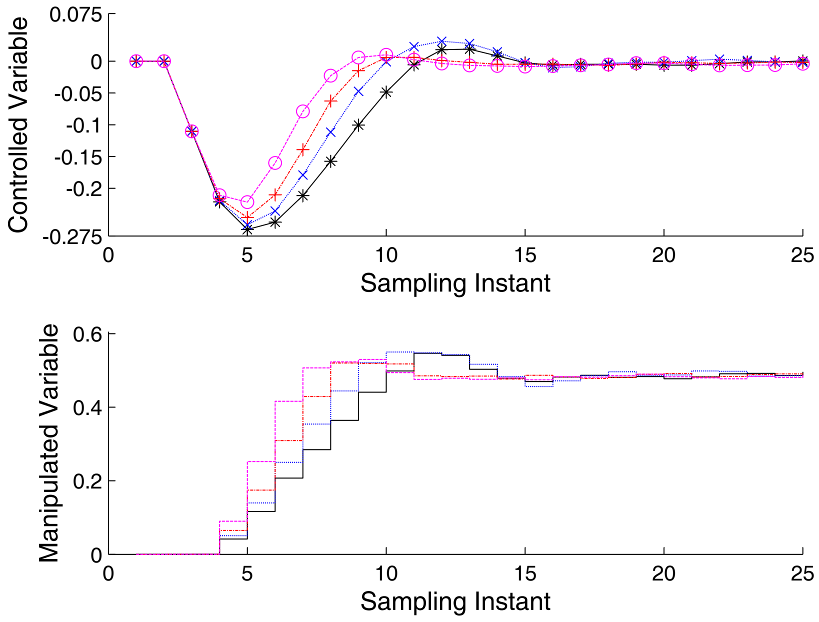
$$
\begin{aligned}
\mathbf{H}^{\mathrm{L}} &= 2\sqrt{\frac{1}{n_{seq}} \sum_{j=1}^{n_{seq}} \left(\mathbf{VCid}_j^{\mathrm{L}} - \mathbf{VC}^{\mathrm{L}}\right)^2} \\
\mathbf{H}^{\mathrm{NL}} &= 2\sqrt{\frac{1}{n_{seq}} \sum_{j=1}^{n_{seq}} \left(\mathbf{VCid}_j^{\mathrm{NL}} - \mathbf{VC}^{\mathrm{NL}}\right)^2}
\end{aligned}
\tag{5.41}
$$

SISO case study A

Figure 5.2 shows different closed-loop controlled and manipulated variables profiles to a step like disturbance that consisted of a change in the value of β from $\beta = 0.3$ to $\beta = 0.21$ at time $k = 2$. For these simulations the set-point is equal to zero, $p = 10$ and $m = 2$. To illustrate the effect of the weight $W_i^{\Delta u}$, $i = 1, \ldots, m$, as defined in equation (5.21) i.e., the weight that penalizes the manipulated variable changes, closed-loop simulations were conducted with the proposed controller for different values of this weight. It can be seen from Figure 5.2 that the weight imposed on the manipulated variable movements can be effectively used to tune the speed of the closed-loop response. Also, the simulations show that the controlled variable always converges to the set-point thus illustrating the presence of integral action achieved by the two mode control strategy proposed at the end of section

5.3, i.e., the model without the uncertainty is used for prediction of the worst deviation as soon as the error is smaller than ϵ'.

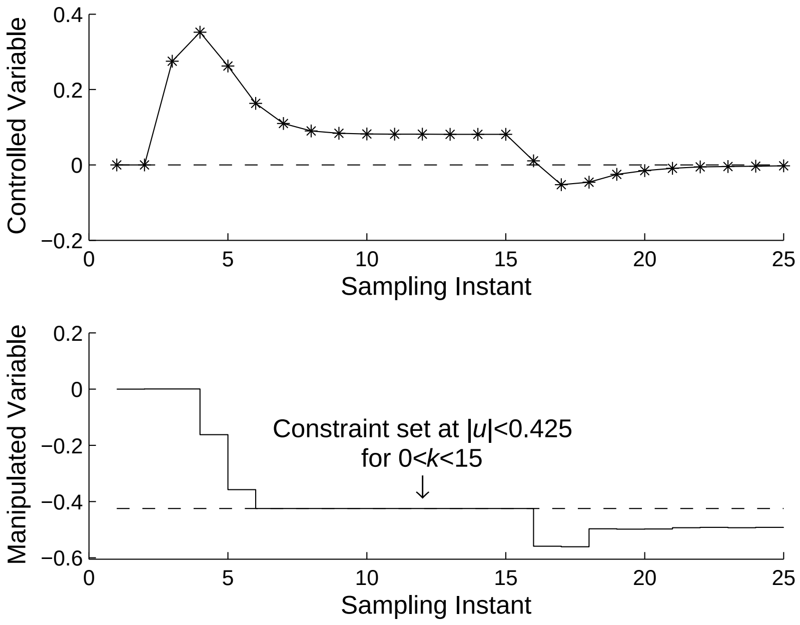
Figure 5.2: Controlled and manipulated variables for SISO case study A, (black $W_i^{\Delta u} = 2.5$), (blue $W_i^{\Delta u} = 2.0$), (red $W_i^{\Delta u} = 1.5$), (magenta $W_i^{\Delta u} = 1.0$)



SISO case study B

To test the ability of the algorithm to accommodate constraints a step-like disturbance in β consisting of a change from $\beta = 0.3$ to $\beta = 0.525$ at time $k = 2$ was simulated with the proposed controller. Figure 5.3 shows the responses of the controlled and manipulated variable when $u(k)$ is restricted to be between $|u(k)| \leq 0.425$ for $0 < k < 15$. It can be seen that the proposed NMPC algorithm is able to keep the value of the manipulated variable between the allowed limits. Figure 5.3 also shows that after removing the constraint at time interval 15 the proposed algorithm calculates the manipulated variable value that forces the controlled variable to the set-point.

Figure 5.3: Controlled and manipulated variables for SISO case study B, $W_i^{\Delta u} = 1.5$



SISO case study C

In order to assess the efficiency of the proposed controller its average performance was quantified for a set of different disturbances. A disturbance profile consisting of the superposition of random changes and step changes shown in Figure 5.4 was used. The characteristics of the disturbance can be found in Appendix E. The rationale for choosing this type of disturbance profile was to test the closed-loop performance around different operating conditions so as to emphasize the effects of nonlinearity. For this disturbance the proposed robust NMPC was compared to a non-robust NMPC for which $\mathbf{H}^{\mathrm{L}} = 0$ and $\mathbf{H}^{\mathrm{NL}} = 0$ and to a non-robust linear MPC for which $\mathbf{H}^{\mathrm{L}} = 0$, $\mathbf{H}^{\mathrm{NL}} = 0$ and all nonlinear terms in equation (5.9) are equal to zero. The comparison of the controllers was performed for two different values of manipulated variables moves weight $W_i^{\Delta u}$. The integral of the absolute error (IAE) values obtained from these simulations are presented in Table 5.1. Figure 5.5 shows the controlled and manipulated variable profiles, respectively. These results clearly corroborate that the nonlinear controllers perform better than the linear controller and more importantly, they indicate that the robust NMPC performs better than the non-robust NMPC.

Figure 5.4: Disturbance for SISO case study C

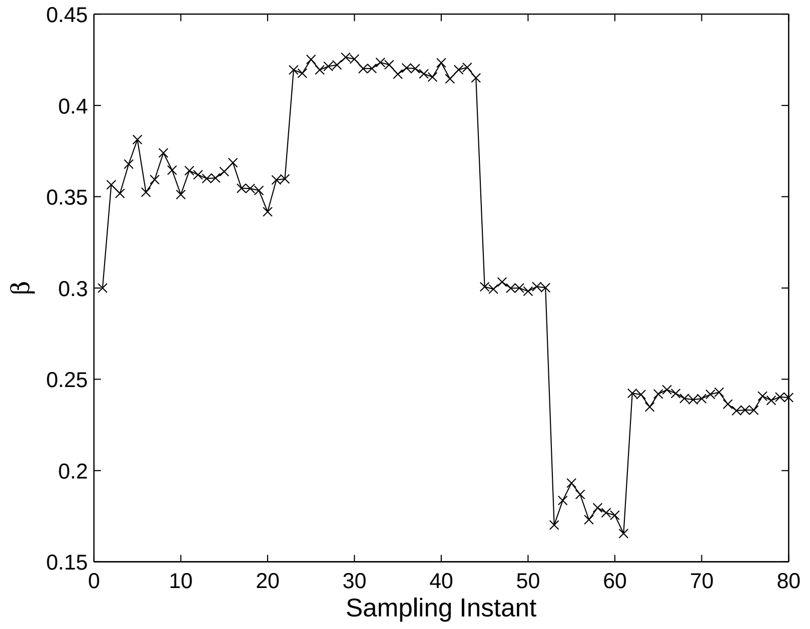


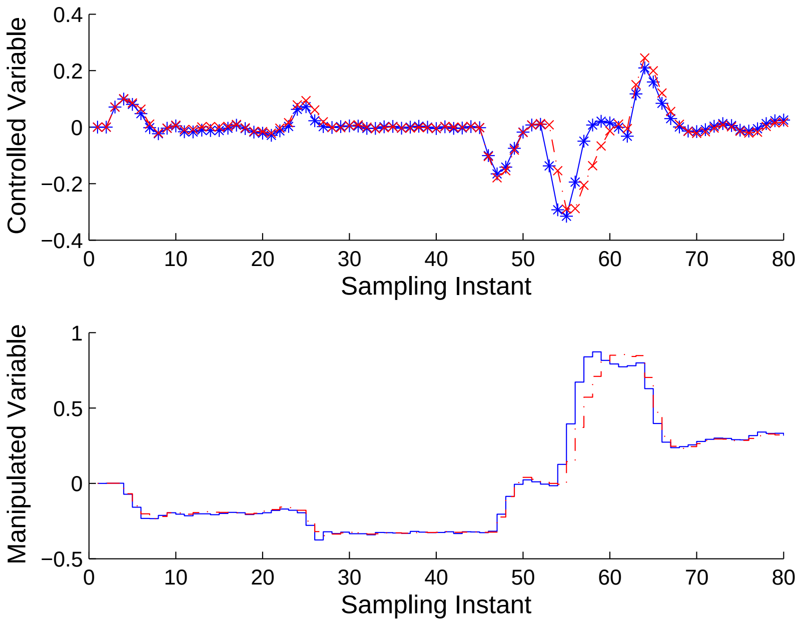
Table 5.1: IAE comparisons between robust NMPC, non-robust NMPC and non-robust LMPC for SISO case study C

$W_i^{\Delta u}$	$\mathrm{IAE}_{\mathrm{robust\,NMPC}}$	$\mathrm{IAE}_{\mathrm{non-robust\,NMPC}}$	$\mathrm{IAE}_{\mathrm{non-robust\,LMPC}}$
0.50	2.6249	2.9165	3.0104
0.75	3.0444	3.4696	3.5435

SISO case study D

To further illustrate the effect of accounting for robustness on the control performance the proposed robust NMPC was compared to its non-robust NMPC counterpart ($\mathbf{H}^{\mathrm{L}} = 0$ and $\mathbf{H}^{\mathrm{NL}} = 0$) for different possible disturbances and for different values of the controller weight $W_i^{\Delta u}$, 24 different step-like disturbances were used to simulate the system with both controllers. The characteristics of the disturbance can be found in Appendix E. For each one of these disturbances the IAE was calculated for the robust and non-robust NMPC

Figure 5.5: Controlled and manipulated variables for SISO case study C, $p = 10$, $m = 2$, $W_i^{\Delta u} = 0.75$



controllers. Table 5.2 shows the percentage of cases were the robust NMPC controller IAE was smaller than that of the non-robust NMPC for different values of $W_i^{\Delta u}$. The interpretation of this result is that as the controller weight is smaller the controller is more aggressive causing the closed-loop to be more sensitive to model error. Thus, for smaller weights the results showed as expected that the robust controller, that takes into account the model error, performs overall better than the non-robust controller. Thus, although robust controllers tend to be conservative, they perform better than the non-robust controller when the controller is tuned aggressively, i.e., with small manipulated variables' weights.

Table 5.2: IAE comparisons between robust NMPC, non-robust NMPC and non-robust LMPC for SISO case study D

$W_i^{\Delta u}$	% of cases with $\mathrm{IAE}_{\text{robust NMPC}} < \mathrm{IAE}_{\text{non-robust NMPC}}$
0.25	66.66%
0.50	66.66%
0.75	50.00%
1.00	45.83%
1.50	41.66%
2.00	25.00%
2.50	20.83%

5.4.2 MIMO system

This case study involves the pH neutralization system (Nahas et al., 1992) schematically shown in Figure 5.6 where an acid (q_1), a base (q_3) and a buffer (q_2) are mixed in a tank.

Assuming perfect mixing, constant density and completely solubility of the ions involved, the chemical equilibrium can be modeled according to the following two reaction invariants for each stream $i = 1, \ldots, 4$:

$$W_{ai} = \left[\mathrm{H}^+\right]_i - \left[\mathrm{OH}^-\right]_i - \left[\mathrm{HCO}_3^-\right]_i - 2\left[\mathrm{CO}_3^{2-}\right]_i \tag{5.42}$$

$$W_{bi} = \left[\mathrm{H_2CO_3}\right]_i + \left[\mathrm{HCO}_3^-\right]_i + \left[\mathrm{CO}_3^{2-}\right]_i \tag{5.43}$$

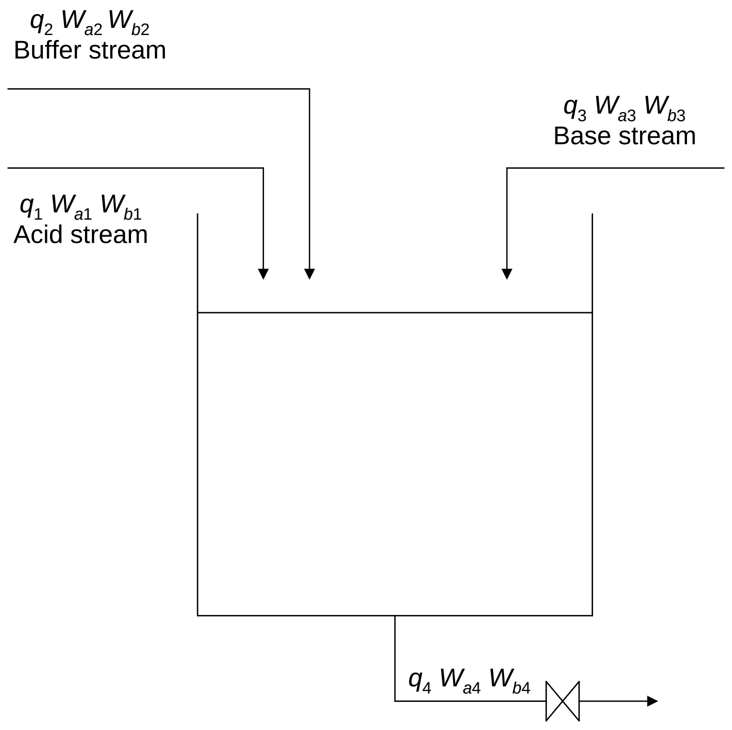
The overall mass balance on the tank yields:

$$A\frac{\mathrm{d}h}{\mathrm{d}t} = q_1 + q_2 + q_3 - q_4 \tag{5.44}$$

where q_4 is calculated as:

$$q_4 = Cv\,(h)^n \tag{5.45}$$

Figure 5.6: pH neutralization system



The equations that describe the dynamics of the effluent reaction invariants (W_{a4}, W_{b4}) can be obtained from a mass balance on each of the ionic species as follows:

$$Ah\frac{\mathrm{d}W_{a4}}{\mathrm{d}t} = q_1\left(W_{a1} - W_{a4}\right) + q_2\left(W_{a2} - W_{a4}\right) + q_3\left(W_{a3} - W_{a4}\right) \tag{5.46}$$

$$Ah\frac{\mathrm{d}W_{b4}}{\mathrm{d}t} = q_1\left(W_{b1} - W_{b4}\right) + q_2\left(W_{b2} - W_{b4}\right) + q_3\left(W_{b3} - W_{b4}\right) \tag{5.47}$$

Finally, the pH can be obtained from W_a and W_b by using the following nonlinear relationship:

$$W_a + 10^{\mathrm{pH}-14} + W_b\frac{1 + 2\times 10^{\mathrm{pH}-pK_2}}{1 + 10^{pK_1-\mathrm{pH}} + 10^{\mathrm{pH}-pK_2}} - 10^{-\mathrm{pH}} = 0 \tag{5.48}$$

The control problem consists in maintaining the tank height h and the outlet stream pH at its predefined set-point by manipulating the flows of the acid q_1 and base q_3 streams, in the presence of perturbations in the flow of the buffer stream q_2. The value of the set-points and the operating conditions for the process are shown on Table 5.3.

Table 5.3: Process set-points and operating conditions for MIMO case study

h^{sp}	$=$	14cm	q_1	$=$	3×10^{-3}M HNO$_3$
pH$^{\text{sp}}$	$=$	7	q_2	$=$	3×10^{-2}M NaHCO$_3$
A	$=$	207 cm^2	q_3	$=$	3×10^{-3}M NaOH + 5×10^{-5}M NaHCO$_3$
Cv	$=$	8.75 ml / cm s	W_{a1}	$=$	3×10^{-3}
n	$=$	0.607	W_{a2}	$=$	-3×10^{-2}
pK_1	$=$	6.35	W_{a3}	$=$	-3.05×10^{-2}
pK_2	$=$	10.25	W_{v1}	$=$	0
q_1	$=$	16.6 ml/s	W_{b2}	$=$	3×10^{-2}
q_2	$=$	0.55 ml/s	W_{b3}	$=$	5×10^{-5}
q_3	$=$	15.6 ml/s			

Similar to the SISO case study it was found by trial and error that the choices of $M = 3$ and $n_{\text{ARX}} = 1$ for the Volterra model resulted in a reasonable fitting between the process and the model. The sampling time used for identification and simulation purposes was 25s. To prevent numerical difficulties during the identification process the manipulated and control variables were normalized. The procedure to obtain the Volterra series coefficients was the same one used for the SISO case study. The main difference with respect to the first case study is that in these case two Volterra series models are required: one for the level given by $h = f(q1, q3)$ and one for the pH given by pH $= f(q1, q3)$.

Figure 5.7 shows the closed-loop controlled variables' responses to a step like disturbance that consisted in a change in the value of q_2 from $q_2 = 0.3$ ml/s to $q_2 = 0.525$ ml/s at time $k = 2$ with the proposed robust NMPC algorithm. For these simulations the set-points are equal to $h^{\text{sp}} = 14$ cm and pH$^{\text{sp}} = 7$ or $y_1^{\text{sp}} = 0$ and $y_2^{\text{sp}} = 0$ in deviation variables with respect to the initial steady-state. The manipulated variables movement penalization terms are equal to $W_i^{\Delta u_1} = 0.75$, and $W_i^{\Delta u_2} = 1.5$, $i = 1, \ldots, m$. Figure 5.8 shows the responses of u_1 and u_2, respectively. It can be seen that the controller is able to reject the disturbance in the process converging to the set-points at steady-state.

As done for the SISO system example, simulations were conducted to show under which circumstances the robust NMPC can surpass the performance of the non-robust NMPC

Figure 5.7: Controlled variables profiles for MIMO case study, $p = 10$, $m = 2$, $W_i^{\Delta u_1} = 0.75$, $W_i^{\Delta u_2} = 1.5$, (black = robust NMPC), (blue = non-robust NMPC), (red = non-robust LMPC) (The information is in deviation variables)

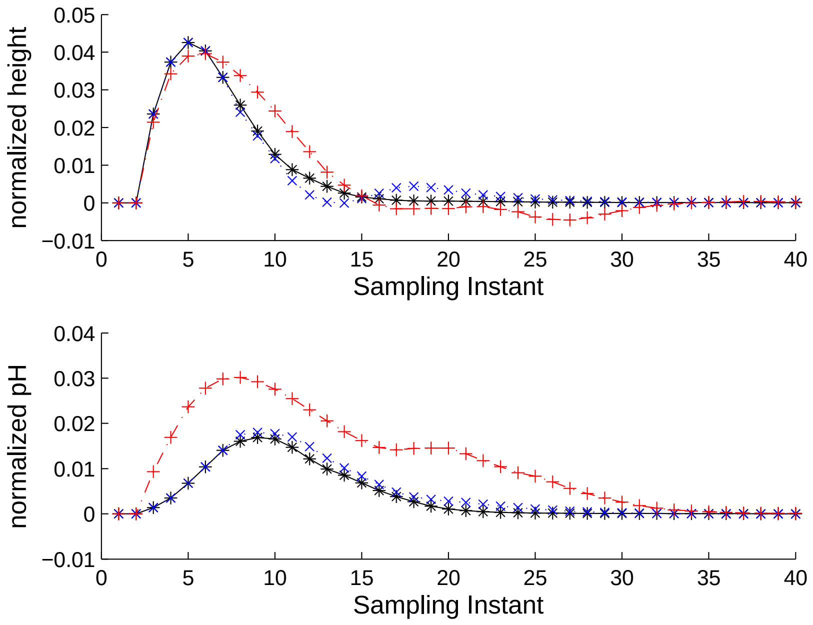
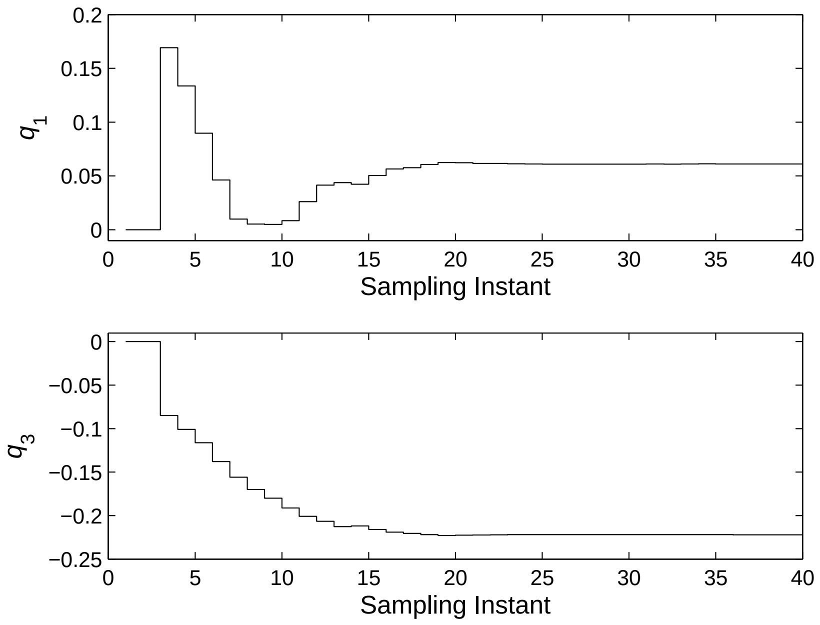


Figure 5.8: Manipulated variables profiles for MIMO case study, $p = 10$, $m = 2$, $W_i^{\Delta u_1} = 0.75$, $W_i^{\Delta u_2} = 1.5$ (The information is in deviation variables)



and the non-robust linear MPC. Eight different step-like disturbances were used to obtain the manipulated and controlled variables profiles. The characteristics of the disturbance can be found in Appendix E. The comparison of the three controllers was conducted for two different sets of values of the manipulated variables movement penalization terms, i.e., $W_i^{\Delta u_1}$, and $W_i^{\Delta u_2}$, $i = 1, \ldots, m$ and the IAE was used as an index to assess the closed-loop performance.

Table 5.4 shows the percentage of cases where the robust NMPC controller IAE was smaller than that of the non-robust NMPC and non-robust LMPC controllers for different values of $W_i^{\Delta u_1}$, and $W_i^{\Delta u_2}$, $i = 1, \ldots, m$. The trends of results are consistent with those obtained for the SISO system. Thus, when the manipulated variables' weights are small the controller is more aggressive and the closed-loop is more sensitive to model error. Thus for small weights the robust controller is better whereas for large weights the non-robust controller is superior.

Table 5.4: Comparison between robust NMPC, non-robust NMPC and linear MPC controllers for MIMO case study

$W_i^{\Delta u_1}$	$W_i^{\Delta u_2}$	% of cases with $\text{IAE}_{\text{robust NMPC}} < \text{IAE}_{\text{non-robust NMPC}}$ and $\text{IAE}_{\text{robust NMPC}} < \text{IAE}_{\text{non-robust LMPC}}$
0.1875	0.3750	66.66%
0.75	1.5	33.33%

5.5 Conclusions

A robust NMPC controller based on a Volterra series model was presented. The robustness tests are solved based on a structured singular value calculation of a predefined cost function. The Volterra series model is convenient for robust analysis and design because its structure can be easily split into a nominal and an uncertain part. Correspondingly, an interconnection matrix required for the robustness tests can be built. The algorithm solves an on-line min-max problem where the worst prediction obtained from a set of Volterra models describing the process is used to calculate the manipulated variables profile.

The algorithm includes enforcement of constraints in manipulated variables, penalization of manipulated variables moves and enforcement of a terminal condition that ensures convergence to a neighborhood of the set-point. To achieve cancellation of steady-state offset, a two mode control operation is used whereby the nonlinear model without uncertainty is used for prediction as soon as the output converges to a predefined neighborhood of the set-point defined by the terminal condition.

To test the proposed methodology two case studies were studied. It was found that the nonlinear controllers perform consistently better than a linear MPC controller for which predictions are done with the linear part of the Volterra model. The robust NMPC controller performs increasingly better than the non-robust NMPC counterpart as the penalization weight for manipulated variable moves decreases.

Chapter 6

Comparison between a robust nonlinear empirical model-based predictive controller and a first-principles nonlinear model-based predictive controller [*]

Overview

A robust nonlinear empirical model-based predictive controller is compared against a non-robust nonlinear first-principles model-based predictive controller. The robust controller uses a Volterra series representation of the system studied. Since the structure of the Volterra series can be decomposed into a nominal and an uncertain part a robustness test based on the structured singular value concept can be proposed. If the parameters of the first-principles model coincide with the operating parameters the performance of the first-principles controller will surpass that of the robust controller. However, if the parameters are different the performance of the first-principles controller starts deteriorating. Because the formulation of the robust controller explicitly considers that there will be a discrepancy between the process and the model, it is shown that in some case its performance can be superior to that of the first-principles controller. The study also shows a similar behavior when manipulated variables constraints are included in the formulation.

[*]The contents of this chapter were submitted to Industrial & Engineering Chemistry Research

6.1 Introduction

One of the most widely used control strategies in the process industry is model predictive control (MPC). An MPC controller minimizes at each sampling instant a cost function comprised of the sum of a term that penalizes the deviations of the predicted outputs from its predefined set-points plus a term that penalizes the manipulated variables movements. At the end of every sampling instant a new set of plant measurements is obtained and the optimization is re-done considering the new information. In order to calculate the values of the predicted outputs that will be used in the objective function of the controller, it is required to use a model that describes the interrelationship between the controlled and manipulated variables. The original MPC work and most current industrial implementations use linear models for prediction. However, the current research in MPC has mainly focused on the use of nonlinear models following the recognition that most processes are nonlinear thus the use of nonlinear models results in significantly improved closed-loop performance (Findeisen and Allgöwer, 2002). MPC algorithms that use nonlinear models to calculate the output predictions are referred to as nonlinear model predictive control (NMPC) algorithms. There are two types of models that have been used for calculating the predicted outputs in NMPC implementations: first-principles based and empirical models. First-principles models are based on the mass, energy and momentum balances of the process whereas empirical models are directly calibrated from input-output data. Empirical models are easier to obtain than first-principles models since they are based on straightforward regression of input-output data. On the other hand first-principles models have better extrapolation accuracy than empirical models implying that they are generally superior for predicting the process behavior at conditions that are different than the ones used for model calibration. Both types of models have been used in previous NMPC algorithms. First-principles model based NMPC has been addressed by (Wright and Edgar, 1994), (Santos et al., 2001) and (Nygaard and Nævdal, 2006). Some empirical models that have been considered for NMPC design are: polynomial autoregressive moving average models (Hérnandez and Arkun, 1993), Hammerstein models (Fruzzetti et al., 1997), Volterra series models (Doyle III et al., 1995), (Maner et al., 1996) and (Maner and Doyle III, 1997), Volterra-Laguerre models (Parker and Doyle III, 2001) and Wiener models (Norquay et al., 1999), (Gerkšič et al., 2000) and (Jeong et al., 2001).

Although several studies have been conducted on stability and on the use of different types of models for predictions, the robustness of these algorithms to model error has been identified as one of the key issues to be addressed for successful implementation (Findeisen and Allgöwer, 2002). For any mathematical model a discrepancy between the process output and the model output will always exists. Ignoring this discrepancy could result in poor closed-loop performance and in extreme cases closed-loop instability may occur. Thus, it is imperative to consider the effect of mismatch between the model and the process by designing a controller that will be robust with respect to this mismatch. The robustness of NMPC algorithms based on first-principles models have been only addressed through simulation studies (Nagy and Braatz, 2003), (Kawohl et al., 2007), (Magni and Scattolini, 2007), (Diehl et al., 2008) and (Zavala and Biegler, 2009) since analytical conditions for robustness are not possible with these models or have yet to be developed.

On the other hand it has been shown in (Díaz-Mendoza and Budman, 2010) that certain nonlinear empirical models can be used to formulate a novel NMPC algorithm that can be systematically tuned for robustness with respect to model error. The robust controller proposed in (Díaz-Mendoza and Budman, 2010) uses the structure of the Volterra series model to formulate a robust NMPC problem as a μ-structured singular value (SSV) (Doyle, 1982) test that can be used to calculate on-line the optimal control actions. At each sampling instant the μ-SSV based test is calculated and used for bounding the worst case predictions and constraint violations along predefined prediction and control horizons in the presence of disturbances, set-point changes and uncertainty in the parameters of the Volterra series model. This newly proposed Volterra based algorithm exhibited good robustness properties when operated around a fixed operating condition. However, the performance of this algorithm has not been investigated when the system is operated around a range of operating conditions due to changes in process parameters. In this case the uncertainty has to be formulated so as to capture the effect of these process parameters' variations. Also, the empirical model based algorithm proposed in (Díaz-Mendoza and

Budman, 2010) has not been compared before to NMPC algorithms that are based on a first-principles model where the latter has been generally the model of choice in most of the recent research in NMPC.

The objective of the current work is to compare the algorithm proposed in (Díaz-Mendoza and Budman, 2010) with a first-principles model based NMPC in the presence of disturbances and process parameter uncertainty. For this purpose a NMPC algorithm based on a first-principles model used in combination with an observer previously proposed by (Henson and Seborg, 1994) is formulated. This comparison is intended to address the relative performance tradeoffs between the expected superior extrapolating accuracy of first-principles model based NMPC versus the robustness properties of the empirical model based NMPC. It will be shown that for certain levels of model errors and in the presence of disturbances it is essential to consider robustness to improve the performance of the controller. Then, since first-principles models cannot be systematically designed for robustness, empirical model based NMPC will be shown to be an attractive option for NMPC implementation. To assess the relative merits of the controllers, a case study involving a multiple-input-multiple-output (MIMO) chemical process was considered.

This chapter is organized as follows. In section 6.2 the Volterra series model based robust NMPC controller is reviewed. Section 6.3 briefly explains the NMPC based on the nonlinear first-principles ordinary differential equations of the system that is used for comparison with the Volterra series model based controller. Section 6.4 presents the case study and conclusions are presented in section 6.5.

6.2 Robust nonlinear model predictive control based on Volterra series models

This section briefly describes the robust nonlinear model predictive control (RNMPC) algorithm recently proposed in (Díaz-Mendoza and Budman, 2010). The algorithm is based on Volterra series models that have been shown to effectively capture nonlinear behavior (Parker et al., 2001). The specific motivation to use this type of model for designing the robust NMPC is that it has a structure that can be split in two parts where the first part contains the nominal values of the Volterra series coefficients whereas the second part contains the values of the uncertainty associated to each parameter. This feature is used to formulate an on-line mathematical robust performance test.

To obtain a more compact representation of the model the RNMPC uses an autoregressive Volterra series model for which the relationship between the outputs and the inputs for a MIMO process is as follows:

$$\begin{aligned}\hat{y}_{\chi}(k) = &\sum_{q=1}^{n_{\text{ARX}}} h_{q_{\chi}} \hat{y}_{\chi}(k-q) + \\ &\left(\sum_{n=0}^{M-1} h_{n(\chi,1)} u_1(k-n) + \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} h_{i,j(\chi,1)} u_1(k-i)\, u_1(k-j) \right) + \ldots + \\ &\left(\sum_{n=0}^{M-1} h_{n(\chi,n_u)} u_{n_u}(k-n) + \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} h_{i,j(\chi,n_u)} u_{n_u}(k-i)\, u_{n_u}(k-j) \right)\end{aligned} \tag{6.1}$$

where n_{ARX} is the number of autoregressive terms, M is referred to as the memory of the system and it is generally chosen to correspond to the settling time of the process being modeled and $\chi = 1, \ldots, n_y$. The model parameters, i.e., the $h_i's$, are obtained by least square regression or nonlinear optimization using process input-output data by imposing an appropriate input sequence. In (Nowak and Van Veen, 1994) it was shown that to identify the coefficients for a system with polynomial degree N, it is necessary to use a $N+1$ level pseudorandom multilevel sequence (PRMS). Since the number of coefficients that must be identified significantly decreases with respect to the non-autoregressive case, autoregressive Volterra series models are simpler to use and have smaller sensitivity to noisy data. To provide for robustness when the process is expected to work around several operating regions the process identification procedure must be conducted around these different operating regions. This can be accomplished by using several PRMS around different operating regions and identifying the Volterra series coefficients from all the collected data.

Regardless of the number of terms included in the model and the use of data collected at different operating conditions, the output of a Volterra series model will always be different from the actual process output due to model truncation and round-off errors. Thus, when using a Volterra series model for model based control it is expected that performance will deteriorate due to the presence of this model error.

The effect that the mismatch has on the controller's performance can be considered by combining the nominal and uncertain parts of the model resulting in a family of models that represents the system to be controlled. The family of models is analyzed by the robust NMPC algorithm to obtain the model that results in the worst closed-loop performance. The basic premise used for designing the controller is that if the worst model in the set satisfies a specific performance index then it can be assumed that all of the other models, within the family of models considered for robust design, will also satisfy that performance level. The analysis of the worst model that is associated with the worst value of the elements of the **Y** vector for the robust NMPC controller can be formulated in terms of a SSV norm (Braatz et al., 1994), (Ma and Braatz, 2001) and (Nagy and Braatz, 2003). Following the above, the first step for designing a robust NMPC consists in modifying equation (6.1) to include coefficient uncertainty as follows:

$$
\begin{aligned}
\hat{y}_\chi(k) = {} & \sum_{q=1}^{n_{\mathrm{ARX}}} h_{q_\chi} \hat{y}_\chi(k-q) + \\
& \left(\sum_{n=0}^{M-1} \left(h_{n(\chi,1)} \pm \delta h_{n(\chi,1)} \right) u_1(k-n) + \right. \\
& \left. \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} \left(h_{i,j(\chi,1)} \pm \delta h_{i,j(\chi,1)} \right) u_1(k-i)\, u_1(k-j) \right) + \ldots + \\
& \left(\sum_{n=0}^{M-1} \left(h_{n(\chi,n_u)} \pm \delta h_{n(\chi,n_u)} \right) u_{n_u}(k-n) + \right. \\
& \left. \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} \left(h_{i,j(\chi,n_u)} \pm \delta h_{i,j(\chi,n_u)} \right) u_{n_u}(k-i)\, u_{n_u}(k-j) \right) + \\
& ic_\chi(k) + w_\chi(k) + d_\chi(k)
\end{aligned}
\tag{6.2}
$$

where δh_n, $n = 0, \ldots, (M-1)$ and $\delta h_{i,j}$, $i = 0, \ldots, (M-1)$, $j = i, \ldots, (M-1)$ are the uncertainties in the coefficients of the linear and nonlinear Volterra series terms, respectively, $ic_\chi(k)$, $\chi = 1, \ldots, n_y$ is the effect of the initial conditions, $w_\chi(k)$, $\chi = 1, \ldots, n_y$ is a feedback term that considers the effect of unmeasured disturbances and $d_\chi(k)$, $\chi =$

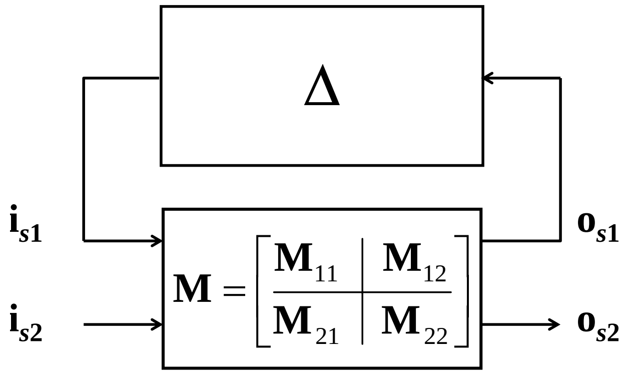
$1, \ldots, n_y$ is the effect of measured disturbances. The feedback term $w\chi(k)$ is calculated according to the following equation

$$w_\chi(k) = y_\chi^{\text{real}}(k-1) - \hat{y}_\chi(k-1) \tag{6.3}$$

For the purpose of prediction, the feedback term in equation (6.3) is assumed to be equal for all time intervals along the prediction horizon.

To formulate a robustness test based on a structured singular value calculation an appropriate interconnection matrix $\mathbf{M}$ and accompanying uncertainty description $\boldsymbol{\Delta}$ are needed. The interconnection between $\mathbf{M}$ and $\boldsymbol{\Delta}$ is shown schematically in Figure 6.1, the inputs to $\mathbf{M}$ are $\mathbf{i}_{s1}$ and $\mathbf{i}_{s2}$ and the outputs are $\mathbf{o}_{s1}$ and $\mathbf{o}_{s2}$. If $\mathbf{M}$ is partitioned into a structure that is compatible with the structure of the uncertainty matrix $\boldsymbol{\Delta}$, the equations that describe the system of Figure 6.1 are:

Figure 6.1: Uncertainty description for an upper LFT


$\boldsymbol{\Delta}$
$\mathbf{i}_{s1}$
$\mathbf{i}_{s2}$
$\mathbf{o}_{s1}$
$\mathbf{o}_{s2}$
$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \hline \mathbf{M}_{21} & \mathbf{M}_{22} \end{array}\right]$

$$\begin{bmatrix} \mathbf{o}_{s1} \\ \mathbf{o}_{s2} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{i}_{s1} \\ \mathbf{i}_{s2} \end{bmatrix} \tag{6.4}$$

$$\mathbf{i}_{s1} = \boldsymbol{\Delta}\mathbf{o}_{s1} \tag{6.5}$$

The relationship between the exogenous signals $\mathbf{o}_{s2}$ and $\mathbf{i}_{s2}$ is given by

$$\mathbf{o}_{s2} = \left[\mathbf{M}_{21}\boldsymbol{\Delta}\left[\mathbf{I} - \mathbf{M}_{11}\boldsymbol{\Delta}\right]^{-1}\mathbf{M}_{12} + \mathbf{M}_{22}\right]\mathbf{i}_{s2} \tag{6.6}$$

where $\mathbf{o}_{s2}$ is related to $\mathbf{Y}$ and $\mathbf{i}_{s2}$ is related to $w_{\chi}(k)$ and $d_{\chi}(k)$. Details on the construction of this interconnection are given in (Díaz-Mendoza and Budman, 2010). Based on these interconnected matrices, the worst value of the elements of the $\mathbf{Y}$ vector, i.e., the worst $||\mathbf{Y}||_{\infty}$, can be calculated by the following SSV test (Braatz et al., 1994):

$$\max_{\text{wrt}\,\mathbf{H}^{\text{L}},\mathbf{H}^{\text{NL}}} \|\mathbf{Y}\|_{\infty} \geq k_{ssv} \Leftrightarrow \mu_{\boldsymbol{\Delta}}(\mathbf{M}) \geq k_{ssv} \tag{6.7}$$

where the maximization is carried out with respect to the uncertainty in the Volterra series coefficients defined by the following vectors:

$$\begin{aligned} \mathbf{H}^{\text{L}} &= \left[\mathbf{H}_1^{\text{L}}, \mathbf{H}_2^{\text{L}}, \ldots, \mathbf{H}_{n_y}^{\text{L}}\right] \\ \mathbf{H}^{\text{NL}} &= \left[\mathbf{H}_1^{\text{NL}}, \mathbf{H}_2^{\text{NL}}, \ldots, \mathbf{H}_{n_y}^{\text{NL}}\right] \end{aligned} \tag{6.8}$$

with

$$\begin{aligned} \mathbf{H}_i^{\text{L}} &= \left[\delta h_{0\,(i,1)}, \ldots, \delta h_{0\,(i,n_u)}, \ldots, \delta h_{M-1\,(i,1)}, \ldots, h_{M-1\,(i,n_u)}\right] \\ \mathbf{H}_i^{\text{NL}} &= \left[\delta h_{0,0\,(i,1)}, \ldots, \delta h_{0,0\,(i,n_u)}, \ldots, \delta h_{M-1,M-1\,(i,1)}, \ldots, h_{M-1,M-1\,(i,n_u)}\right] \end{aligned} \tag{6.9}$$

The robustness test proposed in equation (6.7) is used to obtain a bound on the worst deviation of $||\mathbf{Y}||_{\infty}$ for the family of models defined by combining equation (6.2) and (6.8) in the presence of external disturbances. The test in equation (6.7) can be reformulated by the following constrained optimization problem (Braatz et al., 1994):

$$\max_{\text{wrt}\,\mathbf{H}^{\text{L}},\mathbf{H}^{\text{NL}}} \|\mathbf{Y}\|_{\infty} = \max_{\substack{\text{wrt}\,k_{ssv} \\ \text{st}\,\mu_{\boldsymbol{\Delta}}(\mathbf{M}) \geq k_{ssv}}} (k_{ssv}) \tag{6.10}$$

The skew μ problem can be extended to account for input and output constraints by appending the values that should be kept within bounds to the prediction vector $\mathbf{Y}$. Following this rationale it is possible to include: (i) constraints on manipulated variable changes $\Delta\mathbf{U}$ to prevent an excessive movement of the manipulated variables; (ii) constraints on the values of the manipulated variables $\mathbf{u}_{\text{limits}}$ to account for actuator limits and (iii) a terminal value condition $\mathbf{tc}$ to ensure convergence at steady state. The vector with the appended variables related to manipulated variable changes' values, manipulated variables value constraints and to the steady state output prediction can then be used within the following optimization problem:

$$\max_{\text{wrt}\,\mathbf{H}^{\mathrm{L}},\mathbf{H}^{\mathrm{NL}}} \left\| \begin{array}{c} \mathbf{Y} \\ \Delta\mathbf{U} \\ \mathbf{u}_{\text{limits}} \\ \mathbf{tc} \end{array} \right\|_{\infty} \geq k_{ssv} \Leftrightarrow \mu_{\mathbf{\Delta}}\left(\mathbf{M}\right) \geq k_{ssv} \tag{6.11}$$

Then, a skew μ problem is solved where a bound for the augmented vector of variables is solved as follows:

$$\max_{\text{wrt}\,\mathbf{H}^{\mathrm{L}},\mathbf{H}^{\mathrm{NL}}} \left\| \begin{array}{c} \mathbf{Y} \\ \Delta\mathbf{U} \\ \mathbf{u}_{\text{limits}} \\ \mathbf{tc} \end{array} \right\|_{\infty} = \max_{\substack{\text{wrt}\,k_{ssv} \\ \text{st}\,\mu_{\mathbf{\Delta}}(\mathbf{M})\geq k_{ssv}}} \left(k_{ssv}\right) \tag{6.12}$$

where the interconnection matrix $\mathbf{M}$ has a structure similar to equation (6.4), $\mathbf{M}_{11}$ and $\mathbf{M}_{22}$ are zero matrices, $\mathbf{M}_{12}$ is a function of the scalar k_{ssv} and the elements of the $\mathbf{U}$ vector, i.e., $\mathbf{M}_{12} = f\left(k_{ssv}, \mathbf{U}\right)$ and $\mathbf{M}_{21}$ is a function of k_{ssv} and the elements of the vectors $\mathbf{Y}$, $\Delta\mathbf{U}$, $\mathbf{u}_{\text{limits}}$ and $\mathbf{tc}$ i.e., $\mathbf{M}_{21} = \left(k_{\text{ssv}}, \mathbf{Y}, \Delta\mathbf{U}, \mathbf{u}_{\text{limits}}, \mathbf{tc}\right)$. Complete details on how to formulate the interconnection matrix $\mathbf{M}$, the prediction vector $\mathbf{Y}$ and the vectors quantifying $\Delta\mathbf{U}$, $\mathbf{u}_{\text{limits}}$ and $\mathbf{tc}$ to be used in equation (6.12) are provided in Appendix D.

The robust NMPC control law is based on the evaluation of the worst cost function when uncertainty is considered in the Volterra series coefficients by means of the SSV based test proposed in equation (6.12). The objective of the proposed controller is to minimize the value of the worst-case cost function with respect to the manipulated variables vector $\mathbf{U}$ at each sampling instant. Accordingly, the control calculation at each sampling instant consists of the solution of the following problem:

$$J = \min_{\text{wrt}\,\mathbf{U}} \left(\max_{\text{wrt}\,\mathbf{H}^{\mathrm{L}}\mathbf{H}^{\mathrm{NL}}} \left\| \begin{matrix} \mathbf{Y} \\ \Delta\mathbf{U} \\ \mathbf{u}_{\text{limits}} \\ \mathbf{tc} \end{matrix} \right\|_{\infty} \right) = \min_{\text{wrt}\,\mathbf{U}} \left(\max_{\substack{\text{wrt}\,k_{ssv} \\ \text{st}\,\mu_{\mathbf{\Delta}}(\mathbf{M}) \geq k_{ssv}}} (k_{ssv}) \right) \tag{6.13}$$

The controller formulation given by equation (6.13) has a disadvantage; it does not have integral action in the presence of model error as shown in (Díaz-Mendoza and Budman, 2010). To correct for this situation when the error is within a region specified by ϵ', i.e., when $||\mathbf{E}'||_{\infty} < \epsilon'$, then the prediction for the calculation of the worst case is done without considering the uncertainty resulting in zero offset (Díaz-Mendoza and Budman, 2010). This proposed correction results in a two-mode controller where, for $||\mathbf{E}'||_{\infty} \geq \epsilon'$ the Volterra model with uncertainty is used to calculate the worst case whereas for $||\mathbf{E}'||_{\infty} < \epsilon'$ the Volterra model without uncertainty is used for that calculation.

6.3 Nonlinear model predictive control based on first-principles model

This section describes the first-principles model based NMPC algorithm used for comparison with the robust controller presented in the previous section. In this case the equations that describe the interrelationship between the inputs, outputs and states can be written in discrete form as follows:

$$\begin{aligned} \mathbf{x}(k+1) &= f(\mathbf{x}(k), \mathbf{u}(k)) \\ \mathbf{y}(k) &= g(\mathbf{x}(k), \mathbf{u}(k)) \end{aligned} \tag{6.14}$$

where f and g are nonlinear vector fields obtained from the first-principles model, $\mathbf{x}$ are the system states, $\mathbf{u}$ are the system inputs and $\mathbf{y}$ are the system outputs. If some or all system states are not measured an observer or a predictor can be used to estimate their values. The corresponding predictor equations are as follows:

$$\begin{aligned}\hat{\mathbf{x}}(k+1) &= f\left(\hat{\mathbf{x}}(k), \mathbf{u}(k)\right) \\ \mathbf{y}(k) &= g\left(\hat{\mathbf{x}}(k), \mathbf{u}(k)\right)\end{aligned} \tag{6.15}$$

where $\hat{\mathbf{x}}$ denotes a predicted state. To ensure a fair comparison between the empirical based and first principle based NMPC's the cost functions of both algorithms were similar. Accordingly, the cost function can be written as

$$J = \min_{\text{wrt } \mathbf{U}} \left\| \begin{matrix} \mathbf{Y} \\ \Delta\mathbf{U} \\ \mathbf{u}_{\text{limits}} \\ \mathbf{tc} \end{matrix} \right\|_{\infty} \tag{6.16}$$

where $\Delta\mathbf{U}$ accounts for manipulated variables weighting, $\mathbf{u}_{\text{limits}}$ accounts for manipulated variables constraints and $\mathbf{tc}$ accounts for terminal condition constraints. The vector $\hat{\mathbf{Y}}$ that considers the deviation of the predicted controlled variables from its predefined set-points can be written as

$$\hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{y}^{\text{sp}} - \hat{\mathbf{y}}(k) \\ \vdots \\ \mathbf{y}^{\text{sp}} - \hat{\mathbf{y}}(k+p-1) \end{bmatrix} = \begin{bmatrix} \mathbf{y}^{\text{sp}} - \left(g\left(\hat{\mathbf{x}}(k), \mathbf{u}(k)\right) + \mathbf{fb}(k)\right) \\ \vdots \\ \mathbf{y}^{\text{sp}} - \left(g\left(\hat{\mathbf{x}}(k+p-1), \mathbf{u}(k+m-1)\right) + \mathbf{fb}(k)\right) \end{bmatrix} \tag{6.17}$$

where $\mathbf{fb}$ is a feedback term that is calculated as follows:

$$\mathbf{fb}(k) = \mathbf{y}^{\text{process}}(k-1) - \mathbf{y}(k-1) \tag{6.18}$$

The equation used to calculate the system states have the following form:

$$\hat{\mathbf{X}} = \begin{bmatrix} \hat{\mathbf{x}}(k+1) \\ \vdots \\ \hat{\mathbf{x}}(k+p-1) \end{bmatrix} = \begin{bmatrix} f\left(\hat{\mathbf{x}}(k), \mathbf{u}(k)\right) \\ \vdots \\ f\left(\hat{\mathbf{x}}(k+p-2), \mathbf{u}(k+m-1)\right) \end{bmatrix} \tag{6.19}$$

In this work, in order to isolate the effect of the controller from the impact of the observer the initial conditions of the estimated states were assumed to be equal to the actual states. The μ-based NMPC algorithm considers the constraints, i.e., $\mathbf{u}_{\text{limits}}$ and $\mathbf{tc}$, using the concept of a barrier function. Accordingly, the vectors $\mathbf{u}_{\text{limits}}$ and $\mathbf{tc}$ are formulated in the following way:

$$\mathbf{u}_{\text{limits}} = \begin{bmatrix} f_{\text{barrier}}\left(u_{1\text{bound}}\right)\log\left(1-\left|\frac{u_1(k)}{u_{1\text{bound}}}\right|\right) \\ \vdots \\ f_{\text{barrier}}\left(u_{1\text{bound}}\right)\log\left(1-\left|\frac{u_1(k+m)}{u_{1\text{bound}}}\right|\right) \\ \vdots \\ f_{\text{barrier}}\left(u_{n_u\text{bound}}\right)\log\left(1-\left|\frac{u_{n_u}(k)}{u_{n_u\text{bound}}}\right|\right) \\ \vdots \\ f_{\text{barrier}}\left(u_{n_u\text{bound}}\right)\log\left(1-\left|\frac{u_{n_u}(k+m)}{u_{n_u\text{bound}}}\right|\right) \end{bmatrix} \tag{6.20}$$

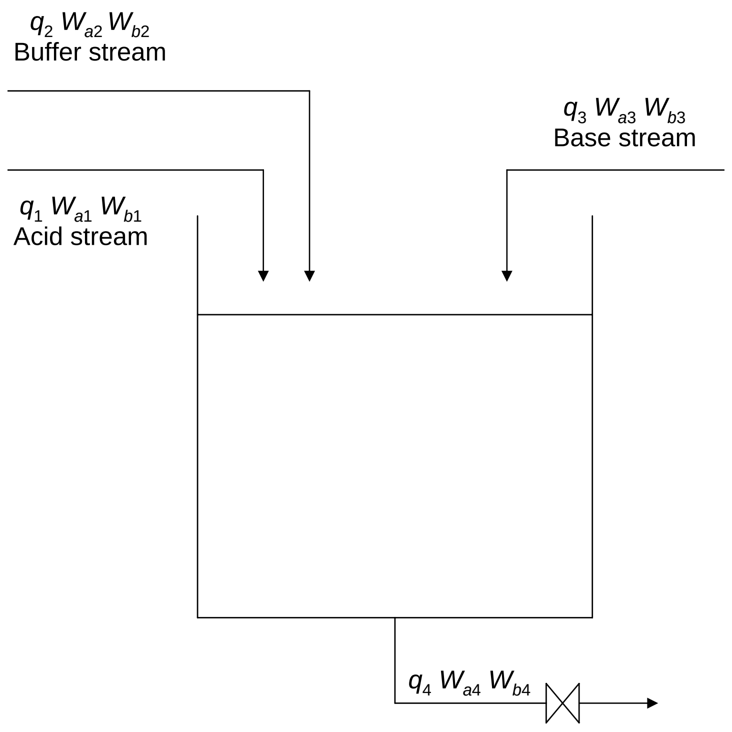
$$\mathbf{tc} = \begin{bmatrix} f_{\text{barrier}}\left(tc_1\right)\log\left(1-\left|\frac{y_1^{\text{sp}}-\hat{y}_1(k+p)}{tc_1}\right|\right) \\ \vdots \\ f_{\text{barrier}}\left(tc_{n_y}\right)\log\left(1-\left|\frac{y_{n_y}^{\text{sp}}-\hat{y}_{n_y}(k+p)}{tc_{n_y}}\right|\right) \end{bmatrix} \tag{6.21}$$

The structure of the vector $\Delta\mathbf{U}$ that accounts for manipulated variables weighting is the same for the RNMPC described in the previous section and the first-principles based NMPC shown in this section.

6.4 Case study

The system studied involves the pH neutralization process (Nahas et al., 1992) schematically shown in Figure 6.2 where an acid (q_1), a base (q_3) and a buffer (q_2) are mixed in a tank. Assuming perfect mixing, constant density and completely solubility of the ions involved, the chemical equilibrium can be modeled according to the following two reaction invariants for each stream $i = 1, \ldots, 4$:

Figure 6.2: pH neutralization system



$$W_{ai} = \left[\mathrm{H}^+\right]_i - \left[\mathrm{OH}^-\right]_i - \left[\mathrm{HCO}_3^-\right]_i - 2\left[\mathrm{CO}_3^{2-}\right]_i \tag{6.22}$$

$$W_{bi} = \left[\mathrm{H}_2\mathrm{CO}_3\right]_i + \left[\mathrm{HCO}_3^-\right]_i + \left[\mathrm{CO}_3^{2-}\right]_i \tag{6.23}$$

The overall mass balance on the tank yields:

$$A\frac{\mathrm{d}h}{\mathrm{d}t} = q_1 + q_2 + q_3 - q_4 \tag{6.24}$$

where q_4 is calculated as:

$$q_4 = Cv\,(h)^n \tag{6.25}$$

The equations that describe the dynamics of the effluent reaction invariants (W_{a4}, W_{b4}) can be obtained from a mass balance on each of the ionic species as follows:

$$Ah\frac{\mathrm{d}W_{a4}}{\mathrm{d}t} = q_1\left(W_{a1} - W_{a4}\right) + q_2\left(W_{a2} - W_{a4}\right) + q_3\left(W_{a3} - W_{a4}\right) \tag{6.26}$$

$$Ah\frac{\mathrm{d}W_{b4}}{\mathrm{d}t} = q_1\left(W_{b1} - W_{b4}\right) + q_2\left(W_{b2} - W_{b4}\right) + q_3\left(W_{b3} - W_{b4}\right) \tag{6.27}$$

Finally, the pH can be obtained from W_a and W_b by using the following nonlinear relationship:

$$W_a + 10^{\mathrm{pH}-14} + W_b\frac{1 + 2\times 10^{\mathrm{pH}-pK_2}}{1 + 10^{pK_1-\mathrm{pH}} + 10^{\mathrm{pH}-pK_2}} - 10^{-\mathrm{pH}} = 0 \tag{6.28}$$

The control problem consists in maintaining the tank height h and the outlet stream pH at its predefined set-points by manipulating the flows of the acid q_1 and base q_3 streams, in the presence of perturbations in the flow of the buffer stream q_2. The values of the operating conditions for the process are shown on Table 6.1 .

Table 6.1: Process operating conditions

A	$=$	207 cm^2	W_{a1}	$=$	3×10^{-3}
pK_1	$=$	6.35	W_{a2}	$=$	-3×10^{-2}
pK_2	$=$	10.25	W_{a3}	$=$	-3.05×10^{-2}
q_1	$=$	3×10^{-3}M HNO_3	W_{b1}	$=$	0
q_2	$=$	3×10^{-2}M $NaHCO_3$	W_{b2}	$=$	3×10^{-2}
q_3	$=$	3×10^{-3}M NaOH + 5×10^{-5}M $NaHCO_3$	W_{b3}	$=$	5×10^{-5}

The RNMPC controller requires a Volterra series model that describes the interrelationship between the manipulated variables (q_1 and q_3) and the controlled variables (h and pH). An autoregressive Volterra series model with $M = 2$ and $n_{\mathrm{ARX}} = 1$ was used for the RNMPC. These values were found by trial and error to result in a reasonable fitting between the process and the model. The sampling time used for identification and simulation purposes was 25s. To prevent numerical difficulties during the identification process the manipulated and controlled variables were normalized as follows:

$$v_n = \frac{v - v_p}{v_d} \tag{6.29}$$

where v_n is the normalized value of the variable v, the values of v_p and v_d can be found in Table 6.2

Table 6.2: Values for normalization

variable	v_p	v_d
y_1 (height)	14	5
y_2 (pH)	7	3
q_1 (acid flow)	q_1^{ss}	$0.2q_1^{\text{ss}}$
q_3 (base flow)	q_3^{ss}	$0.225q_3^{\text{ss}}$

In order to obtain the nominal and uncertain value of the autoregressive Volterra series coefficients r_{seq} different PRMS sequences were applied to the system at different operating conditions and a set of Volterra series coefficients were obtained by regression. These set of coefficients was divided into two subsets corresponding to the coefficients of the linear and nonlinear terms. Correspondingly, **VC** defines the set of Volterra series coefficients as follows:

$$\mathbf{VC} = \left[\mathbf{VC}_1^{\text{L}}, \ldots, \mathbf{VC}_{n_y}^{\text{L}}, \mathbf{VC}_1^{\text{NL}}, \ldots, \mathbf{VC}_{n_y}^{\text{NL}}\right] \tag{6.30}$$

$$\begin{aligned} \mathbf{VC}_i^{\text{L}} &= \left[h_{0(i,1)}, \ldots, h_{0(i,n_u)}, \ldots, h_{M-1(i,1)}, \ldots, h_{M-1(i,n_u)}\right] \\ \mathbf{VC}_i^{\text{NL}} &= \left[h_{0,0(i,1)}, \ldots, h_{0,0(i,n_u)}, \ldots, h_{M-1,M-1(i,1)}, \ldots, h_{M-1,M-1(i,n_u)}\right] \end{aligned} \tag{6.31}$$

for the j-th PRMS sequence the identified Volterra series coefficients are:

$$\begin{aligned} \mathbf{VCid}^{\text{L}} &= \left[\mathbf{VCid}_1^{\text{L}}, \ldots, \mathbf{VCid}_{n_{\text{seq}}}^{\text{L}}\right] \\ \mathbf{VCid}^{\text{NL}} &= \left[\mathbf{VCid}_1^{\text{NL}}, \ldots, \mathbf{VCid}_{n_{\text{seq}}}^{\text{NL}}\right] \end{aligned} \tag{6.32}$$

where the elements in equation (6.32) are also vectors defined as follows

$$\begin{aligned} \mathbf{VCid}_j^{\text{L}} &= \left[\mathbf{VCid}_{j,1}^{\text{L}}, \ldots, \mathbf{VCid}_{j,n_y}^{\text{L}}\right] \\ \mathbf{VCid}_j^{\text{NL}} &= \left[\mathbf{VCid}_{j,1}^{\text{NL}}, \ldots, \mathbf{VCid}_{j,n_y}^{\text{NL}}\right] \end{aligned} \tag{6.33}$$

$$\begin{aligned} \mathbf{VCid}_{j,i}^{\text{L}} &= \left[h_{0(i,1),j}, \ldots, h_{0(i,n_u),j}, \ldots, h_{M-1(i,1),j}, \ldots, h_{M-1(i,n_u),j}\right] \\ \mathbf{VCid}_{j,i}^{\text{NL}} &= \left[h_{0,0(i,1),j}, \ldots, h_{0,0(i,n_u),j}, \ldots, h_{M-1,M-1(i,1),j}, \ldots, h_{M-1,M-1(i,n_u),j}\right] \end{aligned} \tag{6.34}$$

The nominal value of the coefficients that define the nominal model and that were used for the construction of the interconnection matrix $\mathbf{M}$ was calculated as follows:

$$\begin{aligned}\mathbf{VC}^{\mathrm{L}} &= \sum_{j=1}^{n_{seq}} \frac{\mathbf{VCid}_j^{\mathrm{L}}}{n_{seq}} \\ \mathbf{VC}^{\mathrm{NL}} &= \sum_{j=1}^{n_{seq}} \frac{\mathbf{VCid}_j^{\mathrm{NL}}}{n_{seq}}\end{aligned} \tag{6.35}$$

where n_{seq} is the number of PRMS sequences used ($n_{seq} = r_{seq} \times$number of operating regions). The elements of the vectors $\mathbf{VC}^{\mathrm{L}}$ and $\mathbf{VC}^{\mathrm{NL}}$ are equal to the means of [$\mathbf{VCid}_1^{\mathrm{L}}$, …, $\mathbf{VCid}_{n_{seq}}^{\mathrm{L}}$] and [$\mathbf{VCid}_1^{\mathrm{NL}}$, …, $\mathbf{VCid}_{n_{seq}}^{\mathrm{NL}}$] , respectively.

The associated uncertainty matrix used for the construction of the interconnection matrix $\mathbf{M}$ was calculated based on 2 standard deviations as follows:

$$\begin{aligned}\mathbf{H}^{\mathrm{L}} &= 2\sqrt{\frac{1}{n_{seq}} \sum_{j=1}^{n_{seq}} \left(\mathbf{VCid}_j^{\mathrm{L}} - \mathbf{VC}^{\mathrm{L}}\right)^2} \\ \mathbf{H}^{\mathrm{NL}} &= 2\sqrt{\frac{1}{n_{seq}} \sum_{j=1}^{n_{seq}} \left(\mathbf{VCid}_j^{\mathrm{NL}} - \mathbf{VC}^{\mathrm{NL}}\right)^2}\end{aligned} \tag{6.36}$$

Two Volterra series models were proposed. The first describes the relationship between the tank height and q_1 and q_3, i.e., $h = h(q_1, q_3)$, the second set describes the relationship between the pH and q_1 and q_3, i.e., $\mathrm{pH} = \mathrm{pH}(q_1, q_3)$.

The FP-NMPC controller uses the ODE's and algebraic equations that describe the behavior of the process i.e., equations (6.24) to (6.28). Because some of the states were assumed to be not measurable an observer (Henson and Seborg, 1994) was used to estimate W_a and W_b.

Simulations were conducted to test the controllers' abilities to reject disturbances as well as to track set-point changes. A disturbance profile consisting of a superposition of random changes in q_2 shown schematically in Figure 6.3 was used for comparing the controllers whereas the sequence of set-point changes used for the comparisons is shown schematically in Figure 6.4. The characteristics of the disturbance can be found in Appendix F.

Figure 6.3: Disturbance profile

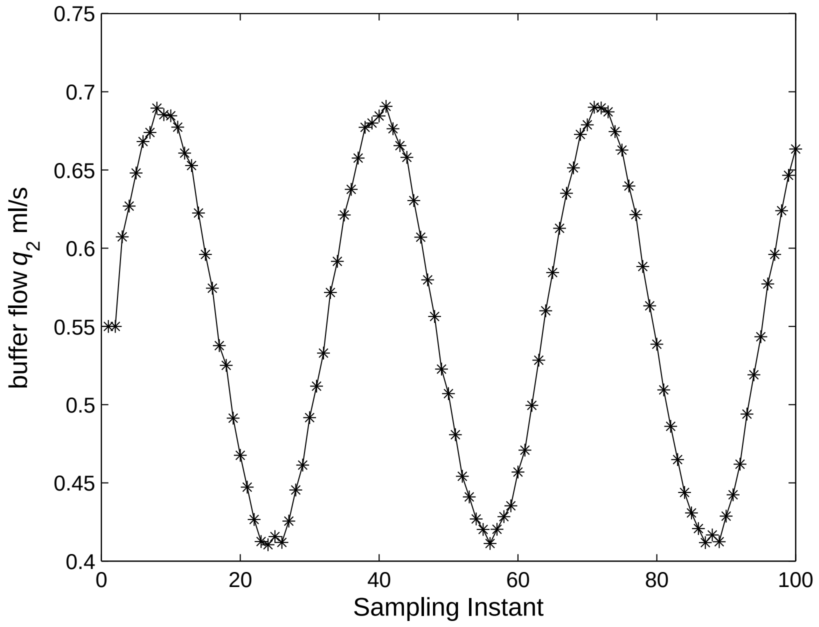
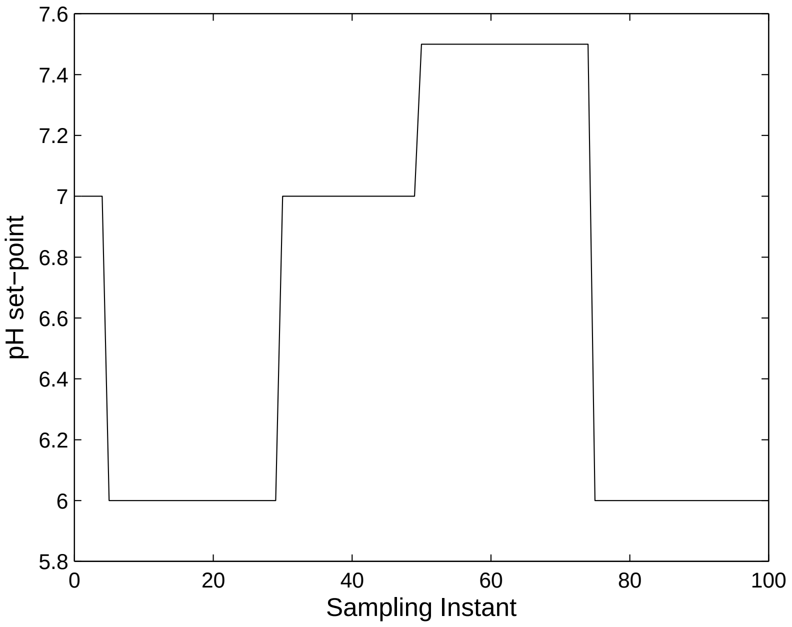


Figure 6.4: Set-point changes profile



In the absence of model errors, the FP-NMPC is obviously expected to provide better performance than the robust controller based on the empirical model because the prediction based on the first-principle model will be more accurate. Thus, the real test between the FP-NMPC and the RNMPC consists in assessing which of the two controllers is superior when model errors are present. The current comparison study considers uncertainty in two key valve parameters: Cv and n. For the purpose of discussion, the values of these parameters that are assumed as the correct ones by the first-principles model will be referred heretofore as Cv_{model} and n_{model} whereas the actual values occurring in the process will be referred to as Cv_{process} and n_{process}. Although the empirical Volterra series model does not explicitly use the values of these parameters within the algorithm, the nominal coefficient values of this model together with the uncertainties associated to these coefficients are identified from input-output data collected from experiments performed around different combinations of the values of Cv and n. Thus the family of models described by the Volterra series model with uncertainty is expected to capture the behavior of the process in the presence of changes in the values of Cv and n. A key challenge for conducting a fair comparison between the controllers is that the robust controller is designed for a worst case scenario whereas the system is not necessarily operated always at worst case conditions. Therefore, the approach adopted for conducting the comparative study is to test the closed-loop operation at different combinations of Cv and n and then assess the performance on an average basis for all these combinations. In that way an average performance for different realizations of the uncertain process parameters can be quantified. The main difference is that while the FP-NMPC used the same values of Cv_{model} and n_{model} for all the combinations of Cv_{process} and n_{process} tested, the robust controller used a model which was identified based on data obtained around the different combinations. Thus, the objective is to test whether the possibility of variations in Cv and n justify the use of a robust controller as compared to a first-principles controller that does not take into account these variations but on the other hand is expected to have better prediction accuracy in the absence of these variations.

Since the objective function of both controllers is based on an infinity norm, the integral of the absolute error (IAE) was selected as the measurement of the controller's performance.

Since a tradeoff is always expected between performance and sensitivity to model error, the comparisons were conducted for different values of the manipulated variables weights which determine the speed of the closed loop response. Six different combinations of weights $\left[W_i^{\Delta u_1}, W_i^{\Delta u_2}\right]$, $\forall i \in [1, m]$ were used as follows: case 1 $[0.15, 0.1]$, case 2 $[0.25, 0.1]$, case 3 $[0.3, 0.1]$, case 4 $[0.1, 0.15]$, case 5 $[0.1, 0.25]$ and case 6 $[0.1, 0.3]$. For all the cases the prediction horizon p was set to 10 and the control horizon m was set to 2. The FP-NMPC was simulated with the following values: $Cv_{\text{model}} = 8.75$ and $n_{\text{model}} = 0.5$. For each case study 9 different combinations of Cv_{process} and n_{process} were simulated. Table 6.3 to Table 6.8 show the values of the IAE for all the 9 combinations for both controllers. Table 6.9 shows the average IAE for both controllers.

Table 6.3: IAE results for case 1 $[\text{IAE}_{\text{FP}-\text{NMPC}}, \text{IAE}_{\text{RNMPC}}]$

$[0.15, 0.10]$	$n = 0.45$	$n = 0.50$	$n = 0.55$
$Cv = 9.25$	$[5.36, 4.90]$	$[4.20, 4.05]$	$[3.54, 5.57]$
$Cv = 8.75$	$[9.64, 5.02]$	$[3.78, 4.40]$	$[4.50, 3.68]$
$Cv = 8.25$	$[4.23, 5.48]$	$[10.06, 4.41]$	$[3.46, 4.26]$

Table 6.4: IAE results for case 2 $[\text{IAE}_{\text{FP}-\text{NMPC}}, \text{IAE}_{\text{RNMPC}}]$

$[0.25, 0.10]$	$n = 0.45$	$n = 0.50$	$n = 0.55$
$Cv = 9.25$	$[3.95, 5.50]$	$[3.14, 4.62]$	$[5.41, 4.62]$
$Cv = 8.75$	$[3.76, 6.01]$	$[4.80, 4.97]$	$[3.61, 4.45]$
$Cv = 8.25$	$[3.23, 6.47]$	$[4.23, 5.35]$	$[3.41, 4.67]$

Table 6.5: IAE results for case 3 $[\text{IAE}_{\text{FP}-\text{NMPC}}, \text{IAE}_{\text{RNMPC}}]$

$[0.30, 0.10]$	$n = 0.45$	$n = 0.50$	$n = 0.55$
$Cv = 9.25$	$[4.41, 5.75]$	$[7.81, 5.12]$	$[3.28, 5.14]$
$Cv = 8.75$	$[3.60, 5.86]$	$[3.26, 6.10]$	$[3.38, 4.49]$
$Cv = 8.25$	$[3.80, 6.96]$	$[3.23, 6.16]$	$[5.57, 5.28]$

Table 6.6: IAE results for case 4 $[\text{IAE}_{\text{FP-NMPC}}, \text{IAE}_{\text{RNMPC}}]$

$[0.10, 0.15]$	$n = 0.45$	$n = 0.50$	$n = 0.55$
$Cv = 9.25$	$[5.39, 5.07]$	$[4.90, 4.18]$	$[3.63, 4.29]$
$Cv = 8.75$	$[2.82, 5.18]$	$[2.74, 4.56]$	$[5.09, 3.61]$
$Cv = 8.25$	$[10.25, 6.10]$	$[3.49, 5.01]$	$[2.63, 4.32]$

Table 6.7: IAE results for case 5 $[\text{IAE}_{\text{FP-NMPC}}, \text{IAE}_{\text{RNMPC}}]$

$[0.10, 0.25]$	$n = 0.45$	$n = 0.50$	$n = 0.55$
$Cv = 9.25$	$[7.35, 5.60]$	$[7.69, 5.13]$	$[4.06, 4.75]$
$Cv = 8.75$	$[3.52, 6.28]$	$[3.00, 5.64]$	$[2.62, 4.30]$
$Cv = 8.25$	$[4.02, 7.00]$	$[4.27, 5.57]$	$[3.25, 5.13]$

Table 6.8: IAE results for case 6 $[\text{IAE}_{\text{FP-NMPC}}, \text{IAE}_{\text{RNMPC}}]$

$[0.10, 0.30]$	$n = 0.45$	$n = 0.50$	$n = 0.55$
$Cv = 9.25$	$[5.13, 5.96]$	$[3.35, 5.76]$	$[9.65, 4.22]$
$Cv = 8.75$	$[4.01, 6.67]$	$[3.37, 5.92]$	$[6.42, 5.16]$
$Cv = 8.25$	$[5.83, 7.29]$	$[6.12, 5.71]$	$[8.47, 5.39]$

Table 6.9: Average IAE for case studies 1 to 6

	$\overline{\text{IAE}}_{\text{FP-NMPC}}$	$\overline{\text{IAE}}_{\text{RNMPC}}$
Case 1	5.42	4.64
Case 2	3.95	5.18
Case 3	4.26	5.65
Case 4	4.54	4.70
Case 5	4.42	5.49
Case 6	5.82	5.79

The first observation that can be made from Table 6.3 to Table 6.8 is that for all combinations of weights, whenever the simulations are conducted without model error (center value in each table) the FP-NMPC outperforms RNMPC in terms of IAE. As mentioned above this result is expected since in the absence of model error, the predictions used in the FP-NMPC are perfect. For these cases the improvements in IAE for the FP-NMPC over the RNMPC range from 1% for case study 2, ($\mathrm{IAE}_{\mathrm{FP-NMPC}}$=4.8 versus $\mathrm{IAE}_{\mathrm{RNMPC}}$=4.97 in Table 6.4), to 53% for case study 3, ($\mathrm{IAE}_{\mathrm{FP-NMPC}}$=3.26 versus $\mathrm{IAE}_{\mathrm{RNMPC}}$=6.1 in Table 6.5). The evolution of the controlled variables with respect to time for these two cases is shown in Figure 6.5 and Figure 6.6, respectively.

Figure 6.5: Controlled variables profiles, $W_i^{\Delta u_1} = 0.25$, $W_i^{\Delta u_2} = 0.10$, $Cv = 8.75$, $n = 0.5$, (blue=FP-NMPC), (red=RNMPC)

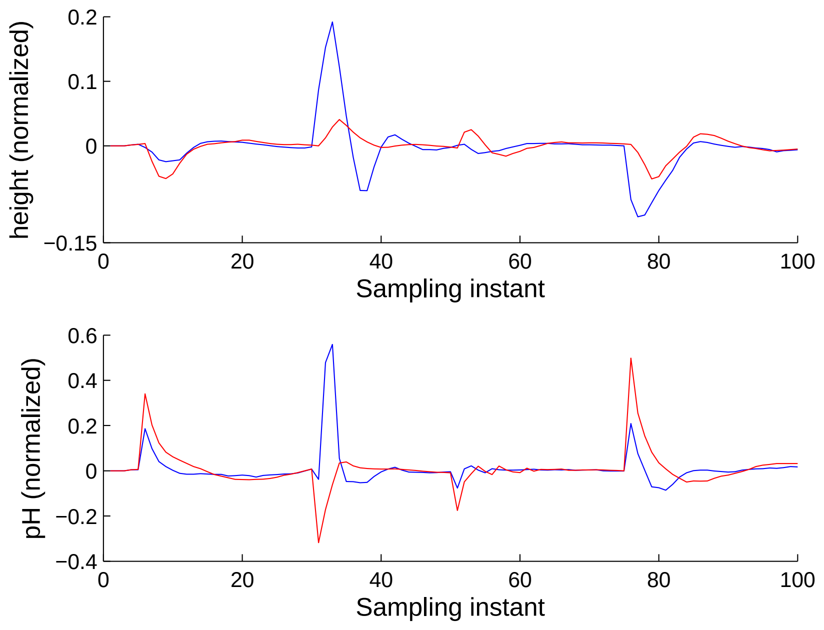
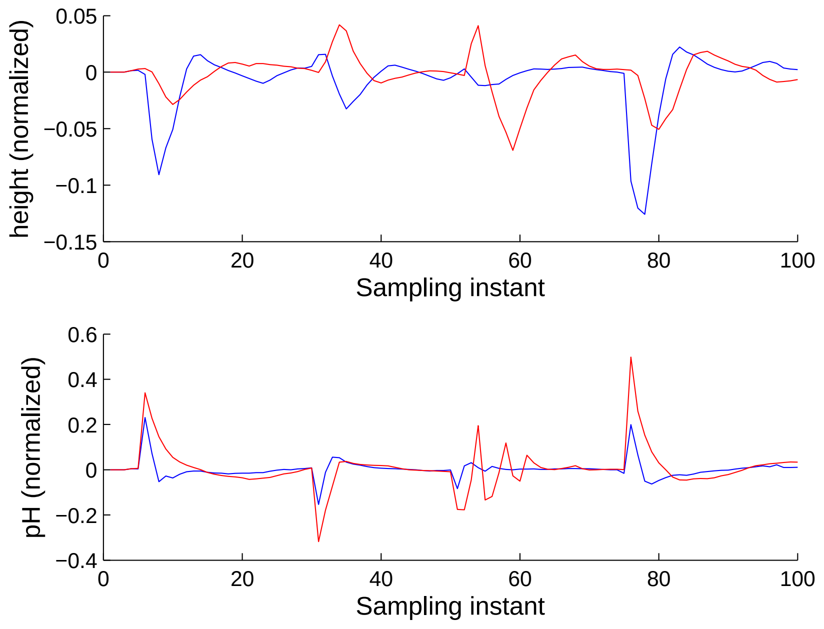


Figure 6.6: Controlled variables profiles, $W_i^{\Delta u_1} = 0.30$, $W_i^{\Delta u_2} = 0.10$, $Cv = 8.75$, $n = 0.5$, (blue=FP-NMPC), (red=RNMPC)



When model error is present, the IAE results show that the performance depends on the particular combination of weights and parameter uncertainty considered. In some cases the FP-NMPC outperforms the RNMPC by as much as 64% corresponding to a case in Table 6.7 ($Cv = 8.75$ and $n = 0.55$) whereas in another cases the RNMPC is better by as much as 57% corresponding to a case in Table 6.8 ($Cv = 8.25$ and $n = 0.55$). In view of this variability in performance of the two controllers, an average IAE for each case study was calculated to enable a more systematic comparison between the two controllers. These calculated average values are presented in Table 6.9. The results from Table 6.9 indicate that for two out of the six cases that were considered (case studies 1 and 6) the IAE values of the RNMPC are lower than the FP-NMPC. If the average IAE is considered the biggest improvement is of 16% corresponding to case study 1. The improvement obtained for case study 1 can be explained by the fact that the manipulated variable weights $W_i^{\Delta u_1}$ and $W_i^{\Delta u_2}$ used for this case were both very small verifying the fact that more aggressive control results in higher sensitivity to model error. For both cases that correspond to a combination of small manipulated variables' weights it was found that there is a relatively high number of Cv and n combinations for which the RNMPC outperforms the FP-NMPC, e.g., 5 cases out of 9 in Table 6.3 ($W_i^{\Delta u_1} = 0.15$ and $W_i^{\Delta u_2} = 0.1$) and 4 cases out of 9 in Table 6.6 ($W_i^{\Delta u_1} = 0.1$ and $W_i^{\Delta u_2} = 0.15$). Thus, the robust controller can be a better alternative than the non-robust controller especially in those cases when the controller weights are small. Table 6.9 also indicates that when the controller weights increase the performance of the RNMPC starts degrading and as a consequence the performance of the FP-NMPC becomes superior. For example as $W_i^{\Delta u_1}$ increases from 0.1 to 0.3 the IAE consistently increases for the RNMPC from 4.64 to 5.65 (Table 6.9). A similar trend is observed when $W_i^{\Delta u_2}$ increases resulting in a corresponding increase of the IAE for RNMPC from 4.70 to 5.79 (Table 6.9).

In general using the average in Table 6.9 one could argue that based on all the simulation studies, the preferable controller is the FP-NMPC with $W_i^{\Delta u_1} = 0.25$ and $W_i^{\Delta u_2} = 0.1$ (case study 2) since it results in the lowest average IAE. However, this will be only true if all the combinations of parameter changes have equal probability which is not necessarily true. For instance, if the case of $Cv = 9.25$ and $n = 0.55$ would have a higher occurrence probability than the other cases the RNMPC will be the controller of choice.

In order to assess the performance of both controllers when manipulated variables constraints are enforced a new set of simulations were carried out for those conditions in which the difference between the RNMPC and FP-NMPC was higher as follows: (i) case study 7 considered a manipulated variable weighting matrix of $[0.15, 0.1]$, $Cv_{\text{process}} = 8.25$ and $n_{\text{process}} = 0.5$, (ii) case study 8 considered a manipulated variable weighting matrix of $[0.1, 0.15]$, $Cv_{\text{process}} = 8.25$ and $n_{\text{process}} = 0.45$. To assess the degradation in performance four different values of the manipulated variables limits were tested.

Table 6.10 and Table 6.11 show the results for case studies 7 and 8, respectively. The results generally indicate as expected that when constraints are considered the performance deteriorates for both controllers since the potential for manipulation overall decreases. However, it is also observed from Table 6.10 and Table 6.11 that for case studies 7 and 8, for which the RNMPC is better than the FP-NMPC in the absence of constraints ($|\mathbf{u}| < \infty$), the difference in IAEs between the two controllers become smaller. This behavior is explained by the fact that in the presence of constraints the controllers are less aggressive resulting in less sensitivity of the FP-NMPC controller to model error as compared to the unconstrained case. Despite this fact, the performance of the RNMPC in the presence of constraints for case studies 7 and 8 remains better than the FP-NMPC.

Table 6.10: IAE for case study 7

	$\mathrm{IAE}_{\mathrm{FP-NMPC}}$	$\mathrm{IAE}_{\mathrm{RNMPC}}$
$\lvert\mathbf{u}\rvert \leq 0.40$	9.54	7.67
$\lvert\mathbf{u}\rvert \leq 0.45$	8.22	7.16
$\lvert\mathbf{u}\rvert \leq 0.50$	7.54	6.66
$\lvert\mathbf{u}\rvert \leq 0.95$	5.17	4.72
$\lvert\mathbf{u}\rvert \leq \infty$	10.06	4.41

Table 6.11: IAE for case study 8

	$\mathrm{IAE}_{\mathrm{FP-NMPC}}$	$\mathrm{IAE}_{\mathrm{RNMPC}}$
$\lvert\mathbf{u}\rvert \leq 0.40$	10.36	9.07
$\lvert\mathbf{u}\rvert \leq 0.45$	8.97	8.56
$\lvert\mathbf{u}\rvert \leq 0.50$	9.04	7.93
$\lvert\mathbf{u}\rvert \leq 0.95$	5.56	5.57
$\lvert\mathbf{u}\rvert \leq \infty$	10.25	6.10

6.5 Conclusions

A comparison between a non-robust NMPC based on a first-principles model and a robust NMPC controller based on a Volterra series model was presented. The first-principles controller was designed based on the set of nonlinear ODE's that represent the process and an observer to account for the unmeasured states. The robust NMPC controller exploited the structure of the Volterra series model to formulate a robustness test based on the structured singular value norm (μ). A key practical advantage of the Volterra series model based controller is that it can be obtained directly from input-output data and it does not require a first-principles model that is often difficult to obtain.

Since it was considered that both controllers must work for a whole range of operating conditions the identification of the parameters of the Volterra series was based on input-output data collected around different operating regions. To account for model parameter changes the RNMPC considered that the parameters of the Volterra series model are bounded between upper and lower values. This variation was mathematically represented as a nominal value with an uncertainty description which was used to formulate the robustness test. On the other hand, unless a difficult adaptation scheme is used, the first-principles nonlinear model is generally used with fixed values of physical model parameters.

The objective function of both controllers penalizes the infinity norm of a vector that is formed by combining four terms: (i) the deviation from the set-point of the future predictions, (ii) the penalization of the manipulated variables movement, (iii) the manipulated variables constraints and (iv) the terminal conditions. Because the objective function was based on an infinity norm the performance of both controllers were assessed based on their IAE.

The comparative study showed that in the presence of model parameters' errors there are many scenarios where the RNMPC controller performance can surpass the performance of a first-principles controller. In general for aggressive controllers obtained by using small manipulated variable weights, the RNMPC is better since it results in less sensitivity to model error. When manipulated variables constraints were imposed on the process, it was observed that the IAE of the RNMPC increased and the IAE of the FP-NMPC decreased. However, the RNMPC still outperformed the FP-NMPC as indicated by lower IAE values.

Chapter 7

Conclusions

This thesis presented two methodologies to design robust nonlinear MPC controllers for processes that can be represented by empirical models. Since most chemical processes are nonlinear it is important that the model used by the control strategy for prediction accounts for the nonlinearity of the process. For model prediction two models were used: a family of linear models for gain-scheduled predictive control and a Volterra series model for robust NMPC. For the purpose of analysis of robustness a state-affine nonlinear model was used for designing a robust gain-scheduled controller. The key advantage in using this model for analysis is that the uncertainty description can be easily separated from the nominal part of the model and this uncertainty description is directly related to the manipulated variables. In the case of the Volterra series model the uncertainty was associated to the model coefficients and since the model is linear with respect to these coefficients the separation between the uncertainty and the nominal model values was also feasible. By using empirical models the proposed methodologies have a wide range of application, since they can be used to control any process that can be represented either by a state-affine or by Volterra series models. These models can be easily identified from input-output data collected from the actual process.

7.1 Robust gain-scheduling NMPC

The first proposed methodology is a gain-scheduling robust predictive controller. The controller works by obtaining local state-affine models around different operating regions. Every model is represented by a combination of linear and nonlinear terms that were obtained from a least-squares calculation using input-output data. The linear terms of the model are arranged to obtain a state-space representation that is used for output prediction within a MPC controller for each operating region, whereas the nonlinear terms can be viewed as the uncertainty associated to this model and is accounted for through a robust design. The use of a state-space model for designing the individual controllers is a significant improvement over a previous step-response model-based algorithm proposed by (Gao, 2004). In that work, due to the use of the step-response model for prediction, the dimensions of the closed-loop matrix representation grew directly as a function of the prediction horizon resulting in a rapid increase in memory and computational demand. This imposed a limitation with respect to the choice of the prediction horizon. Through the use of a state-space representation that limitation has been completely removed in the current study. Also, the current work provides a more consistent design since the output prediction is based on the linear part of the nonlinear state-affine model used for robust analysis, whereas in the previous work (Gao, 2004) the model used for prediction was an impulse-response approximation of the linear part of the nonlinear state-affine model, whereas the model used for design was of state-affine type. Based on the family of linear models used for output prediction and based on the uncertainty description obtained from the nonlinear state-affine model a robust control test was formulated in terms of a finite set of LMI's.

The system of LMI's was solved in Matlab using two different toolboxes: YALMIP (Löfberg, 2004) and SDPT3 (Toh et al., 1999). The YALMIP toolbox works as an interface between Matlab and an external SDP solver, since the LMI system was solved in a 64 bit computer the SDP selected was SDPT3. The external toolboxes solve the system of LMI's faster and are numerically efficient, i.e., the amount of points where the solver cannot find a solution decreases compared with Matlab.

The gain-scheduling part of the control strategy works by switching between the different controllers available according to a pre-specified set of rules. For the case study presented the manipulated variable was chosen as the scheduling variable because, according to the state-affine model structure, it was highly representative of the process nonlinearity. However, a different type of process might have different scheduling variables.

A key aspect of the algorithm is that the manipulated variables values are considered as model uncertainty. Thus, it is necessary to provide a bound over the value that the manipulated variables acquire through the process. In the previous study of (Gao and Budman, 2004) and (Gao, 2004) these bounds were assumed to be large values resulting in very conservative designs. The algorithm proposed calculated the bounds iteratively resulting in much tighter bounds and consequently, in less conservative controllers.

To assess the benefits of the methodology the robust gain-scheduling MPC controller was compared against a robust linear MPC controller, i.e., an MPC controller based on one single linear model. Two case studies were proposed and each one considered three operating regions, in each region a GSMPC controller was obtained. A first step was carried out to check whether the implementation of a GSMPC is worthwhile by testing the performance improvement with respect to the linear MPC in the absence of uncertainty. The results for the case study indicated that the improvement of the GSMPC was of at least 12% and as high as 33%. However, this is a theoretical limit since it corresponds to the case when uncertainties are ignored. When uncertainties were considered the improvement of the GSMPC over the LMPC was of at least 13%. If only a particular region of operation was considered it was found that the biggest improvement could be achieved around $S_{\text{in}} = 2.0$ g/L.

To test the optimization results a series of simulations were carried out to check the controllers' ability to reject disturbances. Several disturbances were simulated and in all cases it was confirmed that the controllers were able to effectively reject their effect by returning the controlled variables to their set-point. The simulations also show that the manipulated variables did not violate the calculated uncertainty bounds used to design the controller. Finally, the γ^{sim} results confirmed the theoretical results, i.e., that the performance of the GSMPC is superior to the LMPC.

The γ index proposed in chapter 3 only considered the effect of the controlled variables. In order to assess the effect that the manipulated variables might have in the controller's performance a new controller was designed in chapter 4. The main difference is that the γ

index of the new controller considered the effect of both the controlled and the manipulated variables within the objective function used for optimization of the tuning parameters.

Similar to chapter 3, the first step taking to assess if it is worthwhile to design a robust GSMPC consists in comparing the $\gamma^{\text{op}-\text{nom}}$ values between the GSMPC and LMPC. Because the $\gamma^{\text{op}-\text{nom}}$ of the GSMPC was less than the $\gamma^{\text{op}-\text{nom}}$ of the LMPC a robust GSMPC controller was designed when uncertainty is considered.

An important thing to mention is that in chapter 4 better models and different corresponding normalizations were used. With these new models the average theoretical improvement of the GSMPC controller over the LMPC was of 16% with the biggest improvement occurring in the zone around an inlet substrate concentration of $S_{\text{in}} = 3.0$ g/L.

The manipulated variables effect was quantified through the term c_e, if $c_e = 0$ it means that the effect of the manipulated variables is not being considered in γ. When the value of c_e increased it was observed that γ also grew and the differences between the analytical γ's for the gain-scheduling and linear controllers decreased. This can be explained by the fact that when c_e is larger the movement of the manipulated variables is restricted resulting in more conservative controllers. This will lead to less sensitivity to model error related to the system nonlinearity and thus less opportunity for improvements by the GSMPC as compared to the LMPC.

The algorithm presented can be used as an alternative for designing an adaptation algorithm in which the controller parameters are changed on-line according to the process behavior. One of the disadvantages of the adaptation algorithm is that it is difficult to address its robustness properties. Since the proposed gain-scheduled MPC does consider uncertainty it represents a much better alternative.

7.2 Volterra series based robust NMPC

The second methodology proposed in this thesis is a nonlinear model predictive control strategy based on a Volterra series model. This model was chosen because it has a structure that can be easily separated into two parts. The first part contains the nominal value of the parameters and the second part contains the uncertainty associated to each parameter. The key advantage obtained by splitting the structure is that a robust test can be proposed based on the SSV also referred to as μ-test.

The basic objective function of the controller minimizes the infinity norm of the difference between the set-points and the predicted outputs. Furthermore, the objective function was augmented to include manipulated variables movements weighting, manipulated variables constraints and terminal conditions.

The predictions were based on an autoregressive Volterra series model for three main reasons: (i) they have less noise sensitivity; (ii) they are easier to identify and (iii) they require less parameters. However, the methodology proposed can also be used with non-autoregressive Volterra series models by setting $h_{q\chi} = 0$, $q = 1, \ldots, n_{\mathrm{ARX}}$, $\chi = 1, \ldots, n_y$ in the interconnection matrix $\mathbf{M}$.

The robustness test proposed is a min-max formulation, where the worst value of the objective function is minimized with respect to the manipulated variables. To test the effectiveness of the controller two case studies were proposed a SISO CSTR and a MIMO pH neutralization process.

7.2.1 SISO case study

The SISO case study was used for several purposes; the first purpose was to check whether the controller can be tuned by using the term $W_i^{\Delta u}$ that accounts for manipulated variable movement penalization. It was observed that as $W_i^{\Delta u}$ grows the controller is slower and if $W_i^{\Delta u}$ decreases the control actions are faster and as a consequence the control inputs reach its set-point faster. Thus, $W_i^{\Delta u}$ can be effectively used to tune the controller speed.

The second case study was considered to test the ability of the algorithm to accommodate manipulated variables constraints. The case study results showed that the proposed algorithm methodology is able to keep the value of the manipulated variables within the allowed bounds.

The third case study was designed to compare the performance of three controllers: a linear MPC, a non-robust NMPC and the proposed robust NMPC. The IAE was used to compare the performance of the different controllers. The results showed that the robust NMPC had the lowest IAE which means that it is a better controller than the other two. This case also showed that when the process nonlinearity is important a linear MPC controller can give a poor performance as compared to a NMPC controller.

A SISO case study was also used to compare the performance of two controllers, a non-robust NMPC and the proposed robust NMPC, for several values of $W_i^{\Delta u}$. The results showed that as $W_i^{\Delta u}$ grows the robust controller performance starts degrading and as a consequence the performance of the non-robust controller becomes superior. This relative performance degradation can be explained by the fact that for large manipulated variables weights the controllers are highly detuned resulting in worst closed-loop performance and less sensitivity to model error. In this case the non-robust controller may perform better than the robust one.

7.2.2 MIMO case study

Based on the information obtained from the SISO case studies the first MIMO case study was designed to compare the performance between a non-robust and the robust NMPC controllers. The case study was designed to study the effect of the manipulated variables weighting terms The results showed a trend similar to the SISO case, i.e., as the weighting decreases the robust controller is less sensitive to model error and its performance is better than the non-robust controller.

The second MIMO case study compared the performance of the proposed RNMPC and a first-principles model based NMPC. The FP-NMPC controller was designed on the basis of a simple nonlinear predictor. Under nominal conditions, i.e., the parameters of the first-principles model match the process parameters; the FP-NMPC was always superior to the RNMPC. However, when there is a mismatch between the first-principles model parameters and the process parameters, the IAE of the FP-NMPC starts degrading. Since the parameters of the RNMPC were identified around several operating regions the Volterra series model is able to capture the process behavior in each region. Thus, there were regions of operations in which the IAE of the RNMPC was lower than the FP-NMPC.

The RNMPC and FP-NMPC were also compared when manipulated variables constraints were considered in the formulation. A barrier function was used in the FP-NMPC to satisfy input constraints. The RNMPC was shown to provide better average performance than the FP-NMPC for small values of the control action weights. Once again, the robust controller was shown to be superior to the non-robust FP-NMPC when the controller is aggressive thus resulting in higher sensitivity to model error.

7.3 Directions for future research

Based on the work developed in this thesis some directions for future research are:

1. The calculation of γ in chapters 3 and 4 requires the value of $\mathbf{u}$ which is calculated iteratively. This step can be simplified by combining the gain-scheduled controller and the switching rules. In this way the $\mathbf{u}$ values will be given directly from the switching rule. With the previous modification the time required to calculate $\mathbf{Q}$ will decrease. This decrease in computational time could be used to extend the methodology to include a mixed integer nonlinear optimization where p, m, $\mathbf{Q}$ and $\mathbf{R}$ are all optimized together to decrease γ.

2. The RNMPC algorithm proposed in chapter 5 is based on a SSV calculation that requires a considerable amount of time in Matlab. Although the SSV is a non-polynomial hard problem it will be highly recommendable for the purpose of real time implementation to develop a program in C or Fortran to decrease the computational time. An additional possibility is to construct a look-up table with different SSV values for different combinations of feedback errors, initial conditions and past manipulated variable moves. In that case the SSV values could be obtained from interpolation using the values in the look-up table.

3. The three main benefits that could be achieved with the development of the C-SSV or Fortran-SSV are:

   (a) The interconnection matrix $\mathbf{M}$ could be modified to consider instead of the $||\cdot||_\infty$ a $||\cdot||_2$.

   (b) p could be increased.

   (c) MIMO systems other than 2×2 could be considered.

4. A potential application of the RNMPC algorithm proposed in this thesis is for pharmaceutical manufacturing processes. These types of processes are generally represented by models which parameters are highly uncertain due to the inherent variability present in cell culture based processes. The pharmaceutical industry is currently interested in assessing the effect of model uncertainty on process control and optimization of cell culture based manufacturing processes. This interest is driven by a number of factors such as assessing the final expected variability in productivity and quality and/or assessing whether it is worthwhile to continue or terminate a lengthy manufacturing process before the normal termination time. It is believed that the algorithms presented in this thesis can be effectively used to accomplish these tasks.

5. The RNMPC controller proposed in this work used Volterra series models for predictions. In principle this methodology could be easily extended to any type of empirical models that is linear with respect to the model coefficients. Thus, nonlinear basis functions different than Volterra series forms such as radial basis functions or others could be also used within the formulation. This could significantly extend the type of nonlinearities that can be represented by the model and could accordingly lead to more compact model representations and less computational demand.

Bibliography

Agachi, P. Ş., Nagy, Z. K., Cristea, M. V., and Imre-Lucaci, A. (2007). *Model based control: Case studies in process engineering*. Wiley.

Badgwell, T. A. (1997). A robust model predictive control algorithm for stable linear plants. In *American Control Conference*, pages 1618–1622.

Bates, D. G. and Postlethwaite, I. (2002). The structured singular value and μ-analysis. In Thoma, M. and Morari, M., editors, *Advanced Techniques for Clearance of Flight Control Laws*, chapter 3, pages 37–55. Springer Berlin / Heidelberg.

Bequette, B. W. (1991). Nonlinear control of chemical processes: A review. *Industrial & Engineering Chemistry Research*, 30(7):1391–1413.

Bequette, B. W. (2003). *Process control : Modeling, design, and simulation*. Prentice Hall PTR, Upper Saddle River, N.J.

Boyd, S. and Chua, L. O. (1985). Fading memory and the problem of approximating nonlinear operators with Volterra series. *IEEE Transactions on Circuits and Systems*, CAS-32(11):1150 – 1161.

Braatz, R. P., Young, P. M., Doyle, J. C., and Morari, M. (1994). Computational complexity of μ calculation. *IEEE Transactions on Automatic Control*, 39(5):1000–1002.

Campo, P. J. and Morari, M. (1987). Robust model predictive control. In *American Control Conference*, pages 1021–1026.

Chen, H. and Allgöwer, F. (1998a). A computationally attractive nonlinear predictive control scheme with guaranteed stability for stable systems. *Journal of Process Control*, 8(5–6):475–485.

Chen, H. and Allgöwer, F. (1998b). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1217.

Cutler, C. R. (1983). *Dynamic matrix control: An optimal multivariable control algorithm with constraints*. PhD thesis, University of Houston.

Cutler, C. R. and Ramaker, B. L. (1980). Dynamic Matrix Control – A Computer Control Algorithm. In *American Control Conference*.

Cuzzola, F. A., Geromel, J. C., and Morari, M. (2002). An improved approach for constrained robust model predictive control. *Automatica*, 38:1183–1189.

Diaz, H. and Desrochers, A. A. (1988). Modeling of nonlinear discrete-time systems from input-output data. *Automatica*, 24(5):629–641.

Díaz-Mendoza, R. and Budman, H. (2010). Structured singular valued based robust nonlinear model predictive controller using Volterra series models. *Journal of Process Control*, 20(5):653–663.

Diehl, M. and Björnberg, J. (2004). Robust dynamic programming for min-max model predictive control of constrained uncertain systems. *IEEE Transactions on Automatic Control*, 49(12):2253–2257.

Diehl, M., Gerhard, J., Marquardt, W., and Mönnigmann, M. (2008). Numerical solution approaches for robust nonlinear optimal control problems. *Computers & Chemical Engineering*, 32(6):1279–1292.

Doyle, J. (1982). Analysis of feedback systems with structured uncertainties. In *IEE Proceedings D Control Theory Applications. 129*, pages 242–250.

Doyle III, F. J., Kwatra, H. S., and Schwaber, J. S. (1998). Dynamic gain scheduled process control. *Chemical Engineering Science*, 53(15):2675–2690.

Doyle III, F. J., Ogunnaike, B. A., and Pearson, R. K. (1995). Nonlinear model-based control using second-order Volterra models. *Automatica*, 31(5):697–714.

Doyle III, F. J., Packard, A. K., and Morari, M. (1989). Robust controller design for a nonlinear CSTR. *Chemical Engineering Science*, 44(9):1929–1947.

Eaton, J. W. and Rawlings, J. B. (1992). Model-predictive control of chemical processes. *Chemical Engineering Science*, 47(4):705–720.

Findeisen, R. and Allgöwer, F. (2002). An Introduction to Nonlinear Model Predictive Control. In *21 Benelux Meeting on Systems and Control*, volume 1, pages 1–23.

Findeisen, R., Imsland, L., Algöwer, F., and Foss, B. A. (2003). State and output feedback nonlinear model predictive control: An overview. *European Journal of Control*, 9(2–3):179–195.

Fruzzetti, K. P., Palazoğlu, A., and McDonald, K. A. (1997). Nonlinear model predictive control using Hammerstein models. *Journal of Process Control*, 7(1):31–41.

Gao, J. (2004). *Robust control design of gain-scheduled controllers for nonlinear processes*. PhD thesis, University of Waterloo.

Gao, J. and Budman, H. M. (2004). Reducing conservatism in the design of a robust gain-scheduled controller for non-linear chemical processes. *International Journal of Control*, 77(11):1050–1061.

Genceli, H. and Nikolaou, M. (1995). Design of robust constrained model-predictive controllers with Volterra series. *AIChE Journal*, 41(9):2098–2107.

Gerkšič, S., Juričic, D., Strmčnik, S., and Drago, M. (2000). Wiener model based nonlinear predictive control. *International Journal of Systems Science*, 31(2):189–202.

Grimm, G., Messina, M. J., Tuna, S. E., and Teel, A. R. (2007). Nominally robust model predictive control with state constraints. *IEEE Transactions on Automatic Control*, 52(10):1856–1870.

Henson, M. A. (1998). Nonlinear model predictive control: Current status and future directions. *Computers & Chemical Engineering*, 23(2):187–202.

Henson, M. A. and Seborg, D. E. (1994). Adaptive nonlinear control of a pH neutralization process. *IEEE Transactions on Control Systems Technology*, 2(3):169–182.

Hérnandez, E. and Arkun, Y. (1993). Control of nonlinear systems using polynomial ARMA models. *AIChE Journal*, 39(3):446–460.

Hoo, K. A. and Kantor, J. C. (1986). Global linearization and control of a mixed-culture bioreactor with competition and external inhibition. *Mathematical Biosciences*, 82(1):43–62.

Imsland, L., Findeisen, R., Bullinger, E., Algöwer, F., and Foss, B. A. (2003). A note on stability, robustness and performance of output feedback nonlinear model predictive control. *Journal of Process Control*, 13(7):633–644.

Jeong, B.-G., Yoo, K.-Y., and Rhee, H.-K. (2001). Nonlinear model predictive control using a Wiener model of a continuous methyl methacrylate polymerization reactor. *Industrial & Engineering Chemistry Research*, 40(25):5968–5977.

Kawohl, M., Heine, T., and R., K. (2007). A new approach for robust model predictive control of biological production processes. *Chemical Engineering Science*, 62(18-20):5212–5215.

Kothare, M., Balakrishnan, V., and Morari, M. (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10):1361–1379.

Kouvaritakis, B., Rossiter, J. A., and Schuurmans, J. (2000). Efficient robust predictive control. *IEEE Transactions on Automatic Control*, 45(8):1545–1549.

Lee, J. H., Morari, M., and Garcia, C. E. (1994). State-space interpretation of model predictive control. *Automatica*, 30(4):707–717.

Leontaritis, I. J. and Billings, S. A. (1985a). Input-output parametric models for non-linear systems Part I: deterministic non-linear systems. *International Journal of Control*, 41(2):303–328.

Leontaritis, I. J. and Billings, S. A. (1985b). Input-output parametric models for non-linear systems Part II: stochastic non-linear systems. *International Journal of Control*, 41(2):329–344.

Löfberg, J. (2004). YALMIP: A toolbox for modelling and optimization in Matlab. In *Proceedings of the IEEE International Symposium on Computer Aided Control Systems Design*, pages 284–289, Taipei, Taiwan.

Luyben, W. L. (2004). Fed-batch reactor temperature control using lag compensation and gain scheduling. *Industrial & Engineering Chemistry Research*, 43(15):4243–4252.

Ma, D. L. and Braatz, R. D. (2001). Worst-case analysis of finite-time control policies. *IEEE Transactions on Control Systems Technology*, 9(5):766–774.

Maciejowski, J. M. (2002). *Predictive control with constraints*. Prentice-Hall.

Magni, L. and Scattolini, R. (2007). Robustness and robust design of MPC for nonlinear discrete-time systems. In Thoma, M. and Morari, M., editors, *Assessment and Future Directions of Nonlinear Model Predictive Control*, chapter 19, pages 239–254. Springer Berlin / Heidelberg.

Maner, B. R. and Doyle III, F. J. (1997). Polymerization reactor control using autoregressive-plus Volterra-based MPC. *AIChE Journal*, 43(7):1763–1784.

Maner, B. R., Doyle III, F. J., Ogunnaike, B. A., and Pearson, R. K. (1996). Nonlinear model predictive control of a simulated multivariable polymerization reactor using second-order Volterra models. *Automatica*, 32(9):1285–1301.

Mayne, D. Q. and Michalska, H. (1990). Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–914.

Morari, M. and Lee, J. H. (1999). Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682.

Morari, M. and Zafiriou, E. (1989). *Robust process control*. Prentice Hall, Englewood Cliffs, N.J. ; Toronto.

Muske, K. R. and Rawlings, J. B. (1993a). Linear model predictive control of unstable processes. *Journal of Process Control*, 3(2):85–96.

Muske, K. R. and Rawlings, J. B. (1993b). Model predictive control with linear models. *AIChE Journal*, 39(2):262–287.

Nagy, Z. K. and Braatz, R. D. (2003). Worst-case and distributional robustness analysis of finite-time control trajectories for nonlinear distributed parameter systems. *IEEE Transactions on Control Systems Technology*, 11(5):694–704.

Nahas, E. P., Henson, M. A., and Seborg, D. E. (1992). Nonlinear internal model control strategy for neural network models. *Computers & Chemical Engineering*, 16(12):1039–1057.

Nikolaou, M. and Misra, P. (2003). Linear control of nonlinear processes: Recent developments and future directions. *Computers & Chemical Enginnering*, 27(8–9):1043–1059.

Norquay, S. J., Palazoğlu, A., and Romagnoli, J. A. (1999). Application of Wiener model predictive control (WMPC) to a pH neutralization experiment. *IEEE Transactions on Control Systems Technology*, 7(40):437–445.

Nowak, R. D. and Van Veen, B. D. (1994). Random and pseudorandom inputs for Volterra filter identification. *IEEE Transactions on Signal Processing*, 42(8):2124–2135.

Nygaard, G. and Nævdal, G. (2006). Nonlinear model predictive control scheme for stabilizing annulus pressure during oil well drilling. *Journal of Process Control*, 16(7):719–732.

Packard, A. and Doyle, J. (1993). The complex structured singular value. *Automatica*, 29(1):71–109.

Parker, R. S. and Doyle III, F. J. (2001). Optimal control of a continuous bioreactor using an empirical nonlinear model. *Industrial & Engineering Chemistry Research*, 40(8):1939–1951.

Parker, R. S., Heemstra, D., Doyle III, F. J., Pearson, R., and Ogunnaike, B. A. (2001). The identification of nonlinear models for process control using tailored "plant friendly" input sequences. *Journal of Process Control*, 11(2):237–250.

Peng, H., Yang, Z.-J., Gui, W., Wu, M., Shioya, H., and Nakano, K. (2007). Nonlinear system modeling and robust predictive control based on RBF-ARX model. *Engineering Applications of Artificial Intelligence*, 20:1–9.

Qin, S. J. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764.

Rangaiah, G. P., Saha, P., and Tadé, M. O. (2002). Nonlinear model predictive control of an industrial four-stage evaporator system via simulation. *Chemical Engineering Journal*, 87(3):285–299.

Rawlings, J. B. and Muske, K. R. (1993). The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, AC-38:1512–1516.

Ricardez, S. L. A., Budman, H. M., and Douglas, P. L. (2008). Simultaneous design and control of processes under uncertainty: A robust modeling approach. *Journal of Process Control*, 18(7–8):735–752.

Rugh, W. J. (1991). Analytical framework for gain scheduling. *IEEE Control Systems Magazine*, 11(1):79–84.

Rugh, W. J. and Shamma, J. S. (2000). Research on gain scheduling. *Automatica*, 36(10):1401–1425.

Santos, L. O., Afonso, P. A. F. N. A., Castro, J. A. A. M., Oliveira, N. M. C., and Biegler, L. T. (2001). On-line implementation of nonlinear MPC: An experimental case study. *Control Engineering Practice*, 9(8):847–857.

Santos, L. O. and Biegler, L. T. (1999). A tool to analyze robust stability for model predictive controllers. *Journal of Process Control*, 9(3):233–246.

Schetzen, M. (1980). *The Volterra and Wiener theories of nonlinear systems*. Robert E. Krieger Publishing Company.

Shamma, J. S. and Athans, M. (1990). Analysis of gain scheduled control for nonlinear plants. *IEEE Transactions on Automatic Control*, 35(8):898–907.

Sheweickhardt, T. and Allgöwer, F. (2007). Linear control on nonlinear systems based on nonlinearity measures. *Journal of Process Control*, 17(3):273–284.

Sistu, P. B. and Bequette, W. (1991). Nonlinear predictive control of uncertain processes: Application to a CSTR. *AIChE Journal*, 37(11):1711–1723.

Soni, A. S. and Parker, R. S. (2007). Tailored sequence design for third-order Volterra model identification. *Industrial & Engineering Chemistry Research*, 46(3):818–829.

Sontag, E. D. (1979). Realization theory of discrete-time nonlinear systems: Part I-The bounded case. *IEEE Transactions on Circuits and Systems*, CAS-26(4):342–356.

Sturm, J. F. (1999). Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1):625–653.

Toh, K. C., Todd, M. J., and Tütüncü, R. H. (1999). SDPT3 – A Matlab software package for semidefinite programming version 1.3. *Optimization Methods and Software*, 11(1):545–581.

Uppal, A., Ray, W. H., and Poore, A. B. (1974). On the dynamic behavior of continuous stirred tank reactors. *Chemical Engineering Science*, 29(4):967–985.

van den Boom, T. J. J. (1997). Robust nonlinear predictive control using feedback linearization and linear matrix inequalities. In *American Control Conference*, pages 3068–3072.

Wan, Z. and Kothare, M. V. (2002). Robust output feedback model predictive control using off-line linear matrix inequalities. *Journal of Process Control*, 12(7):763–774.

Wang, Y. J. and Rawlings, J. B. (2004a). A new robust model predictive control method I: Theory and computation. *Journal of Process Control*, 14(3):231–247.

Wang, Y. J. and Rawlings, J. B. (2004b). A new robust model predictive control method II: Examples. *Journal of Process Control*, 14(3):249–262.

Wright, G. T. and Edgar, T. F. (1994). Nonlinear model predictive control of a fixed-bed water-gas shift reactor: An experimental study. *Computers & Chemical Engineering*, 18(2):83–102.

Zafiriou, E. (1990). Robust model predictive control of processes with hard constraints. *Computers & Chemical Engineering*, 14(4–5):359–371.

Zanovello, R. and Budman, H. (1999). Model predictive control with soft constraints with application to lime kiln control. *Computers & Chemical Engineering*, 23(6):791–806.

Zavala, V. M. and Biegler, L. T. (2009). The advanced-step NMPC controller: Optimality, stability and robustness. *Automatica*, 45(1):86–93.

Zheng, A. (1999). Robust stability analysis of constrained model predictive control. *Journal of Process Control*, 9(4):271–278.

Zheng, Z. Q. and Morari, M. (1993). Robust stability of constrained model predictive control. In *American Control Conference*, pages 379–383.

Appendix A

State-Affine model parameters for chapter 3

Table A.1: Value of the matrices $\mathbf{F}_0$, $\mathbf{F}_1$ and $\mathbf{F}_2$

S^{in}	$\mathbf{F}_0$	$\mathbf{F}_1$	$\mathbf{F}_2$
2.0	$\begin{bmatrix} +0.8663 & -0.1439 \\ -0.0256 & +0.9420 \end{bmatrix}$	$\begin{bmatrix} +0.0503 & 0 \\ +0.0189 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & +0.0099 \\ 0 & -0.0011 \end{bmatrix}$
2.5	$\begin{bmatrix} +0.9471 & -0.0578 \\ +0.0052 & +0.9773 \end{bmatrix}$	$\begin{bmatrix} +0.0085 & 0 \\ +0.0015 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & +0.0010 \\ 0 & -0.0050 \end{bmatrix}$
3.0	$\begin{bmatrix} +0.9727 & -0.0312 \\ +0.0150 & +0.9879 \end{bmatrix}$	$\begin{bmatrix} -0.0113 & 0 \\ -0.0069 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.0006 \\ 0 & -0.0058 \end{bmatrix}$
4.0	$\begin{bmatrix} +0.9879 & -0.0132 \\ +0.0208 & +0.9949 \end{bmatrix}$	$\begin{bmatrix} -0.0222 & 0 \\ -0.0137 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.0009 \\ 0 & -0.0059 \end{bmatrix}$

Table A.2: Value of the matrices $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$

S^{in}	$\mathbf{G}_1$	$\mathbf{G}_2$	$\mathbf{G}_3$
2.0	$\begin{bmatrix} -0.1061 & -0.0008 \\ -0.0336 & -0.0073 \end{bmatrix}$	$\begin{bmatrix} +0.0085 & 0 \\ +0.0034 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.0020 \\ 0 & -0.0007 \end{bmatrix}$
2.5	$\begin{bmatrix} -0.1074 & -0.0005 \\ -0.0341 & -0.0073 \end{bmatrix}$	$\begin{bmatrix} +0.0048 & 0 \\ +0.0017 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.0038 \\ 0 & -0.0018 \end{bmatrix}$
3.0	$\begin{bmatrix} -0.1074 & -0.0004 \\ -0.0340 & -0.0072 \end{bmatrix}$	$\begin{bmatrix} +0.0016 & 0 \\ -0.0006 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.0033 \\ 0 & -0.0018 \end{bmatrix}$
4.0	$\begin{bmatrix} -0.1078 & -0.0002 \\ -0.0340 & -0.0072 \end{bmatrix}$	$\begin{bmatrix} +0.0017 & 0 \\ +0.0014 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & +0.0013 \\ 0 & -0.0038 \end{bmatrix}$

Table A.3: Value of the matrix $\mathbf{W}_F$

	$S^{\text{in}} = 2.0$	$S^{\text{in}} = 2.5$	$S^{\text{in}} = 3.0$	$S^{\text{in}} = 4.0$
$\mathbf{W}_F$	$\begin{bmatrix} +1.7693 \\ +0.8209 \end{bmatrix}$	$\begin{bmatrix} +1.5069 \\ +0.4546 \end{bmatrix}$	$\begin{bmatrix} +0.9700 \\ -0.0191 \end{bmatrix}$	$\begin{bmatrix} +0.2242 \\ -0.1508 \end{bmatrix}$

Appendix B

State-Affine model parameters for chapter 4

Table B.1: Value of the matrices $\mathbf{F}_0$, $\mathbf{F}_1$ and $\mathbf{F}_2$

S^{in}	$\mathbf{F}_0$	$\mathbf{F}_1$	$\mathbf{F}_2$
2.0	$\begin{bmatrix} +0.8697 & -0.0698 \\ -0.0262 & +0.9538 \end{bmatrix}$	$\begin{bmatrix} +0.0020 & 0 \\ +0.0002 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & +0.0007 \\ 0 & -0.0002 \end{bmatrix}$
2.5	$\begin{bmatrix} +0.9575 & -0.0285 \\ +0.0170 & +0.9789 \end{bmatrix}$	$\begin{bmatrix} +0.0018 & 0 \\ -0.0043 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & +0.0006 \\ 0 & -0.0009 \end{bmatrix}$
3.0	$\begin{bmatrix} +0.9755 & -0.0166 \\ +0.0327 & +0.9867 \end{bmatrix}$	$\begin{bmatrix} +0.0010 & 0 \\ -0.0080 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & +0.0001 \\ 0 & -0.0010 \end{bmatrix}$

Table B.2: Value of the matrices $\mathbf{G}_1$, $\mathbf{G}_2$ and $\mathbf{G}_3$

S^{in}	$\mathbf{G}_1$	$\mathbf{G}_2$	$\mathbf{G}_3$
2.0	$\begin{bmatrix} -0.0235 & -0.0000 \\ -0.0130 & -0.0027 \end{bmatrix}$	$\begin{bmatrix} -0.0018 & 0 \\ -0.0013 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & +0.0000 \\ 0 & -0.0001 \end{bmatrix}$
2.5	$\begin{bmatrix} -0.0231 & 0.0000 \\ -0.0142 & -0.0030 \end{bmatrix}$	$\begin{bmatrix} -0.0032 & 0 \\ -0.0026 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.0001 \\ 0 & -0.0009 \end{bmatrix}$
3.0	$\begin{bmatrix} -0.0252 & -0.0001 \\ -0.0163 & -0.0041 \end{bmatrix}$	$\begin{bmatrix} -0.0049 & 0 \\ -0.0034 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -0.0003 \\ 0 & -0.0030 \end{bmatrix}$

Table B.3: Value of the matrix $\mathbf{W}_F$

	$S^{\text{in}} = 2.0$	$S^{\text{in}} = 2.5$	$S^{\text{in}} = 3.0$
$\mathbf{W}_F$	$\begin{bmatrix} 0.0100 \\ 0.4044 \end{bmatrix}$	$\begin{bmatrix} 0.0104 \\ 0.4082 \end{bmatrix}$	$\begin{bmatrix} 0.0109 \\ 0.4137 \end{bmatrix}$

Appendix C

Additional information for the interconnection matrix

The robust controller presented in chapters 5 and 6 requires an appropriate interconnection matrix and uncertainty description. There are two possible options to build **M**, in order to show the two options and its benefits and disadvantages a simple example is presented for a system represented by the following equation:

$$\hat{y} = h_{11}^{\mathrm{NL}} u^2 + h_1^{\mathrm{L}} u + d \tag{C.1}$$

The first option was used in chapters 5 and 6 and it explicitly puts the square power of **u** in the structure of **M**, i.e.,

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & k_{ssv} \\ 0 & 0 & 0 & k_{ssv} u \\ 0 & 0 & 0 & k_{ssv} u^2 \\ d & h_1^{\mathrm{L}} & h_{11}^{\mathrm{NL}} & 0 \end{bmatrix} \tag{C.2}$$

the interrelationship between the inputs and outputs can be represented by the following equations:

$$\begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ a(4) \end{bmatrix} = \mathbf{M} \begin{bmatrix} b(1) \\ b(2) \\ b(3) \\ b(4) \end{bmatrix} \tag{C.3}$$

$$\begin{bmatrix} b(1) \\ b(2) \\ b(3) \end{bmatrix} = \begin{bmatrix} \delta_1 & 0 & 0 \\ 0 & \delta_2 & 0 \\ 0 & 0 & \delta_3 \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \\ a(3) \end{bmatrix} \tag{C.4}$$

After performing the operations indicated the following result is obtained:

$$\frac{a(4)}{b(4)} = k_{ssv}\delta_1 \left(h_{11}^{\mathrm{NL}} u^2 + h_1^{\mathrm{L}} u + d\right) \tag{C.5}$$

The second options distributes the square powers of $\mathbf{u}$ in the matrix $\mathbf{M}$, i.e.,

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & k_{ssv} \\ 0 & 0 & 0 & 0 & k_{ssv} \\ 0 & 0 & 0 & 0 & k_{ssv}u \\ 0 & 0 & k_{ssv}u & 0 & 0 \\ d & h_1^{\mathrm{L}} & 0 & h_{11}^{\mathrm{L}} & 0 \end{bmatrix} \tag{C.6}$$

the interrelationship between the inputs and outputs can be represented by the following equations:

$$\begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ a(4) \\ a(5) \end{bmatrix} = \mathbf{M} \begin{bmatrix} b(1) \\ b(2) \\ b(3) \\ b(4) \\ b(5) \end{bmatrix} \tag{C.7}$$

$$\begin{bmatrix} b(1) \\ b(2) \\ b(3) \\ b(4) \end{bmatrix} = \begin{bmatrix} \delta_1 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & 0 \\ 0 & 0 & \delta_1 & 0 \\ 0 & 0 & 0 & \delta_1 \end{bmatrix} \begin{bmatrix} a(1) \\ a(2) \\ a(3) \\ a(4) \end{bmatrix} \tag{C.8}$$

and after performing the operations indicated the following result is obtained:

$$\frac{a(5)}{b(5)} = k_{ssv}\delta_1 \left(k_{ssv}\delta_1 h_{11}^{\mathrm{NL}} u^2 + h_1^{\mathrm{L}} u + d\right) \tag{C.9}$$

The disadvantages of option two are the following:

1. The dimensions of $\mathbf{M}$ increase in comparison with the first option. This represents a major computational problem since the time required to calculate μ increase with the dimensions of $\mathbf{M}$.

2. The product $k_{ssv}\delta_1$ can be equal to $+1$ or -1. It can be seen form equation (C.9) that if $k_{ssv}\delta_1 = -1$ the algorithm is finding the worst case for a different system, i.e., the algorithm instead of finding the worst case for the system represented by equation (C.1) is finding the worst case the following system:

$$\hat{y} = h_{11}^{\mathrm{NL}} u^2 - h_1^{\mathrm{L}} u - d \tag{C.10}$$

The interconnection matrix constructed according to the rules of option 1 does not suffer from the previous disadvantages.

Appendix D

Interconnection matrix construction

The interconnection matrix $\mathbf{M}$ is as follows:

$$\mathbf{M} = \left[\begin{array}{c|c} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \hline \mathbf{M}_{21} & \mathbf{M}_{22} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{M}_{\mathrm{A}} & \mathbf{M}_{\mathrm{B}} \\ \hline \mathbf{M}_{\mathrm{C}} & \mathbf{M}_{\mathrm{D}} \end{array}\right] \tag{D.1}$$

the matrices $\mathbf{M}_{\mathrm{A}}$ and $\mathbf{M}_{\mathrm{D}}$ are matrices of appropriate dimensions with all elements equal to zero. The notation $\mathbf{0}_{(i\times j)}$ refers heretofore to a matrix of i rows and j columns that have all of its elements equal to zero. If i and j are not specified then $\mathbf{0}$ refers to a matrix of appropriate dimensions that have all of its elements equal to zero. $\mathbf{M}_{\mathrm{B}}$ is constructed as follows:

$$\mathbf{M}_{\mathrm{B}} = \left[\mathbf{M}_{\mathrm{B1}}, \mathbf{M}_{\mathrm{B2}}\right]^{\mathrm{T}} \tag{D.2}$$

$$\mathbf{M}_{\mathrm{B1}} = \mathrm{diag}\left[\mathbf{M}_{\mathrm{B1A}}, \mathbf{M}_{\mathrm{B1B}}\right] \tag{D.3}$$

$$\mathbf{M}_{\mathrm{B1A}} = k_{ssv}\mathrm{diag}\left[\left(\mathbf{I}_p\right)_1, \ldots, \left(\mathbf{I}_p\right)_{n_y}\right] \tag{D.4}$$

$$\mathbf{M}_{\mathrm{B1B}} = k_{ssv}\mathrm{diag}\left[\mathbf{M}_{\mathrm{B1BA}}, \mathbf{M}_{\mathrm{B1BB}}\right] \tag{D.5}$$

$$\mathbf{M}_{\mathrm{B1BA}} = \left[\mathbf{M}_{\mathrm{B1BA},1}, \ldots, \mathbf{M}_{\mathrm{B1BA},n_u}\right] \tag{D.6}$$

$$\mathbf{M}_{\mathrm{B1BA},q} = \mathrm{diag}\left[\left[\begin{array}{c} u_q(k-1) \\ u_q(k) \end{array}\right], \ldots, \left[\begin{array}{c} u_q(k+m-1) \\ u_q(k+m) \end{array}\right]\right]_{q=1,\ldots,n_u} \tag{D.7}$$

$$\mathbf{B}_{\mathrm{B1BB}} = \left[\mathrm{diag}\left[u_1(k), \ldots, u_1(k+m)\right], \ldots, \mathrm{diag}\left[u_{n_u}(k), \ldots, u_{n_u}(k+m)\right]\right] \tag{D.8}$$

$$\mathbf{M}_{\mathrm{B2}} = k_{ssv}\left[\left[\begin{array}{c} \mathbf{U}^{\mathrm{L}} \\ \mathbf{U}^{\mathrm{NL}} \end{array}\right], \mathbf{0}\right] \tag{D.9}$$

$$\mathbf{U}^{\mathrm{L}} = \left[\begin{array}{c} \left[\begin{array}{c} \mathbf{U}_1^{\mathrm{L1}} \\ \mathbf{U}_1^{\mathrm{L2}} \end{array}\right] \\ \vdots \\ \left[\begin{array}{c} \mathbf{U}_p^{\mathrm{L1}} \\ \mathbf{U}_p^{\mathrm{L2}} \end{array}\right] \end{array}\right] \tag{D.10}$$

$$\begin{aligned} \mathbf{U}_i^{\mathrm{L1}} =& \mathrm{diag}\Big[\left[\mathbf{0}_{((p+1-i)\times(i-1))}, u_1(i)\,\mathbf{I}_{p+1-i}\right], \ldots, \\ & \left[\mathbf{0}_{((p+1-i)\times((p\times(n_u-1))+(i-1)))}, u_{n_u}(i)\,\mathbf{I}_{p+1-i}\right]\Big]_{i=1,\ldots,p} \end{aligned} \tag{D.11}$$

$$\begin{aligned} \mathbf{U}_i^{\mathrm{L2}} =& \mathrm{diag}\Big[\left[\mathbf{0}_{((p+1-i)\times(i-1))}, (u_1(i))^2\,\mathbf{I}_{p+1-i}\right], \ldots, \\ & \left[\mathbf{0}_{((p+1-i)\times((p\times(n_u-1))+(i-1)))}, (u_{n_u}(i))^2\,\mathbf{I}_{p+1-i}\right]\Big]_{i=1,\ldots,p} \end{aligned} \tag{D.12}$$

$$\mathbf{U}^{\mathrm{NL}} = \left[\mathbf{U}_{1,1}^{\mathrm{NL}}, \mathbf{U}_{1,2}^{\mathrm{NL}}, \ldots, \mathbf{U}_{ii,jj-1}^{\mathrm{NL}}, \mathbf{U}_{ii,jj}^{\mathrm{NL}}\right]_{ii=1,\ldots,p-1;\, jj=1,\ldots,p-ii} \tag{D.13}$$

$$\mathbf{U}_{ii,jj}^{\mathrm{NL}} = \mathrm{diag}\left[\mathbf{U}_{ii,jj,1}^{\mathrm{NL}}, \ldots, \mathbf{U}_{ii,jj,n_u}^{\mathrm{NL}}\right]_{ii=1,\ldots,p;\, jj=1,\ldots,p-ii} \tag{D.14}$$

$$\begin{aligned}\mathbf{U}^{\mathrm{NL}}_{ii,jj,q} = \big[&\mathbf{0}_{((p+1-ii-jj)\times(p\times(n_u-1)+(ii+jj-1)))},\\ &u_q\,(ii+jj)\,u_q\,(jj)\,\mathbf{I}_{p+1-ii-jj}\big]_{ii=1,\ldots,p;\,jj=1,\ldots,p;\,q=1,\ldots,n_u}\end{aligned} \tag{D.15}$$

$\mathbf{M}_{\mathrm{C}}$ is constructed according to the following equations:

$$\mathbf{M}_{\mathrm{C}} = \begin{bmatrix} \mathbf{M}_{\mathrm{C1}} \\ \mathbf{M}_{\mathrm{C2}} \end{bmatrix} \tag{D.16}$$

$$\begin{bmatrix} \mathbf{M}_{\mathrm{C1}} \\ \mathbf{M}_{\mathrm{C2}} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{\mathrm{C1A}}, \mathbf{M}_{\mathrm{C1B}}, \mathbf{M}_{\mathrm{C1C}}, \mathbf{M}_{\mathrm{C1D}}, \mathbf{M}_{\mathrm{C1E}} \\ \mathbf{M}_{\mathrm{C2A}}, \mathbf{M}_{\mathrm{C2B}}, \mathbf{M}_{\mathrm{C2C}}, \mathbf{M}_{\mathrm{C2D}}, \mathbf{M}_{\mathrm{C2E}} \end{bmatrix} \tag{D.17}$$

$$\mathbf{M}_{\mathrm{C2A}} = \begin{bmatrix} [\mathbf{M}_{\mathrm{C2AA1}}, \mathbf{M}_{\mathrm{C2AA2}}] & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_{\mathrm{C2AB}} \end{bmatrix} \tag{D.18}$$

$$\mathbf{M}_{\mathrm{C2AA1}} = \mathrm{diag}\left[d_1\mathbf{I}_p, \ldots, d_{n_y}\mathbf{I}_p\right] \tag{D.19}$$

$$\mathbf{M}_{\mathrm{C2AA2}} = \mathrm{diag}\left[ic_1\mathbf{I}_p, \ldots, ic_{n_y}\mathbf{I}_p\right] \tag{D.20}$$

$$\mathbf{M}_{\mathrm{C2AB}} = \mathrm{diag}\left[\mathbf{M}_{\mathrm{C2AB1}}, \mathbf{M}_{\mathrm{C2AB2}}\right] \tag{D.21}$$

$$\mathbf{M}_{\mathrm{C2AB1}} = \mathrm{diag}\left[\mathbf{M}_{\mathrm{C2AB1},1}, \ldots, \mathbf{M}_{\mathrm{C2AB1},n_u}\right] \tag{D.22}$$

$$\mathbf{M}_{\mathrm{C2AB1},q} = \mathrm{diag}\left[\left[W_1^{\Delta u,q}, -W_1^{\Delta u,q}\right], \ldots, \left[W_m^{\Delta u,q}, -W_m^{\Delta u,q}\right]\right]_{q=1,\ldots,n_u} \tag{D.23}$$

$$\mathbf{M}_{\mathrm{C2AB2}} = \mathrm{diag}\left[\mathbf{M}_{\mathrm{C2AB2},1}, \ldots, \mathbf{M}_{\mathrm{C2AB2},n_u}\right] \tag{D.24}$$

$$\mathbf{M}_{\mathrm{C2AB2},q} = \mathbf{I}_m \left(\frac{k_{ssv}}{u_q^{\mathrm{bound}}} \right)_{q=1,\ldots,n_u} \tag{D.25}$$

$$\mathbf{M}_{\mathrm{C2B}} = \begin{bmatrix} [\mathbf{M}_{\mathrm{C2B},1}, \ldots, \mathbf{M}_{\mathrm{C2B},n_u}] \\ 0 \end{bmatrix} \tag{D.26}$$

$$\mathbf{M}_{\mathrm{C2B},q} = \begin{bmatrix} \mathrm{diag}\left[\mathbf{M}_{\mathrm{C2BL},1,1,q}, \ldots, \mathbf{M}_{\mathrm{C2BL},1,n_y,q}\right], \mathrm{diag}\left[\mathbf{M}_{\mathrm{C2BNL},1,1,q}, \ldots, \mathbf{M}_{\mathrm{C2BNL},1,n_y,q}\right] \\ \vdots \\ \mathrm{diag}\left[\mathbf{M}_{\mathrm{C2BL},p,1,q}, \ldots, \mathbf{M}_{\mathrm{C2BL},p,n_y,q}\right], \mathrm{diag}\left[\mathbf{M}_{\mathrm{C2BNL},p,1,q}, \ldots, \mathbf{M}_{\mathrm{C2BNL},p,n_y,q}\right] \end{bmatrix}^{\mathrm{T}}_{q=1,\ldots,n_u} \tag{D.27}$$

if $(p-1) \geq M$, then $\mathbf{M}_{\mathrm{C2BL}}$, $\mathbf{M}_{\mathrm{C2BNL}}$, $\mathbf{H}_{\mathrm{L1}}$ and $\mathbf{H}_{\mathrm{L2}}$ are calculated as follows:

$$\mathbf{M}_{\mathrm{C2BL},i,j,q} = \begin{bmatrix} \mathbf{0}_{((i-1)\times M)} & \mathbf{0}_{((i-1)\times 1)} \\ \mathbf{H}_{\mathrm{L1},i,j,q} & \mathbf{0}_{(M\times 1)} \\ \mathbf{0}_{((M-(i-1))\times M)} & \mathbf{0}_{((M-(i-1))\times 1)} \end{bmatrix}_{j=1,\ldots,n_y;\, q=1,\ldots,n_u} \tag{D.28}$$

$$\mathbf{M}_{\mathrm{C2BNL},i,j,q} = \begin{bmatrix} \mathbf{0}_{((i-1)\times M)} & \mathbf{0}_{((i-1)\times 1)} \\ \mathbf{H}_{\mathrm{L2},i,j,q} & \mathbf{0}_{(M\times 1)} \\ \mathbf{0}_{((M-(i-1))\times M)} & \mathbf{0}_{((M-(i-1))\times 1)} \end{bmatrix}_{j=1,\ldots,n_y;\, q=1,\ldots,n_u} \tag{D.29}$$

$\mathbf{H}_{\mathrm{L1},i,j,q}$ is calculated as follows, if $i < M$

$$\mathbf{H}_{\mathrm{L1},1,j,q} = \mathrm{diag} \begin{bmatrix} \left[h^{\mathrm{NL}}_{(1,M)\,(j,q)} u_q(-M+2), \ldots, h^{\mathrm{NL}}_{(1,2)\,(j,q)} u_q(0), h^{\mathrm{L}}_{(1)\,(j,q)}\right] \\ \vdots \\ \left[h^{\mathrm{NL}}_{(M-1,M)\,(j,q)} u_q(0), h^{\mathrm{L}}_{(M-1)\,(j,q)}\right] \\ h^{\mathrm{L}}_{(M)\,(j,q)} \end{bmatrix}_{j=1,\ldots,n_y;\, q=1,\ldots,n_u} \tag{D.30}$$

$$\mathbf{H}_{\mathrm{L1},2,j,q} = \operatorname{diag}\begin{bmatrix} \left[h^{\mathrm{NL}}_{(1,M)\,(j,q)} u_q\left(-M+3\right), \ldots, h^{\mathrm{NL}}_{(1,3)\,(j,q)} u_q\left(0\right), h^{\mathrm{L}}_{(1)\,(j,q)}\right] \\ \vdots \\ \left[h^{\mathrm{NL}}_{(M-2,M)\,(j,q)} u_q\left(0\right), h^{\mathrm{L}}_{(M-2)\,(j,q)}\right] \\ h^{\mathrm{L}}_{(M-1)\,(j,q)} \\ h^{\mathrm{L}}_{(M)\,(j,q)} \end{bmatrix}_{j=1,\ldots,n_y;\,q=1,\ldots,n_u} \tag{D.31}$$

$$\mathbf{H}_{\mathrm{L1},M-1,j,q} = \operatorname{diag}\left[\,\left[h^{\mathrm{NL}}_{(1,M)\,(j,q)} u_q\left(0\right), h^{\mathrm{L}}_{(1)\,(j,q)}\right], h^{\mathrm{L}}_{(2)\,(j,q)}, \ldots, h^{\mathrm{L}}_{(M)\,(j,q)}\right]_{j=1,\ldots,n_y;\,q=1,\ldots,n_u} \tag{D.32}$$

if $i \geq M$

$$\mathbf{H}_{\mathrm{L1},i,j,q} = \operatorname{diag}\left[h^{\mathrm{L}}_{(1)\,(j,q)}, \ldots, h^{\mathrm{L}}_{(M)\,(j,q)}\right]_{j=1,\ldots,n_y;\,q=1,\ldots,n_u} \tag{D.33}$$

$$\mathbf{H}_{\mathrm{L2},i,j,q} = \operatorname{diag}\left[h^{\mathrm{NL}}_{(1,1)\,(j,q)}, h^{\mathrm{NL}}_{(2,2)\,(j,q)}, \ldots, h^{\mathrm{NL}}_{(M-1,M-1)\,(j,q)}, h^{\mathrm{NL}}_{(M,M)\,(j,q)}\right]_{j=1,\ldots,n_y;\,q=1,\ldots,n_u} \tag{D.34}$$

If $(p-1) < M$, then $\mathbf{M}_{\mathrm{C2BL}}$, $\mathbf{M}_{\mathrm{C2BNL}}$, $\mathbf{H}_{\mathrm{L1}}$ and $\mathbf{H}_{\mathrm{L2}}$ are calculated as follows:

$$\mathbf{M}_{\mathrm{C2BL},i,j,q} = \begin{bmatrix} \mathbf{0}_{(i-1)\times(p+1-i)} & \mathbf{0}_{((ii-1)\times 1)} \\ \mathbf{H}_{\mathrm{L1},i,j,q} & \mathbf{0}_{((p+1-i)\times 1)} \end{bmatrix}_{j=1,\ldots,n_y;\,q=1,\ldots,n_u} \tag{D.35}$$

$$\mathbf{M}_{\mathrm{C2BNL},i,j,q} = \begin{bmatrix} \mathbf{0}_{(i-1)\times(p+1-i)} & \mathbf{0}_{((ii-1)\times 1)} \\ \mathbf{H}_{\mathrm{L2},i,j,q} & \mathbf{0}_{((p+1-i)\times 1)} \end{bmatrix}_{j=1,\ldots,n_y;\,q=1,\ldots,n_u} \tag{D.36}$$

$$\mathbf{H}_{\mathrm{L1},i,j,q} = \operatorname{diag}\left[h^{\mathrm{L}}_{(1)\,(j,q)}, \ldots, h^{\mathrm{L}}_{(p+1-i)\,(j,q)}\right]_{j=1,\ldots,n_y;\,q=1,\ldots,n_u} \tag{D.37}$$

$$\mathbf{H}_{\mathrm{L2},i,j,q} = \mathrm{diag}\left[h^{\mathrm{NL}}_{(1,1)\,(j,q)}, h^{\mathrm{NL}}_{(2,2)\,(j,q)}, \ldots, h^{\mathrm{NL}}_{(p-i,p-i)\,(j,q)}, h^{\mathrm{NL}}_{(p+1-i,p+1-i)\,(j,q)}\right]_{j=1,\ldots,n_y;\, q=1,\ldots,n_u} \tag{D.38}$$

$$\mathbf{M}_{\mathrm{C2C}} = \begin{bmatrix} [\mathbf{M}_{\mathrm{C2CA},1}, \ldots, \mathbf{M}_{\mathrm{C2CA},n_u}] \\ \mathbf{0} \end{bmatrix} \tag{D.39}$$

$$\mathbf{M}_{\mathrm{C2CA},q} = \begin{bmatrix} \mathrm{diag}\left[\mathbf{H}^{\mathrm{NL}}_{(1,1)\,(1,q)}, \ldots, \mathbf{H}^{\mathrm{NL}}_{(1,1)\,(n_y,q)}\right] \\ \mathrm{diag}\left[\mathbf{H}^{\mathrm{NL}}_{(1,2)\,(1,q)}, \ldots, \mathbf{H}^{\mathrm{NL}}_{(1,2)\,(n_y,q)}\right] \\ \vdots \\ \mathrm{diag}\left[\mathbf{H}^{\mathrm{NL}}_{(1,p-1)\,(1,q)}, \ldots, \mathbf{H}^{\mathrm{NL}}_{(1,p-1)\,(n_y,q)}\right] \\ \vdots \\ \mathrm{diag}\left[\mathbf{H}^{\mathrm{NL}}_{(M-1,1)\,(1,q)}, \ldots, \mathbf{H}^{\mathrm{NL}}_{(M-1,1)\,(n_y,q)}\right] \\ \mathrm{diag}\left[\mathbf{H}^{\mathrm{NL}}_{(M-1,2)\,(1,q)}, \ldots, \mathbf{H}^{\mathrm{NL}}_{(M-1,2)\,(n_y,q)}\right] \\ \vdots \\ \mathrm{diag}\left[\mathbf{H}^{\mathrm{NL}}_{(M-1,p-(M-1))\,(1,q)}, \ldots, \mathbf{H}^{\mathrm{NL}}_{(M-1,p-(M-1))\,(1,q)}\right] \end{bmatrix}^{\mathrm{T}} \tag{D.40}$$

if $jj < (M-1)$, then $\mathbf{H}^{\mathrm{NL}}$ is calculated as:

$$\mathbf{H}^{\mathrm{NL}}_{(ii,jj)\,(j,q)} = \begin{bmatrix} \mathbf{0}_{((ii+jj-1)\times(M-ii))} & \mathbf{0}_{((ii+jj-1)\times 1)} \\ \mathrm{diag}\left[h^{\mathrm{NL}}_{(1,1+ii)\,(j,q)}, \ldots, h^{\mathrm{NL}}_{(M-ii,M)\,(j,q)}\right] & \mathbf{0}_{((M-ii)\times 1)} \\ \mathbf{0}_{((p+1-M-jj)\times(M-ii))} & \mathbf{0}_{((p+1-M-jj)\times 1)} \end{bmatrix} \tag{D.41}$$

else if $jj \geq (M-1)$, then $\mathbf{H}^{\mathrm{NL}}$ is calculated as:

$$\mathbf{H}^{\mathrm{NL}}_{(ii,jj)\,(j,q)} = \begin{bmatrix} \mathbf{0}_{((ii+jj-1)\times(M-ii))} & \mathbf{0}_{((ii+jj-1)\times 1)} \\ \mathrm{diag}\left[h^{\mathrm{NL}}_{(1,1+ii)\,(j,q)}, \ldots, h^{\mathrm{NL}}_{(2M-jj-1-ii,2M-jj-1)\,(j,q)}\right] & \mathbf{0}_{((2M-jj-1-ii)\times 1)} \end{bmatrix} \tag{D.42}$$

$$\mathbf{M}_{\mathrm{C2D}} = \begin{bmatrix} \left[\left[\mathbf{M}_{\mathrm{C2D},1,1}, \ldots, \mathbf{M}_{\mathrm{C2D},M,1}\right], \ldots, \left[\mathbf{M}_{\mathrm{C2D},1,n_u}, \ldots, \mathbf{M}_{\mathrm{C2D},M,n_u}\right]\right] \\ \mathbf{0} \end{bmatrix} \tag{D.43}$$

$$\mathbf{M}_{\mathrm{C2D},ii,q} = \begin{bmatrix} \operatorname{diag}\left[\mathbf{M}_{\mathrm{C2DVL},ii,1,q}, \ldots, \mathbf{M}_{\mathrm{C2DVL},ii,n_y,q}\right] \\ \operatorname{diag}\left[\mathbf{M}_{\mathrm{C2DVNL},ii,1,q}, \ldots, \mathbf{M}_{\mathrm{C2DVNL},ii,n_y,q}\right] \end{bmatrix}^{\mathrm{T}}_{ii=1,\ldots,M;\, q=1,\ldots,n_u} \tag{D.44}$$

$$\mathbf{M}_{\mathrm{C2DVL},ii,j,q} = \begin{bmatrix} \mathbf{0}_{((ii-1)\times(p+1-ii))} & \mathbf{0}_{(ii\times(p-ii))} \\ \delta h^{\mathrm{L}}_{(ii,ii)\,(j,q)} \mathbf{I}_{p+1-ii} & h_{\mathrm{ARX}(j)} \delta h^{\mathrm{L}}_{(ii,ii)\,(j,q)} \mathbf{I}_{p-ii} \end{bmatrix}_{ii=1,\ldots,M;\, j=1,\ldots,n_y; q=1,\ldots,n_u} \tag{D.45}$$

$$\mathbf{M}_{\mathrm{C2DVNL},ii,j,q} = \begin{bmatrix} \mathbf{0}_{((ii-1)\times(p+1-ii))} & \mathbf{0}_{(ii\times(p-ii))} \\ \delta h^{\mathrm{NL}}_{(ii)\,(j,q)} \mathbf{I}_{p+1-ii} & h_{\mathrm{ARX}(j)} \delta h^{\mathrm{NL}}_{(ii)\,(j,q)} \mathbf{I}_{p-ii} \end{bmatrix}_{ii=1,\ldots,M;\, j=1,\ldots,n_y; q=1,\ldots,n_u} \tag{D.46}$$

$$\mathbf{M}_{\mathrm{C2E}} = \begin{bmatrix} \operatorname{diag}\left[\mathbf{M}_{\mathrm{C2E},1,1}, \ldots, \mathbf{M}_{\mathrm{C2E},n_y,1}\right], \ldots, \operatorname{diag}\left[\mathbf{M}_{\mathrm{C2E},1,n_u}, \ldots, \mathbf{M}_{\mathrm{C2E},n_y,n_u}\right] \\ \mathbf{0} \end{bmatrix} \tag{D.47}$$

$$\mathbf{M}_{\mathrm{C2E},j,q} = \begin{bmatrix} \mathbf{H}^{\mathrm{VNL}}_{(1,2)\,(j,q)} \\ \mathbf{H}^{\mathrm{VNL}}_{(2,3)\,(j,q)} \\ \vdots \\ \mathbf{H}^{\mathrm{VNL}}_{(M-1,M)\,(j,q)} \\ \mathbf{H}^{\mathrm{VNL}}_{(1,3)\,(j,q)} \\ \mathbf{H}^{\mathrm{VNL}}_{(2,4)\,(j,q)} \\ \vdots \\ \mathbf{H}^{\mathrm{VNL}}_{(M-2,M)\,(j,q)} \\ \vdots \\ \mathbf{H}^{\mathrm{VNL}}_{(1,M)\,(j,q)} \end{bmatrix}^{\mathrm{T}}_{j=1,\ldots,n_y;\, q=1,\ldots,n_u} \tag{D.48}$$

$$\mathbf{H}^{\mathrm{VNL}}_{(ii,jj)\,(j,q)} = \left[\begin{array}{cc} \mathbf{0}_{((jj-1)\times(p+1-jj))} & \mathbf{0}_{(jj\times(p-jj))} \\ \delta h^{\mathrm{NL}}_{(ii,jj)\,(j,q)}\mathbf{I}_{p+1-jj} & h_{ARX(j)}\delta h^{\mathrm{NL}}_{(ii,jj)\,(j,q)}\mathbf{I}_{p-jj} \end{array}\right]_{j=1,\ldots,n_y;q=1,\ldots,n_u} \tag{D.49}$$

The structure of $\mathbf{M}_{\mathrm{C1}}$ is as follows

$$\mathbf{M}_{\mathrm{C1}} = [\mathbf{M}_{\mathrm{C1A}}, \mathbf{M}_{\mathrm{C1B}}, \mathbf{M}_{\mathrm{C1C}}, \mathbf{M}_{\mathrm{C1D}}, \mathbf{M}_{\mathrm{C1E}}] \tag{D.50}$$

$\mathbf{M}_{\mathrm{C1A}}$,$\mathbf{M}_{\mathrm{C1D}}$ and $\mathbf{M}_{\mathrm{C1E}}$ are matrices of zeros of appropriate dimensions

$$\mathbf{M}_{\mathrm{C1CCD}} = [\mathbf{M}_{\mathrm{C1C}}, \mathbf{M}_{\mathrm{C1D}}] \tag{D.51}$$

In order to obtain the matrix $\mathbf{M}_{\mathrm{C1CCD}}$ a column vector is constructed that contains the Volterra series coefficients according to the following structure:

$$\mathbf{VE} = \begin{bmatrix} \mathbf{VE}_{\mathrm{A1}} \\ \mathbf{VE}_{\mathrm{A2}} \\ \mathbf{VE}_{\mathrm{A3}} \end{bmatrix} \tag{D.52}$$

$$\mathbf{VE}_{\mathrm{A1}} = \begin{bmatrix} \left[h^{\mathrm{L}}_{(1)\,(1,1)}, \ldots, h^{\mathrm{L}}_{(M)\,(1,1)}\right]^{\mathrm{T}} \\ \vdots \\ \left[h^{\mathrm{L}}_{(1)\,(n_y,1)}, \ldots, h^{\mathrm{L}}_{(M)\,(n_y,1)}\right]^{\mathrm{T}} \\ \\ \left[h^{\mathrm{L}}_{(1)\,(1,2)}, \ldots, h^{\mathrm{L}}_{(M)\,(1,2)}\right]^{\mathrm{T}} \\ \vdots \\ \left[h^{\mathrm{L}}_{(1)\,(n_y,n_u)}, \ldots, h^{\mathrm{L}}_{(M)\,(n_y,n_u)}\right]^{\mathrm{T}} \end{bmatrix} \tag{D.53}$$

$$\mathbf{VE}_{\mathrm{A2}} = \begin{bmatrix} \left[h^{\mathrm{NL}}_{(1,1)\,(1,1)}, h^{\mathrm{NL}}_{(2,2)\,(1,1)}, \ldots, h^{\mathrm{NL}}_{(M-1,M-1)\,(1,1)}, h^{\mathrm{NL}}_{(M,M)\,(1,1)}\right]^{\mathrm{T}} \\ \vdots \\ \left[h^{\mathrm{NL}}_{(1,1)\,(n_y,1)}, h^{\mathrm{NL}}_{(2,2)\,(n_y,1)}, \ldots, h^{\mathrm{NL}}_{(M-1,M-1)\,(n_y,1)}, h^{\mathrm{NL}}_{(M,M)\,(n_y,1)}\right]^{\mathrm{T}} \\ \\ \left[h^{\mathrm{NL}}_{(1,1)\,(1,2)}, h^{\mathrm{NL}}_{(2,2)\,(1,2)}, \ldots, h^{\mathrm{NL}}_{(M-1,M-1)\,(1,2)}, h^{\mathrm{NL}}_{(M,M)\,(1,2)}\right]^{\mathrm{T}} \\ \vdots \\ \left[h^{\mathrm{NL}}_{(1,1)\,(n_y,n_u)}, h^{\mathrm{NL}}_{(2,2)\,(n_y,n_u)}, \ldots, h^{\mathrm{NL}}_{(M-1,M-1)\,(n_y,n_u)}, h^{\mathrm{NL}}_{(M,M)\,(n_y,n_u)}\right]^{\mathrm{T}} \end{bmatrix} \tag{D.54}$$

$$\mathbf{VE}_{\mathrm{A3}} = \begin{bmatrix} \mathbf{VE}_{\mathrm{A31},(1,2)} \\ \mathbf{VE}_{\mathrm{A31},(2,3)} \\ \vdots \\ \mathbf{VE}_{\mathrm{A31},(M-1,M)} \\ \mathbf{VE}_{\mathrm{A31},(1,3)} \\ \mathbf{VE}_{\mathrm{A31},(2,4)} \\ \vdots \\ \mathbf{VE}_{\mathrm{A31},(M-2,M)} \\ \mathbf{VE}_{\mathrm{A31},(1,4)} \\ \mathbf{VE}_{\mathrm{A31},(2,5)} \\ \vdots \\ \mathbf{VE}_{\mathrm{A31},(M-3,M)} \\ \vdots \\ \mathbf{VE}_{\mathrm{A31},(1,M)} \end{bmatrix} \tag{D.55}$$

$$\mathbf{VE}_{\mathrm{A31},(ii,jj)} = \begin{bmatrix} h^{\mathrm{NL}}_{(ii,jj)\,(1,1)} \\ \vdots \\ h^{\mathrm{NL}}_{(ii,jj)\,(n_y,1)} \\ h^{\mathrm{NL}}_{(ii,jj)\,(1,2)} \\ \vdots \\ h^{\mathrm{NL}}_{(ii,jj)\,(n_y,2)} \\ \vdots \\ h^{\mathrm{NL}}_{(ii,jj)\,(1,n_u)} \\ \vdots \\ h^{\mathrm{NL}}_{(ii,jj)\,(n_y,n_u)} \end{bmatrix} \tag{D.56}$$

After **VE** has been constructed the following code can be used to assign the corresponding values to the vector matrix *index*:

Algorithm 1:

```
1  $k_c = 0$
2  for $ii = 1$ to number of rows of $[\mathbf{VE}]$ do
3      for $ir = 1$ to $n_y \times p$ do
4          for $ic = 1$ to number of columns $[\mathbf{M}_{\mathrm{C2B}}, \mathbf{M}_{\mathrm{C2C}}]$ do
5              $kc = kc + 1$
6              if $[\mathbf{M}_{\mathrm{C2B}}, \mathbf{M}_{\mathrm{C2C}}]_{ir,ic} = \mathbf{VE}_{ii,1}$ then
7                  $index_{k_c,1} = ic$
8              endif
9          endfor
10     endfor
11 endfor
```

The elements of the matrix **MC1CCD** are zero except for the following terms:

Algorithm 2:

```
1 for $ir = 1$ to $k_c$ do
2     $\mathbf{M}_{\mathrm{C1CCD}(ir,index(k_c,1))} = k_{ssv}$
3 endfor
```

The effect of the autoregressive terms is included in the interconnection matrix by multiplying the elements in the main diagonal of $\mathbf{M}_{\mathrm{C2AA2}}$ by:

Algorithm 3:

```
1 for $i = 1$ to $n_y$ do
2     for $ii = ((i-1) \times p) + 2$ to $(i \times p)$ do
3         $ir = (nrMA \times (ii-1)) \times ((p \times n_y) + 1 : ncM)$
4         $ir = ((p \times n_y) + 1 : ncM) \times (nrMA \times (ii-1))$
5         $\mathbf{M}_{\mathrm{C2AAC2}(ii,ii)} = \left(\frac{1}{k_{ssv}}\right) h_{\mathrm{ARX}(i)} \mathbf{M}_{ir} \mathbf{M}_{ic}$
6     endfor
7 endfor
```

where $nrMA$ is equal to the number of rows of the $\mathbf{M}_{\mathrm{A}}$ matrix and ncM is equal to the number of columns of the $\mathbf{M}$ matrix.

After the interconnection matrix has been constructed the terminal condition is accounted for by multiplying all the columns of the $p \times i, \forall i \in [1, n_y]$ row of $\mathbf{M}_{\mathrm{C2}}$ by (k_{ssv}/ϵ_i) , $\forall i \in [1, n_y]$.

$$\mathbf{M}_{\mathrm{C2}((p \times i),:)} = \frac{k_{ssv}}{\epsilon_i} \mathbf{M}_{\mathrm{C2}((p \times i),:)} \text{ for } i = 1, \ldots, n_y \tag{D.57}$$

The uncertainty block $\boldsymbol{\Delta}$ is composed of $nb_{\boldsymbol{\Delta}}$ different $\boldsymbol{\Delta}$ sub-blocks where

$$nb_{\boldsymbol{\Delta}} = 2 + 2Mn_yn_u + n_yn_u \left(\sum_{j=1}^{M-1} \sum_{i=1}^{M-j} (i) \right) \tag{D.58}$$

the $\boldsymbol{\Delta}$ sub-blocks are arranged according to the following structure $\boldsymbol{\Delta} = \mathrm{diag}\ (\boldsymbol{\Delta}_1, \ldots, \boldsymbol{\Delta}_{nb_{\boldsymbol{\Delta}}})$ where the first $nb_{\boldsymbol{\Delta}} - 1$ blocks are real scalar square matrices related to the uncertainty of the Volterra series coefficients. The dimensions of $\boldsymbol{\Delta}_1$ are calculated from the following expression:

$$\mathbf{\Delta}_1 = (\text{number of columns}\,[\mathbf{M}_{\text{C2C}}]) \times (\text{number of columns}\,[\mathbf{M}_{\text{C2C}}]) \tag{D.59}$$

The dimensions of the blocks $\mathbf{\Delta}_j,\ j \in [2, (Mn_yn_u + 1)]$ are calculated with the following code where $nc\mathbf{M}_{\text{C2DVL},ii,j,q}$ is the number of columns of the matrix block $\mathbf{M}_{\text{C2DVL},ii,j,q}$

Algorithm 4:

```
$du = 1$
for $j = 1$ to $n_y$ do
    for $q = 1$ to $n_u$ do
        for $ii = 1$ to $M$ do
            $du = du + 1$
            $\Delta_{du} = nc\mathbf{M}_{\text{C2DVL},ii,j,q} \times nc\mathbf{M}_{\text{C2DVL},ii,j,q}$
        endfor
    endfor
endfor
```

The dimensions of the blocks $\mathbf{\Delta}_j,\ j \in [(Mn_yn_u + 2), (2Mn_yn_u + 1)]$ are calculated with the following code where $nc\mathbf{M}_{\text{C2DVNL},ii,j,q}$ is the number of columns of the matrix block $\mathbf{M}_{\text{C2DVNL},ii,j,q}$

Algorithm 5:

```
du = M n_y n_u + 1
for j = 1 to n_y do
    for q = 1 to n_u do
        for ii = 1 to M do
            du = du + 1
            Δ_du = ncM_{C2DVNL,ii,j,q} × ncM_{C2DVNL,ii,j,q}
        endfor
    endfor
endfor
```

The dimensions of the blocks $\mathbf{\Delta}_j$, $j \in [(2Mn_yn_u+2), (nb_{\mathbf{\Delta}}-1)]$ are calculated with the following code where $nc\mathbf{H}^{\mathrm{VNL}}_{(a,b)\,(j,q)}$ is the number of columns of the matrix block $\mathbf{H}^{\mathrm{VNL}}_{(a,b)\,(j,q)}$

Algorithm 6:

```
du = 2 M n_y n_u + 2
for j = 1 to n_y do
    for q = 1 to n_u do
        for a = 1 to (M − 1) do
            for b = 1 to (M − a) do
                du = du + 1
                Δ_du = (ncH^{VNL}_{(a,b)(j,q)}) × (ncH^{VNL}_{(a,b)(j,q)})
            endfor
        endfor
    endfor
endfor
```

The block $\mathbf{\Delta}_j$, $j = nb_{\mathbf{\Delta}}$ is a complex scalar square matrix related to performance of dimensions $(pn_y + 2n_um) \times (pn_y + 2n_um)$.

Appendix E

Disturbance characteristics for chapter 5

Table E.1: Disturbance characteristics for SISO case study C (k = sampling instant)

k	β	k	β	k	β	k	β
1	0.3000	21	0.3592	41	0.4146	61	0.1656
2	0.3565	22	0.3598	42	0.4195	62	0.2424
3	0.3518	23	0.4195	43	0.4208	63	0.2417
4	0.3678	24	0.4176	44	0.4150	64	0.2349
5	0.3813	25	0.4252	45	0.3007	65	0.2419
6	0.3525	26	0.4195	46	0.2994	66	0.2443
7	0.3594	27	0.4215	47	0.3033	67	0.2423
8	0.3740	28	0.4222	48	0.2999	68	0.2395
9	0.3646	29	0.4264	49	0.3000	69	0.2389
10	0.3512	30	0.4254	50	0.2983	70	0.2395
11	0.3643	31	0.4202	51	0.3007	71	0.2418
12	0.3620	32	0.4203	52	0.3002	72	0.2430
13	0.3600	33	0.4236	53	0.1702	73	0.2364
14	0.3603	34	0.4223	54	0.1836	74	0.2328
15	0.3638	35	0.4172	55	0.1931	75	0.2332
16	0.3687	36	0.4205	56	0.1869	76	0.2331
17	0.3547	37	0.4202	57	0.1731	77	0.2408
18	0.3545	38	0.4173	58	0.1797	78	0.2385
19	0.3535	39	0.4156	59	0.1769	79	0.2404
20	0.3418	40	0.4234	60	0.1755	80	0.2400

Table E.2: Disturbance characteristics for SISO case study D, $\beta(k=1)=0.3$

Case study	$\beta(k), k=2,\dots,80$	Case study	$\beta(k), k=2,\dots,80$
1	0.4125	13	0.2700
2	0.4050	14	0.2625
3	0.3975	15	0.2550
4	0.3900	16	0.2475
5	0.3825	17	0.2400
6	0.3750	18	0.2325
7	0.3675	19	0.2250
8	0.3600	20	0.2175
9	0.3525	21	0.2100
10	0.3450	22	0.2025
11	0.3375	23	0.1950
12	0.3300	24	0.1875

Table E.3: Disturbance characteristics for MIMO case study, $q_2(k=1)=0.5500$

Case study	$q_2(k), k=2,\dots,40$	Case study	$q_2(k), k=2,\dots,40$
1	0.6600	5	0.5225
2	0.6325	6	0.4950
3	0.6050	7	0.4675
4	0.5775	8	0.4400

Appendix F

Disturbance characteristics for chapter 6

Table F.1: Disturbance characteristics for MIMO case study (k = sampling instant)

k	q_2	k	q_2	k	q_2	k	q_2	k	q_2
1	0.5500	21	0.4473	41	0.6906	61	0.4709	81	0.5095
2	0.5500	22	0.4266	42	0.6764	62	0.4996	82	0.4861
3	0.6073	23	0.4126	43	0.6655	63	0.5285	83	0.4649
4	0.6269	24	0.4106	44	0.6580	64	0.5600	84	0.4439
5	0.6481	25	0.4157	45	0.6304	65	0.5844	85	0.4309
6	0.6682	26	0.4119	46	0.6070	66	0.6128	86	0.4208
7	0.6739	27	0.4256	47	0.5797	67	0.6351	87	0.4118
8	0.6896	28	0.4455	48	0.5564	68	0.6513	88	0.4168
9	0.6853	29	0.4614	49	0.5227	69	0.6727	89	0.4125
10	0.6847	30	0.4917	50	0.5071	70	0.6788	90	0.4288
11	0.6773	31	0.5118	51	0.4809	71	0.6901	91	0.4425
12	0.6608	32	0.5330	52	0.4542	72	0.6896	92	0.4620
13	0.6528	33	0.5717	53	0.4410	73	0.6871	93	0.4940
14	0.6225	34	0.5916	54	0.4270	74	0.6745	94	0.5121
15	0.5960	35	0.6212	55	0.4203	75	0.6627	95	0.5435
16	0.5744	36	0.6376	56	0.4113	76	0.6398	96	0.5772
17	0.5377	37	0.6576	57	0.4204	77	0.6215	97	0.5960
18	0.5251	38	0.6772	58	0.4284	78	0.5883	98	0.6240
19	0.4914	39	0.6800	59	0.4354	79	0.5632	98	0.6466
20	0.4677	40	0.6845	60	0.4570	80	0.5387	99	0.6634

Appendix G

Volterra series model parameters for chapter 5

SISO case studies

$h_{\text{ARX}} = 0.4671$

Table G.1: Volterra series model parameters and uncertainty for SISO case study

	$(i,j) = (1,1)$		$(i,j) = (1,1)$
$h_{0\,(i,j)}$	0.2846	$\delta h_{0\,(i,j)}$	0.0368
$h_{1\,(i,j)}$	0.1471	$\delta h_{1\,(i,j)}$	0.0145
$h_{2\,(i,j)}$	-0.0266	$\delta h_{2\,(i,j)}$	0.0053
$h_{0,0\,(i,j)}$	0.0310	$\delta h_{0,0\,(i,j)}$	0.0018
$h_{1,1\,(i,j)}$	-0.4277	$\delta h_{1,1\,(i,j)}$	0.00117
$h_{2,2\,(i,j)}$	0.0280	$\delta h_{2,2\,(i,j)}$	0.0025
$h_{0,1\,(i,j)}$	0.3591	$\delta h_{0,1\,(i,j)}$	0.0075
$h_{0,2\,(i,j)}$	-0.4160	$\delta h_{0,2\,(i,j)}$	0.0073
$h_{1,2\,(i,j)}$	0.3262	$\delta h_{1,2\,(i,j)}$	0.0195

MIMO case study

$h_{\mathrm{ARX1}} = 0.7697$, $h_{\mathrm{ARX}} = 0.7092$

Table G.2: Nominal Volterra series model parameters for MIMO case study

	$(i,j)=(1,1)$	$(i,j)=(1,2)$	$(i,j)=(2,1)$	$(i,j)=(2,2)$
$h_{0\,(i,j)}$	0.0972	0.0959	-0.0930	0.0905
$h_{1\,(i,j)}$	0.0035	-0.0036	0.0134	-0.0131
$h_{2\,(i,j)}$	-0.0019	0.0019	-0.0027	0.0014
$h_{0,0\,(i,j)}$	-0.0024	-0.0030	0.0081	0.0021
$h_{1,1\,(i,j)}$	0.0450	-0.0923	-0.0096	0.0006
$h_{2,2\,(i,j)}$	-0.0024	-0.0036	-0.0110	-0.0159
$h_{0,1\,(i,j)}$	-0.0404	0.0986	0.0112	0.0116
$h_{0,2\,(i,j)}$	0.0403	-0.0988	0.0247	0.0198
$h_{1,2\,(i,j)}$	-0.0398	0.0994	-0.0026	0.0009

Table G.3: Uncertainty associated to the Volterra series model parameters for MIMO case study

	$(i,j)=(1,1)$	$(i,j)=(1,2)$	$(i,j)=(2,1)$	$(i,j)=(2,2)$
$\delta h_{0\,(i,j)}$	0.0010	0.0011	0.0043	0.0038
$\delta h_{1\,(i,j)}$	0.0019	0.0021	0.0049	0.0041
$\delta h_{2\,(i,j)}$	0.0010	0.0011	0.0028	0.0024
$\delta h_{0,0\,(i,j)}$	0.0012	0.0020	0.0035	0.0025
$\delta h_{1,1\,(i,j)}$	0.0032	0.0042	0.0045	0.0005
$\delta h_{2,2\,(i,j)}$	0.0016	0.0016	0.0044	0.0045
$\delta h_{0,1\,(i,j)}$	0.0025	0.0032	0.0042	0.0054
$\delta h_{0,2\,(i,j)}$	0.0022	0.0031	0.0035	0.0064
$\delta h_{1,2\,(i,j)}$	0.0021	0.0032	0.0023	0.0052

Appendix H

Volterra series model parameters for chapter 6

$h_{\mathrm{ARX1}} = 0.8744$, $h_{\mathrm{ARX2}} = 0.7511$

Table H.1: Nominal Volterra series model parameters for MIMO case study

	$(i,j) = (1,1)$	$(i,j) = (1,2)$	$(i,j) = (2,1)$	$(i,j) = (2,2)$
$h_{0\,(i,j)}$	0.0727	0.0778	-0.1630	0.1690
$h_{1\,(i,j)}$	-0.0016	-0.0025	0.0023	-0.0002
$h_{0,0\,(i,j)}$	-0.0052	-0.0165	0.0271	0.0239
$h_{1,1\,(i,j)}$	0.0085	0.0127	0.0146	0.0746
$h_{0,1\,(i,j)}$	-0.0009	0.0068	-0.0141	-0.0715

Table H.2: Uncertainty associanted to the Volterra series model parameters for MIMO case study

	$(i,j) = (1,1)$	$(i,j) = (1,2)$	$(i,j) = (2,1)$	$(i,j) = (2,2)$
$\delta h_{0\,(i,j)}$	0.0078	0.0081	0.0119	0.0130
$\delta h_{1\,(i,j)}$	0.0004	0.0006	0.0030	0.0047
$\delta h_{0,0\,(i,j)}$	0.0008	0.0035	0.0096	0.0064
$\delta h_{1,1\,(i,j)}$	0.0008	0.0036	0.0075	0.0101
$\delta h_{0,1\,(i,j)}$	0.0002	0.0011	0.0073	0.0123

Appendix I

Matlab codes

This Appendix shows the code of the main programs used in this thesis.

Table I.1 shows the programs that were used to calculate the disturbances used in chapters 3 and 4.

Table I.1: List of programs to calculate **d**

Program Name	Description
generate_d.m	Disturbance generator for Chapter 3
generate_d2.m	Disturbance generator for Chapter 4

```
function d = generate_d(S_in,nit,sig_ma)
% generate_d: disturbance generator for chapter 3
% S_in   --> value of S_{in}
% nit    --> number of smapling intervals
% sig_ma --> desired value for the standard deviation of d
a = round(randn(nit,1));
d = S_in + sig_ma*randn(nit,1).*(-1).^a;
```

```
function d = generate_d2(S_in,nit,pr_fc,an_fr,ph_as)
% generate_d: disturbance generator for chapter 3
% S_in   --> value of S_{in}
% nit    --> number of smapling intervals
% pr_fc  --> perturbation factor for the sinusoidal wave
% an_fr  --> desired angular frecuency
% ph_as  --> desired phase angle
a = [1:nit]';
d = S_in + sin(an_fr.*a+ph_as) + ...
    pr_fc*randn(nit,1).*-1.^round(randn(nit,1));
```

Table I.2 shows the programs that were used to calculate the values of γ.

Table I.2: List of programs to calculate γ

Program Name	Description
pesou.m	Internal Program
pesoy.m	Internal Program
mgamota.m	Calculate $\mathbf{\Gamma}$ matrix
mpsiota.m	Calculate $\mathbf{\Psi}$ matrix
mtetota.m	Calculate $\mathbf{\Theta}$ matrix
kmpcss.m	Calculate $\mathrm{K}_{\mathrm{MPC}}$ matrix
calculategamma.m	Generic program to calculate γ
calgama.m	Internal Program
mat_mimomodel.m	Internal Program
biswi_dcp.m	Internal Program
crealmiwi_dcp.m	Solve LMI feasibility problem ($\mathbf{u}$ bounds)
incer_c_bode.m	Internal Program
jalalmi_dcp.m	LMI structure
pbodei.m	Internal Program
crealmi_dcp_esta.m	Solve LMI feasibility problem (stability)
jalalmi_dcp_esta	LMI structure (stability problem)
bis_grande_dcp.m	Internal Program
crealmi_dcp.m	Solve LMI feasibility problem (γ)
incer_c_gen3.m	Internal Problem

```
function pes_u = pesou(m,n_u,weiu)
pes_u = zeros(length(m*n_u),length(m*n_u));
for j=1:n_u
    for i=j:n_u:(n_u*m)-(n_u-j)
        pes_u(i,i) = weiu(1,j);
    end
end
```

```
function pes_y = pesoy(p,n_y,weiy)
pes_y = zeros(length(p*n_y),length(p*n_y));
for j=1:n_y
    for i=j:n_y:(n_y*p)-(n_y-j)
        pes_y(i,i) = weiy(1,j);
    end
end
```

```
function sc2 = mgamota(A,B,C,p,m)
if m>p, error('Incorrect␣control␣horizon'), end
[ro,co] = size(B);  n_u = co;  [ro,co] = size(C);  n_y = ro;
sc2 = zeros(p*n_y,n_u);
for i=0:p-1
    aux = (i*n_y+1):(i+1)*n_y;  au2 = zeros(n_y,n_u);
    for j=0:i
        au2 = au2 + C*A^j*B;
    end
    sc2(aux(1,1):aux(1,length(aux)),1:n_u) = au2;
end
```

```
function sc2 = mpsiota(A,C,p,m)
if m>p, error('Incorrect␣control␣horizon'), end
[ro,co] = size(A);  n_x = co;  [ro,co] = size(C);  n_y = ro;
sc2 = zeros(p*n_y,n_x);
for i=1:p
    i1 = ((i-1)*n_y) + 1;  i2 = i*n_y;  sc2(i1:i2,:) = C*A^i;
end
```

```
function sc2 = mtetota(A,B,C,p,m)
if m>p, error('Incorrect␣control␣horizon'), end
[ro,co] = size(B);  n_u = co;  [ro,co] = size(C);  n_y = ro;
sc2 = zeros(p*n_y,m*n_u);
for i=0:p-1
    aux = (i*n_y+1):(i+1)*n_y;  au2 = zeros(n_y,n_u);
    for j=0:i
        au2 = au2 + C*A^j*B;
    end
    sc2(aux(1,1):aux(1,length(aux)),1:n_u) = au2;
end
for col=1:m-1
    aux1 = (col*n_u+1):(col+1)*n_u;  aux1a = ((col-1)*n_u+1):col*n_u;
    for i=0:p-2
        ii = i+1;
        auxa = (i*n_y+1)     :  (i+1)*n_y;
        auxf = ((i+1)*n_y+1) : ((i+1)+1)*n_y;
        ro = auxf(1,1):auxf(1,length(auxf));
        co = aux1(1,1):aux1(1,length(aux1));
        roa = auxa(1,1):auxa(1,length(auxa));
        coa = aux1a(1,1):aux1a(1,length(aux1a));
        sc2(ro,co) = sc2(roa,coa);
    end
end
```

```
function [mmpc,mmpc2] = kmpcss(B,C,p,m,weiy,weiu,mt)
[ro,co] = size(B);  n_u = co;  [ro,co] = size(C);  n_y = ro;
Q = pesoy(p,n_y,weiy);  R = pesou(m,n_u,weiu);  mI = zeros(n_u,m*n_u);
mI(1:n_u,1:n_u) = eye(n_u);  mmpc = mI*( ( (mt'*Q*mt) + R ) \ (mt'*Q) );
mmpc2 = ( ( (mt'*Q*mt) + R ) \ (mt'*Q) );
```

```
% Generic program to calculate gamma Generic program to calculate gamma
d_pro = % filename with state-affine matrices of the process
d_mod = % filename with state-affine matrices of the model
load (%filename with WF information',...
    'WF','BW','an_fr')
igs.weiy = ones(1,2);
igs.BW = BW;
igs.p  = 200; % prediction horizon
igs.m  = 4;   % control horizon
igs.WF = WF;
igs.tbode = 5; % discretization time
igs.prnt = 0;  igs.tuwi = 10;  igs.tmwi = -0.2;  igs.tugr = 10;
igs.tmgr = -0.2;  igs.pau = 0; %value of cu
%% LMI solver options
op.solver = 'sdpt3'; op.verb = 0; op.warn = 0;
fac.a = 1; fac.b = 1; fac.c = 1;
%% Uncertainty information
lc1 = 0.10; % if nominal case is analyzed lc1 = 0;
lc2 = 0.10; % if nominal case is analyzed lc2 = 0;
qcu(1,1) = uint8(1); % if nominal case is analyzed qcu(1,1) = uint8(0)
qcu(2,1) = uint8(1); % if nominal case is analyzed qcu(2,1) = uint8(0)
```

```
%% calculations performed in parralell
v_fr_an = 10*an_fr;  igs.libode = 0.01*v_fr_an;  igs.lsbode = v_fr_an;
weiu = %value of the weight matrix R
[ga_ma,u1,u2] = cal_gama(d_mod,d_pro,weiu,lc1,lc2,qcu,v_fr_an,igs,op,fac);
```

```
function [re_g,iu1g,iu2g] = cal_gama(d_mod,d_pro,numeritos,...
    lc1,lc2,qcu,v_fr_an,igs,op,fac)
pmod = load(d_mod,'p1');  ppro = load(d_pro,'p1');
[F0_mod F1_mod F2_mod G1_mod] = mat_mimomodel(pmod.p1);
[F0_pro F1_pro F2_pro G1_pro G2_pro G3_pro] = mat_mimomodel(ppro.p1);
clear F1_mod F2_mod
C = eye(2);  igs.n_x = size(F0_mod,1);  igs.n_u = size(G1_mod,1);
igs.n_y = size(C,1);  mpsi = mpsiota(F0_mod,C,igs.p,igs.m);
mgama = mgamota(F0_mod,G1_mod,C,igs.p,igs.m);
mteta = mtetota(F0_mod,G1_mod,C,igs.p,igs.m);
N2 = pcn2(C,igs.p);
%% Parte de las LMI's
ro_number = size(numeritos);  re_g = zeros(ro_number(1,1),10);  tol = 1E-2;
ctdr = 0;  ce = 0;
for i=1:ro_number(1,1)
    ctdr = ctdr + 1;  weiu = numeritos(i,:);
    mpct = kmpcss(G1_mod,C,igs.p,igs.m,igs.weiy,weiu,mteta);
    kc = 0;  iu1g = 1E-5;  iu2g = 1E-5;  salidaciclo = 0;
    pter = zeros(10,2);
    while salidaciclo == ((tol<=cv1(1,1)) & (tol<=cv2(1,1)))
        if igs.prnt == 1
            fprintf('weiu(1)␣=␣%1.2f␣,␣weiu(2)␣=␣%1.2f',weiu(1),...
                weiu(2)) , fprintf('\n')
        end
        kc = kc + 1;
        if qcu(1,1) == 1
            [gw1,ex1,kc1] = biswi_dcp(igs,F0_mod,G1_mod,...
                F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
                mgama,mpsi,mpct,N2,op,fac,iu1g,iu2g,1,lc1,lc2);
        else
            gw1 = 0;  ex1 = 0;  kc1 = 0;  e1  = 0;
        end
        if qcu(2,1) == 1
            [gw2,ex2,kc2] = biswi_dcp(igs,F0_mod,G1_mod,...
                F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
                mgama,mpsi,mpct,N2,op,fac,iu1g,iu2g,2,lc1,lc2);
        else
            gw2 = 0;  ex2 = 0;  kc2 = 0;  e2  = 0;
        end
        Wgama = [gw1(1,1);gw2(1,1)];
        if ( (ex1 == 0 ) && (ex2 == 0))
            if qcu(1,1) == 1
                u1_bode = pbodei(F0_mod,G1_mod,mpsi,mgama,mpct,igs,Wgama,...
                    N2,1,v_fr_an);
                cv1(1,1)= abs((iu1g - 2*u1_bode(1,1))/iu1g);
            else
                u1_bode = 0;  cv1(1,1)= 0;
            end
            if qcu(2,1) == 1
                u2_bode = pbodei(F0_mod,G1_mod,mpsi,mgama,mpct,igs,Wgama,...
                    N2,2,v_fr_an);
                cv2(1,1)= abs((iu2g - 2*u2_bode(1,1))/iu2g);
            else
                u2_bode = 0;  cv2(1,1)= 0;
            end
            if igs.prnt == 1
                fprintf('weiu(1)␣=␣%1.2f␣,␣weiu(2)␣=␣%1.2f',weiu(1),...
                    weiu(2)) , fprintf('\n')
                fprintf('cv1␣=␣%1.2e␣,␣cv2␣=␣%1.2e',cv1,cv2)
                fprintf('␣Wgama1␣=␣%1.4e␣,␣Wgama2␣=␣%1.4e',...
                    Wgama(1,1),Wgama(2,1))
                fprintf('\n')
                fprintf('u1b␣=␣%1.3f␣,␣u2b␣=␣%1.3f␣',u1_bode,u2_bode)
                fprintf('u1g␣=␣%1.3f␣,␣u2g␣=␣%1.3f',2*u1_bode,2*u2_bode')
                fprintf('\n') , fprintf('\n')
            end
            if kc > 15
                cv1 = 0;  cv2 = 0;  ce = 1;
                fprintf('Increase␣iterations␣number␣\n')
            end
            pter(kc,1) = cv1(1,1);  pter(kc,2) = cv2(1,1);
            if kc>=3
                if (( abs(pter(kc,1)-pter(kc-2,1)) <= 1E-5) && ...
                        abs(pter(kc,2)-pter(kc-2,2)) <= 1E-5 )
                    fprintf('Convergence␣problems␣\n')
                    fprintf('Peso␣%1.9f␣%1.9f␣\n',weiu(1),weiu(2))
                    cv1 = 0;  cv2 = 0;  ce = 2;
                end
            end
            iu1g = 2*u1_bode(1,1);  iu2g = 2*u2_bode(1,1);
            a1 = tol<=cv1(1,1);  a2 = tol<=cv2(1,1);
            if (a1+a2) ~= 0
                salidaciclo = 0;
            else
                salidaciclo = 1;
            end
        else
            ce = -9;
            salidaciclo = 1;
        end
    end
    c_esta=0;
    if ce == 0
        c_esta = crealmi_dcp_esta(igs,F0_mod,G1_mod,...
            F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
            mgama,mpsi,mpct,N2,op,fac,...
            2*u1_bode(1,1),2*u2_bode(1,1),...
            lc1,lc2);
        if c_esta ~=0
            fprintf('Unstable␣at␣\n␣%1.9f␣%1.9f',weiu(1),weiu(2))
            gy = 5.99;  ex3 = -2.5;  kc3 = -5.5;
        else
            [gy,ex3,kc3] = bis_grande_dcp(igs,F0_mod,G1_mod,...
                F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
                mgama,mpsi,mpct,N2,op,fac,...
                2*u1_bode(1,1),2*u2_bode(1,1),...
                lc1,lc2);
        end
    else
        gy = 5.99;  ex3 = -ce;  kc3 = -5;
    end
    re_g(ctdr,:) = [weiu , gy , ex1 , kc1 , ex2 , kc2 , ex3 , kc3 , kc];
    end
```

```
function [F0 F1 F2 G1 G2 G3 H0] = mat_mimomodel(param1)
par1 = param1;
F0 = [par1(1,1) par1(2,1) ; par1(3,1) par1(4,1) ];  e = eig(F0);%
if norm(e(1,1))>=1
    error('Eigenvalues␣outside␣the␣unit␣circle')
```

```
end
if norm(e(2,1))>=1
    error('Eigenvalues␣outside␣the␣unit␣circle')
end
F1 = [par1(5,1)  0 ; par1(7,1)  0];
F2 = [0 par1(6,1) ; 0 par1(8,1) ];
G1 = [par1(9,1) par1(10,1) ; par1(11,1) par1(12,1) ];
G2 = [par1(13,1) 0 ; par1(15,1) 0];
G3 = [0 par1(14,1) ; 0 par1(16,1)];
H0 = [1 0 ; 0 1];
```

```
function [gamita,ex,kc] = biswi_dcp(igs,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
    mgama,mpsi,mpct,N2,op,fac,iu1,iu2,cual,ic1,ic2)
t_upper = igs.tuwi;  t_lower = 0;  t_med   = igs.tmwi;
si_z = t_upper:t_med:t_lower;  con = ones(length(si_z),2);
sol = crealmiwi_dcp(t_lower,igs,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,mgama,mpsi,...
    mpct,N2,op,fac,iu1,iu2,cual,ic1,ic2);
ro = 1;  con(ro,:) = [t_lower sol];
for i=t_lower+abs(t_med):abs(t_med):t_upper
    ro = ro + 1;
    sol = crealmiwi_dcp(i,igs,F0_mod,G1_mod,...
        F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,mgama,mpsi,...
        mpct,N2,op,fac,iu1,iu2,cual,ic1,ic2);
    con(ro,:) = [i sol];
    if con(ro,2) * con(ro-1,2) == 0
        break
    end
end
if igs.prnt == 1
    for i=1:ro
        fprintf('%1.1f␣,␣%1.0f␣\n',con(i,1),con(i,2))
    end
    fprintf('\n')
end
in = find((con(:,2) == 0));  tol = 1E-8;  kc = 0;  salida = 1;
%% Iterations
if isempty(in) == 1
    gamita = 1.9;  kc = -2;  ex = -9;
else
    t_upper = con(in(length(in),1),1);
    t_lower = t_upper+t_med;  t_works = t_upper;
    while abs((t_upper - t_lower))>tol
        kc = kc + 1;  t_test = (t_upper+t_lower)/2;
        sol = crealmiwi_dcp(t_test,igs,F0_mod,G1_mod,...
            F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,mgama,mpsi,...
            mpct,N2,op,fac,iu1,iu2,cual,ic1,ic2);
        con(ro,:) = [i sol];
        if sol ~= 0
            t_lower = t_test;
        else
            t_upper = t_test;  t_works = t_test;
        end
        if kc > 50
            t_upper = t_lower;  salida = 0;
            fprintf('Iterations␣reached␣at␣wi␣\n')
        end
    end
    if salida == 0  gamita = 1.8;  ex = -8;
    else
        if t_works >= 0
```

```
            gamita = sqrt(t_works);
            ex = crealmiwi_dcp(t_works,igs,F0_mod,G1_mod,...
                F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,mgama,mpsi,...
                mpct,N2,op,fac,iu1,iu2,cual,ic1,ic2);
        else
            gamita = 1.7;  ex = -7;
        end
    end
end
```

```
function respuesta = crealmiwi_dcp(gmt,igs,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
    mgama,mpsi,mpct,N2,op,fac,iu1,iu2,cual,ic1,ic2)
B11 = zeros(igs.n_x,1);  B21 = zeros(igs.n_x,1);  B31 = zeros(igs.n_u,1);
B41 = (1-igs.BW);  B_m = [B11 ; B21 ; B31 ; B41];
D11= zeros(1,1);
P = sdpvar(7,7,'symmetric');
ll_a = @jalalmi_dcp(gmt,P,du_1,du_2,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C_in,...
    mpct,mpsi,mgama,...
    igs,N2,B_m,C_m,D11,fac);
du_1 = 0;  du_2 = 0;  dc_1 = 0;  dc_2 = 0;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L1 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = -ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L2 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = +ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L3 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = -ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L4 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = +ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L5 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = -ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L6 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = +ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L7 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = -ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L8 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = +ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L9 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = -ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L10 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = +ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L11 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = -ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L12 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = +ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L13 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = -ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L14 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = +ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L15 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = -ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L16 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = +ic2;
C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2);  C_in = C + C*[dc_1 0 ; 0 dc_2];  L17 = ll_a;  clear C_m
F = set(P>0) + set(L1<0) + set(L2<0) + set(L3<0) + set(L4<0) + ...
    set(L5<0) + set(L6<0) + set(L7<0) + set(L8<0) + set(L9<0) + ...
    set(L10<0) + set(L11<0) + set(L12<0) + set(L13<0) + set(L14<0) + ...
    set(L15<0) + set(L16<0) + set(L17<0);
options = sdpsettings('solver',op.solver,...
    'verbose',op.verb,'warning',op.warn,...
    'cachesolvers',1);
```

```
F_sol = solvesdp(F,[],options);
respuesta = F_sol.problem;
```

```
function C_m = incer_c_bode(C,igs,mpct,N2,cual,dc_1,dc_2)
C_in = C + C*[dc_1 0 ; 0 dc_2];
if cual == 1
    cu = [1 0];
elseif cual == 2
    cu = [0 1];
end
C11 = cu*mpct*N2*C;  C12 = -cu*mpct*N2*C_in;  C13 = cu*zeros(igs.n_y,igs.n_u);  C14 = -cu*mpct*N2*igs.WF;
C_m = [C11 , C12 , C13 , C14];
```

```
function l_mi = jalalmi_dcp(gmt,P,du_1,du_2,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
    mpct,mpsi,mgama,igs,N2,B_m,C_m,D_m,fac)
A11 = F0_mod + (G1_mod*mpct*(-mpsi + N2*eye(igs.n_y)));  A12 = -G1_mod*mpct*N2*C;
A13 = G1_mod*( eye(igs.n_u) + mpct*(-mgama) );  A14 = -G1_mod*mpct*N2*igs.WF;
A21 = (G1_pro + du_1*G2_pro + du_2*G3_pro)*( mpct*( -mpsi + N2*eye(igs.n_y) ) );
A22 = (F0_pro + du_1*F1_pro + du_2*F2_pro) + (G1_pro + du_1*G2_pro + du_2*G3_pro)*(-mpct*N2*C);
A23 = (G1_pro + du_1*G2_pro + du_2*G3_pro)*(eye(igs.n_u) + mpct*( -mgama ) );
A24 = (G1_pro + du_1*G2_pro + du_2*G3_pro)*(-mpct*N2*igs.WF);
A31 = mpct*( -mpsi + N2*eye(igs.n_y) );  A32 = -mpct*N2*C;
A33 = fac.a*eye(igs.n_u) + mpct*( -mgama );  A34 = -mpct*N2*igs.WF;
A41 = zeros(1,igs.n_x);  A42 = zeros(1,igs.n_x);  A43 = zeros(1,igs.n_u);  A44 = igs.BW;
A_m = [...
    A11 , A12 , A13 , A14 ; A21 , A22 , A23 , A24 ; ...
    A31 , A32 , A33 , A34 ; A41 , A42 , A43 , A44];
e = eig(A_m);  eo = abs(e) >= 1;
if sum(eo) ~= 0, fprintf('outside␣unit␣circle\n'),  end
l_mi = [...
    ((A_m'*P*A_m)-P) , A_m'*P*B_m       , C_m' ; ...
    B_m'*P*A_m       , (B_m'*P*B_m)-gmt , D_m' ; ...
    C_m              , D_m              , -eye(size(C_m,1))];
```

```
function su_p = pbodei(F0,G1,mpsi,mgama,mpct,igs,...
    Wgama,N2,cual,v_fr_an)
A11 = F0 - G1*mpct*mpsi;  A12 = G1*( eye(igs.n_u) - mpct*mgama );
A21 = -mpct*mpsi;  A22 = eye(igs.n_u)-mpct*mgama;
A_m = [A11 A12 ; A21 A22];  e = eig(A_m);  eo = e >= 1;
if sum(eo) ~= 0
    error('outside␣for␣Bode\n')
end
B11 = G1*Wgama;  B21 = Wgama;  B_m = [B11;B21];
if cual == 1
    C_m = [[1 0]*A21 [1 0]*A22];
elseif cual == 2
    C_m = [[0 1]*A21 [0 1]*A22];
end
if cual == 1
    D_m = [1 0]*B21;% caso antes --> % Wgama(1,1);
elseif cual == 2
    D_m = [0 1]*B21;% caso antes --> % Wgama(2,1);
end
sys = ss(A_m,B_m,C_m,D_m,igs.tbode);
mag = bode(sys,{igs.libode,igs.lsbode});
su_p = mag(mag==max(mag)); %Logical Indexing
```

```
function [respuesta,pnumerica] = crealmi_dcp_esta(igs,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
    mgama,mpsi,mpct,N2,op,fac,iu1,iu2,ic1,ic2)
```

```
ll_a = @jalalmi_dcp_esta(P,du_1,du_2,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C_con_i,...
    mpct,mpsi,mgama,igs,N2,fac);
P = sdpvar(7,7,'symmetric');
du_1 = 0;  du_2 = 0;  dc_1 = 0;  dc_2 = 0;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L1 = ll_a;
du_1 = -iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = -ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L2 = ll_a;
du_1 = -iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = +ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L3 = ll_a;
du_1 = -iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = -ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L4 = ll_a;
du_1 = -iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = +ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L5 = ll_a;
du_1 = -iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = -ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L6 = ll_a;
du_1 = -iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = +ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L7 = ll_a;
du_1 = -iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = -ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L8 = ll_a;
du_1 = -iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = +ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L9 = ll_a;
du_1 = +iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = -ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L10 = ll_a;
du_1 = +iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = +ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L11 = ll_a;
du_1 = +iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = -ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L12 = ll_a;
du_1 = +iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = +ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L13 = ll_a;
du_1 = +iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = -ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L14 = ll_a;
du_1 = +iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = +ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L15 = ll_a;
du_1 = +iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = -ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L16 = ll_a;
du_1 = +iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = +ic2;  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L17 = ll_a;
F = set(P>0) + set(L1<0) + set(L2<0) + set(L3<0) + set(L4<0) + ...
    set(L5<0) + set(L6<0) + set(L7<0) + set(L8<0) + set(L9<0) + ...
    set(L10<0) + set(L11<0) + set(L12<0) + set(L13<0) + set(L14<0) + ...
    set(L15<0) + set(L16<0) + set(L17<0);
options = sdpsettings('solver',op.solver,...
    'verbose',op.verb,'warning',op.warn,...
    'cachesolvers',1);
F_sol = solvesdp(F,[],options);
respuesta = F_sol.problem;  pnumerica = double(P);
```

```
function l_mi = jalalmi_dcp_esta(P,du_1,du_2,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
    mpct,mpsi,mgama,igs,N2,fac)
A11 = F0_mod + (G1_mod*mpct*(-mpsi + N2*eye(igs.n_y)));  A12 = -G1_mod*mpct*N2*C;
A13 = G1_mod*( eye(igs.n_u) + mpct*(-mgama) );   A14 = -G1_mod*mpct*N2*igs.WF;
A21 = (G1_pro + du_1*G2_pro + du_2*G3_pro)*( mpct*( -mpsi + N2*eye(igs.n_y) ) );
A22 = (F0_pro + du_1*F1_pro + du_2*F2_pro) + (G1_pro + du_1*G2_pro + du_2*G3_pro)*(-mpct*N2*C);
A23 = (G1_pro + du_1*G2_pro + du_2*G3_pro)*(eye(igs.n_u) + mpct*( -mgama ) );
A24 = (G1_pro + du_1*G2_pro + du_2*G3_pro)*(-mpct*N2*igs.WF);
A31 = mpct*( -mpsi + N2*eye(igs.n_y) );  A32 = -mpct*N2*C;
A33 = fac.a*eye(igs.n_u) + mpct*( -mgama );  A34 = -mpct*N2*igs.WF;
A41 = zeros(1,igs.n_x);  A42 = zeros(1,igs.n_x);  A43 = zeros(1,igs.n_u);  A44 = igs.BW;
A_m = [...
    A11 , A12 , A13 , A14 ; A21 , A22 , A23 , A24 ; ...
    A31 , A32 , A33 , A34 ; A41 , A42 , A43 , A44];
l_mi = (A_m'*P*A_m)-P;
```

```
function [gamita,ex,kc] = bis_grande_dcp(igs,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
    mgama,mpsi,mpct,N2,op,fac,iu1,iu2,ic1,ic2)
t_upper = igs.tugr;  t_lower = 0;  t_med   = igs.tmgr;
si_z = t_upper:t_med:t_lower;  con = ones(length(si_z),2);
sol = crealmi_dcp(t_lower,igs,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
    mgama,mpsi,mpct,N2,op,fac,iu1,iu2,ic1,ic2);
ro = 1; con(ro,:) = [t_lower sol];
for i=t_lower+abs(t_med):abs(t_med):t_upper
    ro = ro + 1;
    sol = crealmi_dcp(i,igs,F0_mod,G1_mod,...
        F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
        mgama,mpsi,mpct,N2,op,fac,iu1,iu2,ic1,ic2);
    con(ro,:) = [i sol];
    if con(ro,2) * con(ro-1,2) == 0
        break
    end
end
if igs.prnt == 1
    for i=1:ro
        fprintf('%1.3f␣,␣%1.0f␣\n',con(i,1),con(i,2))
    end
    fprintf('\n')
end
in = find((con(:,2) == 0));  tol = 1E-8;  kc = 0;  salida = 1;
if isempty(in) == 1
    gamita = 1.9;  kc     = -2; ex     = -9;
else
    t_upper = con(in(length(in),1),1);
    t_lower = t_upper+t_med;  t_works = t_upper;
    while abs((t_upper - t_lower))>tol
        kc = kc + 1;  t_test = (t_upper+t_lower)/2;
        sol = crealmi_dcp(t_test,igs,F0_mod,G1_mod,...
            F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
            mgama,mpsi,mpct,N2,op,fac,iu1,iu2,ic1,ic2);
        con(ro,:) = [i sol];
        if sol ~= 0
            t_lower = t_test;
        else
            t_upper = t_test;  t_works = t_test;
        end
        if kc > 50
            t_upper = t_lower;  salida = 0; fprintf('Increase␣iterations␣\n')
        end
    end
    if salida == 0,
        gamita = 1.8;  ex = -8;
    else
        if t_works >= 0
            gamita = sqrt(t_works);
            ex = crealmi_dcp(t_works,igs,F0_mod,G1_mod,...
                F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
                mgama,mpsi,mpct,N2,op,fac,iu1,iu2,ic1,ic2);
        else
            gamita = 1.7;  ex = -7;
        end
    end
end
```

```
function [respuesta,pnumerica] = crealmi_dcp(gmt,igs,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C,...
    mgama,mpsi,mpct,N2,op,fac,iu1,iu2,ic1,ic2)
B11 = zeros(igs.n_x,1);  B21 = zeros(igs.n_x,1);  B31 = zeros(igs.n_u,1);
B41 = (1-igs.BW);  B_m = [B11 ; B21 ; B31 ; B41];
D11 = [zeros(igs.n_y,1) ; zeros(igs.n_u,1) ];
P = sdpvar(7,7,'symmetric');
ll_a = @jalalmi_dcp(gmt,P,du_1,du_2,F0_mod,G1_mod,...
    F0_pro,F1_pro,F2_pro,G1_pro,G2_pro,G3_pro,C_con_i,...
    mpct,mpsi,mgama,...
    igs,N2,B_m,C_m,D11,fac);
du_1 = 0;  du_2 = 0;  dc_1 = 0;  dc_2 = 0;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L1 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = -ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L2 = ll_a;  clear C_m
```

```
du_1 = -iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = +ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L3 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = -ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L4 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = +ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L5 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = -ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L6 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = +ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L7 = ll_a;  clear C_m
du_1 = -iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = -ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L8= ll_a;   clear C_m
du_1 = -iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = +ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L9 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = -ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L10 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = -iu2;  dc_1 = -ic1;  dc_2 = +ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L11 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = -ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L12 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = -iu2;  dc_1 = +ic1;  dc_2 = +ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L13 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = -ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2]; L14 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = +iu2;  dc_1 = -ic1;  dc_2 = +ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L15 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = -ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L16 = ll_a;  clear C_m
du_1 = +iu1;  du_2 = +iu2;  dc_1 = +ic1;  dc_2 = +ic2;
C_m = incer_c_gen3(C,igs,dc_1,dc_2,mpct,mpsi,N2,mgama);  C_con_i = C + C*[dc_1 0 ; 0 dc_2];  L17 = ll_a;  clear C_m
F = set(P>0) + set(L1<0) + set(L2<0) + set(L3<0) + set(L4<0) + ...
    set(L5<0) + set(L6<0) + set(L7<0) + set(L8<0) + set(L9<0) + ...
    set(L10<0) + set(L11<0) + set(L12<0) + set(L13<0) + set(L14<0) + ...
    set(L15<0) + set(L16<0) + set(L17<0);
options = sdpsettings('solver',op.solver,...
    'verbose',op.verb,'warning',op.warn,...
    'cachesolvers',1);
F_sol = solvesdp(F,[],options);
respuesta = F_sol.problem;
pnumerica = double(P);
```

```
function C_m = incer_c_gen3(C,igs,dc1,dc2,...
    mpct,mpsi,N2,mgama)
C_con_i = C + C*[dc1 0 ; 0 dc2];
C11 = zeros(igs.n_y,igs.n_x);  C12 = C_con_i;  C13 = zeros(igs.n_y,igs.n_u);  C14 = igs.WF;
C21 = igs.pau*(mpct*( -mpsi + N2*eye(igs.n_y) ));  C22 = igs.pau*(-mpct*N2*C_con_i);
C23 = igs.pau*(eye(igs.n_u) + mpct*( -mgama ));  C24 = igs.pau*(-mpct*N2*igs.WF);
C_m = [C11 , C12 , C13 , C14 ; C21 , C22 , C23 , C24];
```

Tables I.3 to I.6 show the programs used to calculates the interconnections matrices and the μ based robustness test.

Table I.3: List of programs to calculate $\mathbf{M}$ for SISO case example

Program Name	Description
ssdabmdentro.m	Calculate $\mathbf{M}$
ssarribita.m	Internal program to calculate $\mathbf{M}_{12}$
ssarriba1.m	Internal program to calculate $\mathbf{M}_{12}$
ssbloquev1.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssbloquev2.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssbloquev3.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssbloquev4.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssbloquev5.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssbloquev6.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssbloquev7.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssbloquev8.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssbloquev9.m	Internal program to calculate $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
ssabajito.m	Internal program to calculate $\mathbf{M}_{21}$
ssabajony1.m	Internal program to calculate $\mathbf{M}_{21}$
ssabajovp.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev1a.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev2a.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev3a.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev4a.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev5a.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev6a.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev7a.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev8a.m	Internal program to calculate $\mathbf{M}_{21}$
ssbloquev9a.m	Internal program to calculate $\mathbf{M}_{21}$

```
function m = ssdabmdentro(esi,k,ed,u,h11_ll,h11_cp,ic1,hy1,...
    u1zero,u1menos1,du1,u1r,vhll_11,vhcp_11,vtc,cal_nom)
marriba = arribita(k,u(1:10,1),u1zero);
menmedio = [bloquev1(k) ; bloquev2(k) ; bloquev3(k) ; bloquev4(k) ; ...
    bloquev5(k) ; bloquev6(k) ; bloquev7(k) ; bloquev8(k) ; ...
    bloquev9(k) ];
mabajo = abajito(ed,ic1,h11_ll,h11_cp,u1zero,u1menos1,esi,du1,k,...
    u1r,hy1,vhll_11,vhcp_11,cal_nom);
m = zeros(310,310);
m(001:143,293:310) = marriba;  % 143 X 018
m(144:292,033:143) = menmedio; % 149 X 111
m(293:310,:) = mabajo;         % 018 X 310
m(294,012) = m(294,012) + (1/k)*hy1*m(293,011:292)*m(011:292,293);
m(295,013) = m(295,013) + (1/k)*hy1*m(294,011:292)*m(011:292,294);
m(296,014) = m(296,014) + (1/k)*hy1*m(295,011:292)*m(011:292,295);
m(297,015) = m(297,015) + (1/k)*hy1*m(296,011:292)*m(011:292,296);
m(298,016) = m(298,016) + (1/k)*hy1*m(297,011:292)*m(011:292,297);
m(299,017) = m(299,017) + (1/k)*hy1*m(298,011:292)*m(011:292,298);
```

```
m(300,018) = m(300,018) + (1/k)*hy1*m(299,011:292)*m(011:292,299);
m(301,019) = m(301,019) + (1/k)*hy1*m(300,011:292)*m(011:292,300);
m(302,020) = m(302,020) + (1/k)*hy1*m(301,011:292)*m(011:292,301);
```

```
function mfinal = ssarribita(k,u1,u1zero)
mfinal = zeros(143,18);
mfinal(001:010,01:10) = k*eye(10);   mfinal(011:020,01:10) = k*eye(10);
mfinal(021,11) = k*u1zero;   mfinal(022,11) = k*u1(1);
mfinal(023,12) = k*u1(1);   mfinal(024,12) = k*u1(2);
mfinal(025,13) = k*u1(2);   mfinal(026,13) = k*u1(3);
mfinal(027,14) = k*u1(3);   mfinal(028,14) = k*u1(4);
mfinal(029,15) = k*u1(1);   mfinal(030,16) = k*u1(2);
mfinal(031,17) = k*u1(3);   mfinal(032,18) = k*u1(4);
mfinal(033:143,01:10) = arriba1(k,u1);
```

```
function m = ssarriba1(k,u)
m = zeros(111,10);
m(01,01) = k*u(1);   m(02,01) = k*u(1);   m(03,01) = k*u(1);
m(04,02) = k*u(1);   m(05,02) = k*u(1);   m(06,03) = k*u(1);
m(08,01) = k*u(1)*u(1);   m(09,02) = k*u(1)*u(1);
m(10,03) = k*u(1)*u(1);   m(12,02) = k*u(2);   m(13,02) = k*u(2);
m(14,03) = k*u(2);   m(15,04) = k*u(2);   m(17,02) = k*u(2)*u(2);
m(18,03) = k*u(2)*u(2);   m(19,04) = k*u(2)*u(2);   m(21,03) = k*u(3);
m(22,04) = k*u(3);   m(23,05) = k*u(3);   m(25,03) = k*u(3)*u(3);
m(26,04) = k*u(3)*u(3);   m(27,05) = k*u(3)*u(3);   m(29,04) = k*u(4);
m(30,05) = k*u(4);   m(31,06) = k*u(4);   m(33,04) = k*u(4)*u(4);
m(34,05) = k*u(4)*u(4);   m(35,06) = k*u(4)*u(4);   m(37,05) = k*u(5);
m(38,06) = k*u(5);   m(39,07) = k*u(5);   m(41,05) = k*u(5)*u(5);
m(42,06) = k*u(5)*u(5);   m(43,07) = k*u(5)*u(5);   m(45,06) = k*u(6);
m(46,07) = k*u(6);   m(47,08) = k*u(6);    m(49,06) = k*u(6)*u(6);
m(50,07) = k*u(6)*u(6);   m(51,08) = k*u(6)*u(6);   m(53,07) = k*u(7);
m(54,08) = k*u(7);   m(55,09) = k*u(7);   m(57,07) = k*u(7)*u(7);
m(58,08) = k*u(7)*u(7);   m(59,09) = k*u(7)*u(7);   m(61,08) = k*u(8);
m(62,09) = k*u(8);   m(63,10) = k*u(8);   m(64,08) = k*u(8)*u(8);
m(65,09) = k*u(8)*u(8);   m(66,10) = k*u(8)*u(8);   m(67,09) = k*u(9);
m(68,10) = k*u(9);   m(69,09) = k*u(9)*u(9);   m(70,10) = k*u(9)*u(9);
m(71,10) = k*u(10);   m(72,10) = k*u(10)*u(10);   m(73,02) = k*u(1)*u(2);
m(74,03) = k*u(1)*u(2);   m(76,03) = k*u(2)*u(3);   m(77,04) = k*u(2)*u(3);
m(79,04) = k*u(3)*u(4);   m(80,05) = k*u(3)*u(4);   m(82,05) = k*u(4)*u(5);
m(83,06) = k*u(4)*u(5);   m(85,06) = k*u(5)*u(6);   m(86,07) = k*u(5)*u(6);
m(88,07) = k*u(6)*u(7);   m(89,08) = k*u(6)*u(7);   m(91,08) = k*u(7)*u(8);
m(92,09) = k*u(7)*u(8);   m(94,09) = k*u(8)*u(9);   m(95,10) = k*u(8)*u(9);
m(96,10) = k*u(9)*u(10);  m(97,03) = k*u(1)*u(3);   m(99,04) = k*u(2)*u(4);
m(101,05) = k*u(3)*u(5); m(103,06) = k*u(4)*u(6); m(105,07) = k*u(5)*u(7);
m(107,08) = k*u(6)*u(8); m(109,09) = k*u(7)*u(9); m(111,10) = k*u(8)*u(10);
```

```
function m = ssbloquev1(k)
m = zeros(19,111);
g1 = [03 , 13 , 21 , 29 , 37 , 45 , 53 , 61 , 67 , 71];
for i=1:10,    m(i,g1(i)) = k;   end
g2 = [05 , 14 , 22 , 30 , 38 , 46 , 54 , 62 , 68];
for i=1:9,    m(10+i,g2(i)) = k;   end
```

```
function m = ssbloquev2(k)
m = zeros(19,111);
g1 = [08 , 17 , 25 , 33 , 41 , 49 , 57 , 64 , 69 , 72];
for i=1:10,    m(i,g1(i)) = k;   end
g2 = [09 , 18 , 26 , 34 , 42 , 50 , 58 , 65 , 70];
for i=1:9,    m(10+i,g2(i)) = k;   end
```

```
function m = ssbloquev3(k)
m = zeros(17,111);
g1 = [05 , 14 , 22 , 30 , 38 , 46 , 54 , 62 , 68];
for i=1:9,    m(i,g1(i)) = k;   end
g2 = [06 , 15 , 23 , 31 , 39 , 47 , 55 , 63];
for i=1:8,    m(9+i,g2(i)) = k;   end
```

```
function m = ssbloquev4(k)
m = zeros(17,111);
g1 = [09 , 18 , 26 , 34 , 42 , 50 , 58 , 65 , 70];
for i=1:9,    m(i,g1(i)) = k;   end
g2 = [10 , 19 , 27 , 35 , 43 , 51 , 59 , 66];
for i=1:8,    m(9+i,g2(i)) = k;   end
```

```
function m = ssbloquev5(k)
m = zeros(15,111);
g1 = [06 , 15 , 23 , 31 , 39 , 47 , 55 , 63];
for i=1:8,    m(i,g1(i)) = k;   end
g2 = [07 , 16 , 24 , 32 , 40 , 48 , 56];
for i=1:7,    m(8+i,g2(i)) = k;   end
```

```
function m = ssbloquev6(k)
m = zeros(15,111);
g1 = [10 , 19 , 27 , 35 , 43 , 51 , 59 , 66];
for i=1:8,    m(i,g1(i)) = k;   end
g2 = [11 , 20 , 28 , 36 , 44 , 52 , 60];
for i=1:7,    m(8+i,g2(i)) = k;   end
```

```
function m = ssbloquev7(k)
m = zeros(17,111);
g1 = [73 , 76 , 79 , 82 , 85 , 88 , 91 , 94 , 96];
for i=1:9,    m(i,g1(i)) = k;   end
g2 = [74 , 77 , 80 , 83 , 86 , 89 , 92 , 95];
for i=1:8,    m(9+i,g2(i)) = k;   end
```

```
function m = ssbloquev8(k)
m = zeros(15,111);
g1 = [97 , 99 , 101 , 103 , 105 , 107 , 109 , 111];
for i=1:8,    m(i,g1(i)) = k;   end
g2 = [98 , 100 , 102 , 104 , 106 , 108 , 110];
for i=1:7,    m(8+i,g2(i)) = k;   end
```

```
function m = ssbloquev9(k)
m = zeros(15,111);
g1 = [74 , 77 , 80 , 83 , 86 , 89 , 92 , 95];
for i=1:8,    m(i,g1(i)) = k;   end
g2 = [75 , 78 , 81 , 84 , 87 , 90 , 93];
for i=1:7,    m(8+i,g2(i)) = k;   end
```

```
function mfinal = ssabajito(ed,ic1,hll_11,hcp_11,u1zero,u1menos1,esi,du1,...
    k,u1r,hy1,vhll_11,vhcp_11,cal_nom)
mfinal = zeros(18,310);
for i=1:10,    mfinal(i,i) = ed(i,1);   end
for i=1:10,    mfinal(i,10+i) = ic1(i,1);   end
mfinal(11,21) = +du1(1);   mfinal(11,22) = -du1(1);
mfinal(12,23) = +du1(2);   mfinal(12,24) = -du1(2);
mfinal(13,25) = +du1(3);   mfinal(13,26) = -du1(3);
mfinal(14,27) = +du1(4);   mfinal(14,28) = -du1(4);
mfinal(15,29) = k/u1r(1);   mfinal(16,30) = k/u1r(2);
mfinal(17,31) = k/u1r(3);   mfinal(18,32) = k/u1r(4);
mfinal(01:10,033:143) = abajony1(hll_11,hcp_11,u1zero,u1menos1);
```

```
bvar = abajovp(hy1,vhll_11,vhcp_11,cal_nom);
mfinal(01:10,144:292) = bvar;
mfinal(01:18,293:310) = esi*eye(18);
```

```
function m = ssabajony1(hll,hcp,uzero,umenos1)
m = zeros(10,111);
m(01,001) = hcp(1,3)*umenos1;   m(01,002) = hcp(1,2)*uzero;
m(01,003) = hll(1,1);   m(02,004) = hcp(2,3)*uzero;
m(02,005) = hll(1,2);   m(03,006) = hll(1,3);   m(01,008) = hcp(1,1);
m(02,009) = hcp(2,2);   m(03,010) = hcp(3,3);   m(02,012) = hcp(1,3)*uzero;
m(02,013) = hll(1,1);   m(03,014) = hll(1,2);   m(04,015) = hll(1,3);
m(02,017) = hcp(1,1);   m(03,018) = hcp(2,2);   m(04,019) = hcp(3,3);
m(03,021) = hll(1,1);   m(04,022) = hll(1,2);   m(05,023) = hll(1,3);
m(03,025) = hcp(1,1);   m(04,026) = hcp(2,2);   m(05,027) = hcp(3,3);
m(04,029) = hll(1,1);   m(05,030) = hll(1,2);   m(06,031) = hll(1,3);
m(04,033) = hcp(1,1);   m(05,034) = hcp(2,2);   m(06,035) = hcp(3,3);
m(05,037) = hll(1,1);   m(06,038) = hll(1,2);   m(07,039) = hll(1,3);
m(05,041) = hcp(1,1);   m(06,042) = hcp(2,2);   m(07,043) = hcp(3,3);
m(06,045) = hll(1,1);   m(07,046) = hll(1,2);   m(08,047) = hll(1,3);
m(06,049) = hcp(1,1);   m(07,050) = hcp(2,2);   m(08,051) = hcp(3,3);
m(07,053) = hll(1,1);   m(08,054) = hll(1,2);   m(09,055) = hll(1,3);
m(07,057) = hcp(1,1);   m(08,058) = hcp(2,2);   m(09,059) = hcp(3,3);
m(08,061) = hll(1,1);   m(09,062) = hll(1,2);   m(10,063) = hll(1,3);
m(08,064) = hcp(1,1);   m(09,065) = hcp(2,2);   m(10,066) = hcp(3,3);
m(09,067) = hll(1,1);   m(10,068) = hll(1,2);   m(09,069) = hcp(1,1);
m(10,070) = hcp(2,2);   m(10,071) = hll(1,1);   m(10,072) = hcp(1,1);
m(02,073) = hcp(1,2);   m(03,074) = hcp(2,3);   m(03,076) = hcp(1,2);
m(04,077) = hcp(2,3);   m(04,079) = hcp(1,2);   m(05,080) = hcp(2,3);
m(05,082) = hcp(1,2);   m(06,083) = hcp(2,3);   m(06,085) = hcp(1,2);
m(07,086) = hcp(2,3);   m(07,088) = hcp(1,2);   m(08,089) = hcp(2,3);
m(08,091) = hcp(1,2);   m(09,092) = hcp(2,3);   m(09,094) = hcp(1,2);
m(10,095) = hcp(2,3);   m(10,096) = hcp(1,2);   m(03,097) = hcp(1,3);
m(04,099) = hcp(1,3);   m(05,101) = hcp(1,3);   m(06,103) = hcp(1,3);
m(07,105) = hcp(1,3);   m(08,107) = hcp(1,3);   m(09,109) = hcp(1,3);
m(10,111) = hcp(1,3);
```

```
function m = ssabajovp(hy1,vhll_11,vhcp_11,c_nom)
v1 = bloquev1a(hy1,vhll_11,c_nom);
v2 = bloquev2a(hy1,vhcp_11,c_nom);
v3 = bloquev3a(hy1,vhll_11,c_nom);
v4 = bloquev4a(hy1,vhcp_11,c_nom);
v5 = bloquev5a(hy1,vhll_11,c_nom);
v6 = bloquev6a(hy1,vhcp_11,c_nom);
v7 = bloquev7a(hy1,vhcp_11,c_nom);
v8 = bloquev8a(hy1,vhcp_11,c_nom);
v9 = bloquev9a(hy1,vhcp_11,c_nom);
m = [v1 , v2 , v3 , v4 , v5 , v6 , v7 , v8 , v9];
```

```
function m = ssbloquev1a(hy1,vhll_11,c_nom)
m = zeros(10,19);
m(01:10,01:10) =     vhll_11(1,1)*eye(10);
m(02:10,11:19) = hy1*vhll_11(1,1)*eye(09);
if c_nom == 1
    m(01:09,01:10) = 0*m(01:09,01:10);
    m(02:09,11:19) = 0*m(02:09,11:19);
end
```

```
function m = ssbloquev2a(hy1,vhcp_11,c_nom)
m = zeros(10,19);
m(01:10,01:10) =     vhcp_11(1,1)*eye(10);
m(02:10,11:19) = hy1*vhcp_11(1,1)*eye(09);
```

```
if c_nom == 1
    m(01:09,01:10) = 0*m(01:09,01:10);
    m(02:09,11:19) = 0*m(02:09,11:19);
end
```

```
function m = ssbloquev3a(hy1,vhll_11,c_nom)
m = zeros(10,17);
m(02:10,01:09) =     vhll_11(1,2)*eye(9);
m(03:10,10:17) = hy1*vhll_11(1,2)*eye(8);
if c_nom == 1
    m(02:09,01:09) = 0*m(02:09,01:09);
    m(03:09,10:17) = 0*m(03:09,10:17);
end
```

```
function m = ssbloquev4a(hy1,vhcp_11,c_nom)
m = zeros(10,17);
m(02:10,01:09) =     vhcp_11(2,2)*eye(9);
m(03:10,10:17) = hy1*vhcp_11(2,2)*eye(8);
if c_nom == 1
    m(02:09,01:09) = 0*m(02:09,01:09);
    m(03:09,10:17) = 0*m(03:09,10:17);
end
```

```
function m = ssbloquev5a(hy1,vhll_11,c_nom)\
m = zeros(10,15);
m(03:10,01:08) =     vhll_11(1,3)*eye(8);
m(04:10,09:15) = hy1*vhll_11(1,3)*eye(7);
if c_nom == 1
    m(03:09,01:08) = 0*m(03:09,01:08);
    m(04:09,09:15) = 0*m(04:09,09:15);
end
```

```
function m = ssbloquev6a(hy1,vhcp_11,c_nom)
m = zeros(10,15);
m(03:10,01:08) =     vhcp_11(3,3)*eye(8);
m(04:10,09:15) = hy1*vhcp_11(3,3)*eye(7);
if c_nom == 1
    m(03:09,01:08) = 0*m(03:09,01:08);
    m(04:09,09:15) = 0*m(04:09,09:15);
end
```

```
function m = ssbloquev7a(hy1,vhcp_11,c_nom)
m = zeros(10,17);
m(02:10,01:09) =     vhcp_11(1,2)*eye(9);
m(03:10,10:17) = hy1*vhcp_11(1,2)*eye(8);
if c_nom == 1
    m(02:09,01:09) = 0*m(02:09,01:09);
    m(03:09,10:17) = 0*m(03:09,10:17);
end
```

```
function m = ssbloquev8a(hy1,vhcp_11,c_nom)
m = zeros(10,15);
m(03:10,01:08) =     vhcp_11(1,3)*eye(8);
m(04:10,09:15) = hy1*vhcp_11(1,3)*eye(7);
if c_nom == 1
    m(03:09,01:08) = 0*m(03:09,01:08);
    m(04:09,09:15) = 0*m(04:09,09:15);
end
```

```
function m = ssbloquev9a(hy1,vhcp_11,c_nom)
m = zeros(10,15);
m(03:10,01:08) =     vhcp_11(2,3)*eye(8);
m(04:10,09:15) = hy1*vhcp_11(2,3)*eye(7);
if c_nom == 1
    m(03:09,01:08) = 0*m(03:09,01:08);
    m(04:09,09:15) = 0*m(04:09,09:15);
end
```

Table I.4: List of programs to calculate $\mathbf{M}$ for MIMO case example with $M = 2$

Program Name	Description
dabmdentro_p10.m	Calculate $\mathbf{M}$ with $n_u = 2$, $n_y = 2$ and $M = 2$
arribita_p10.m	Internal program $\mathbf{M}_{12}$
arriba1_p10.m	Internal program $\mathbf{M}_{12}$
bloquev1_p10.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
bloquev2_p10.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
bloquev3_p10.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
bloquev4_p10.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
bloquev7_p10.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\mathrm{C1CCD}}$)
abajito_p10.m	Internal program $\mathbf{M}_{21}$
abajony1_p10.m	Internal program $\mathbf{M}_{21}$
abajony2_p10.m	Internal program $\mathbf{M}_{21}$
abajovp_p10.m	Internal program $\mathbf{M}_{21}$
bloquev1a_p10.m	Internal program $\mathbf{M}_{21}$
bloquev2a_p10.m	Internal program $\mathbf{M}_{21}$
bloquev3a_p10.m	Internal program $\mathbf{M}_{21}$
bloquev4a_p10.m	Internal program $\mathbf{M}_{21}$
bloquev7a_p10.m	Internal program $\mathbf{M}_{21}$

```
function m = dabmdentro_p10(esi,k,ed,u,h11_ll,h12_ll,h21_ll,h22_ll,...
    h11_cp,h12_cp,h21_cp,h22_cp,ic1,ic2,hy1,hy2,u1zero,u1menos1,...
    u2zero,u2menos1,du1,du2,u1r,u2r,vhll_11,vhcp_11,...
    vhll_12,vhcp_12,vhll_21,vhcp_21,vhll_22,vhcp_22,vtc,cal_nom)
marriba = arribita_p10(k,u(01:10,1),u(11:20,1),u1zero,u2zero); % 340 X 28
menmedio = [bloquev1_p10(k) ; bloquev2_p10(k) ; bloquev3_p10(k) ; ...
    bloquev4_p10(k) ; bloquev7_p10(k) ];
mabajo = abajito_p10(ed,ic1,ic2,h11_ll,h11_cp,h12_ll,h12_cp,...
    h21_ll,h21_cp,h22_ll,h22_cp,u1zero,u1menos1,u2zero,u2menos1,...
    esi,du1,du2,k,u1r,u2r,hy1,hy2,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,cal_nom); % 28 X 724
m = zeros(724,724);
m( 001 : 340 , 697 : 724 ) = marriba;
m( 341 : 696 , 053 : 340 ) = menmedio;
m( 697 : 724 ,     :     ) = mabajo;
m(698,022) = m(698,022) + (1/k)*hy1*m(697,021:696)*m(021:696,697);
m(699,023) = m(699,023) + (1/k)*hy1*m(698,021:696)*m(021:696,698);
m(700,024) = m(700,024) + (1/k)*hy1*m(699,021:696)*m(021:696,699);
m(701,025) = m(701,025) + (1/k)*hy1*m(700,021:696)*m(021:696,700);
m(702,026) = m(702,026) + (1/k)*hy1*m(701,021:696)*m(021:696,701);
m(703,027) = m(703,027) + (1/k)*hy1*m(702,021:696)*m(021:696,702);
m(704,028) = m(704,028) + (1/k)*hy1*m(703,021:696)*m(021:696,703);
```

```
m(705,029) = m(705,029) + (1/k)*hy1*m(704,021:696)*m(021:696,704);
m(706,030) = m(706,030) + (1/k)*hy1*m(705,021:696)*m(021:696,705);
m(708,032) = m(708,032) + (1/k)*hy2*m(707,021:696)*m(021:696,707);
m(709,033) = m(709,033) + (1/k)*hy2*m(708,021:696)*m(021:696,708);
m(710,034) = m(710,034) + (1/k)*hy2*m(709,021:696)*m(021:696,709);
m(711,035) = m(711,035) + (1/k)*hy2*m(710,021:696)*m(021:696,710);
m(712,036) = m(712,036) + (1/k)*hy2*m(711,021:696)*m(021:696,711);
m(713,037) = m(713,037) + (1/k)*hy2*m(712,021:696)*m(021:696,712);
m(714,038) = m(714,038) + (1/k)*hy2*m(713,021:696)*m(021:696,713);
m(715,039) = m(715,039) + (1/k)*hy2*m(714,021:696)*m(021:696,714);
m(716,040) = m(716,040) + (1/k)*hy2*m(715,021:696)*m(021:696,715);
```

```
function mfinal = arribita_p10(k,u1,u2,u1zero,u2zero)
mfinal = zeros(340,28);
mfinal(001:020,01:20) = k*eye(20);   mfinal(021:040,01:20) = k*eye(20);
mfinal(041,21) = k*u1zero;   mfinal(043,22) = k*u1(1);
mfinal(044,22) = k*u1(2);   mfinal(045,23) = k*u2zero;
mfinal(046,23) = k*u2(1);   mfinal(047,24) = k*u2(1);
mfinal(048,24) = k*u2(2);   mfinal(049,25) = k*u1(1);
mfinal(050,26) = k*u1(2);   mfinal(051,27) = k*u2(1);
mfinal(052,28) = k*u2(2);
mfinal(053:196,01:20) = arriba1_p10(k,u1); % 144 X 20
mfinal(197:340,01:20) = arriba1_p10(k,u2); % 144 X 20
```

```
function m = arriba1_p10(k,u)
m = zeros(144,20);
m(01,01) = k*u(01);   m(02,01) = k*u(01);   m(03,02) = k*u(01);
m(05,11) = k*u(01);   m(06,11) = k*u(01);   m(07,12) = k*u(01);
m(09,01) = k*u(01)*u(01);   m(10,02) = k*u(01)*u(01);
m(12,11) = k*u(01)*u(01);   m(13,12) = k*u(01)*u(01);   m(15,02) = k*u(02);
m(16,03) = k*u(02);   m(18,12) = k*u(02);   m(19,13) = k*u(02);
m(21,02) = k*u(02)*u(02);   m(22,03) = k*u(02)*u(02);
m(24,12) = k*u(02)*u(02);   m(25,13) = k*u(02)*u(02);   m(27,03) = k*u(03);
m(28,04) = k*u(03);   m(30,13) = k*u(03);   m(31,14) = k*u(03);
m(33,03) = k*u(03)*u(03);   m(34,04) = k*u(03)*u(03);
m(36,13) = k*u(03)*u(03);   m(37,14) = k*u(03)*u(03);   m(39,04) = k*u(04);
m(40,05) = k*u(04);   m(42,14) = k*u(04);   m(43,15) = k*u(04);
m(45,04) = k*u(04)*u(04);   m(46,05) = k*u(04)*u(04);
m(48,14) = k*u(04)*u(04);   m(49,15) = k*u(04)*u(04);   m(51,05) = k*u(05);
m(52,06) = k*u(05);   m(54,15) = k*u(05);   m(55,16) = k*u(05);
m(57,05) = k*u(05)*u(05);   m(58,06) = k*u(05)*u(05);
m(60,15) = k*u(05)*u(05);   m(61,16) = k*u(05)*u(05);   m(63,06) = k*u(06);
m(64,07) = k*u(06);   m(66,16) = k*u(06);   m(67,17) = k*u(06);
m(69,06) = k*u(06)*u(06);   m(70,07) = k*u(06)*u(06);
m(72,16) = k*u(06)*u(06);   m(73,17) = k*u(06)*u(06);   m(75,07) = k*u(07);
m(76,08) = k*u(07);   m(78,17) = k*u(07);   m(79,18) = k*u(07);
m(81,07) = k*u(07)*u(07);   m(82,08) = k*u(07)*u(07);
m(84,17) = k*u(07)*u(07);   m(85,18) = k*u(07)*u(07);   m(87,08) = k*u(08);
m(88,09) = k*u(08);   m(90,18) = k*u(08);   m(91,19) = k*u(08);
m(93,08) = k*u(08)*u(08);   m(94,09) = k*u(08)*u(08);
m(96,18) = k*u(08)*u(08);   m(97,19) = k*u(08)*u(08);   m(99,09) = k*u(09);
m(100,10) = k*u(09);   m(101,19) = k*u(09);   m(102,20) = k*u(09);
m(103,09) = k*u(09)*u(09);   m(104,10) = k*u(09)*u(09);
m(105,19) = k*u(09)*u(09);   m(106,20) = k*u(09)*u(09);
m(107,10) = k*u(10);   m(108,20) = k*u(10);   m(109,10) = k*u(10)*u(10);
m(110,20) = k*u(10)*u(10);   m(111,02) = k*u(01)*u(02);
m(113,12) = k*u(01)*u(02);   m(115,03) = k*u(02)*u(03);
m(117,13) = k*u(02)*u(03);   m(119,04) = k*u(03)*u(04);
m(121,14) = k*u(03)*u(04);   m(123,05) = k*u(04)*u(05);
m(125,15) = k*u(04)*u(05);   m(127,06) = k*u(05)*u(06);
m(129,16) = k*u(05)*u(06);   m(131,07) = k*u(06)*u(07);
```

```
m(133,17) = k*u(06)*u(07);   m(135,08) = k*u(07)*u(08);
m(137,18) = k*u(07)*u(08);   m(139,09) = k*u(08)*u(09);
m(141,19) = k*u(08)*u(09);   m(143,10) = k*u(09)*u(10);
m(144,20) = k*u(09)*u(10);
```

```
function m = bloquev1_p10(k)
m = zeros(76,288);
g1 = [002 , 015 , 027 , 039 , 051 , 063 , 075 , 087 , 099 , 107];
for i=1:10,    m(i,g1(i)) = k;   end
g2 = [003 , 016 , 028 , 040 , 052 , 064 , 076 , 088 , 100];
for i=1:9,    m(10+i,g2(i)) = k;   end
g3 = [006 , 018 , 030 , 042 , 054 , 066 , 078 , 090 , 101 , 108];
for i=1:10,    m(19+i,g3(i)) = k;   end
g4 = [007 , 019 , 031 , 043 , 055 , 067 , 079 , 091 , 102];
for i=1:9,    m(29+i,g4(i)) = k;   end
g5 = 144+g1;
for i=1:10,    m(38+i,g5(i)) = k;   end
g6 = 144+g2;
for i=1:9,    m(48+i,g6(i)) = k;   end
g7 = 144+g3;
for i=1:10,    m(57+i,g7(i)) = k;   end
g8 = 144+g4;
for i=1:9,    m(67+i,g8(i)) = k;   end
```

```
function m = bloquev2_p10(k)
m = zeros(76,288);
g1 = [009 , 021 , 033 , 045 , 057 , 069 , 081 , 093 , 103 , 109];
for i=1:10,    m(i,g1(i)) = k;   end
g2 = [010 , 022 , 034 , 046 , 058 , 070 , 082 , 094 , 104];
for i=1:9,    m(10+i,g2(i)) = k;   end
g3 = [012 , 024 , 036 , 048 , 060 , 072 , 084 , 096 , 105 , 110];
for i=1:10,    m(19+i,g3(i)) = k;   end
g4 = [013 , 025 , 037 , 049 , 061 , 073 , 085 , 097 , 106];
for i=1:9,    m(29+i,g4(i)) = k;   end
g5 = 144+g1;
for i=1:10,    m(38+i,g5(i)) = k;   end
g6 = 144+g2;
for i=1:9,    m(48+i,g6(i)) = k;   end
g7 = 144+g3;
for i=1:10,    m(57+i,g7(i)) = k;   end
g8 = 144+g4;
for i=1:9,    m(67+i,g8(i)) = k;   end
```

```
function m = bloquev3_p10(k)
m = zeros(68,288);
g1 = [003 , 016 , 028 , 040 , 052 , 064 , 076 , 088 , 100];
for i=1:9,    m(i,g1(i)) = k;   end
g2 = [004 , 017 , 029 , 041 , 053 , 065 , 077 , 089];
for i=1:8,    m(9+i,g2(i)) = k;   end
g3 = [007 , 019 , 031 , 043 , 055 , 067 , 079 , 091 , 102];
for i=1:9,    m(17+i,g3(i)) = k;   end
g4 = [008 , 020 , 032 , 044 , 056 , 068 , 080 , 092];
for i=1:8,    m(26+i,g4(i)) = k;   end
g5 = 144+g1;
for i=1:9,    m(34+i,g5(i)) = k;   end
g6 = 144+g2;
for i=1:8,    m(43+i,g6(i)) = k;   end
g7 = 144+g3;
for i=1:9,    m(51+i,g7(i)) = k;   end
g8 = 144+g4;
for i=1:8,    m(60+i,g8(i)) = k;   end
```

```
function m = bloquev4_p10(k)
m = zeros(68,288);
g1 = [010 , 022 , 034 , 046 , 058 , 070 , 082 , 094 , 104];
for i=1:9,    m(i,g1(i)) = k;   end
g2 = [011 , 023 , 035 , 047 , 059 , 071 , 083 , 095];
for i=1:8,    m(9+i,g2(i)) = k;   end
g3 = [013 , 025 , 037 , 049 , 061 , 073 , 085 , 097 , 106];
for i=1:9,    m(17+i,g3(i)) = k;   end
g4 = [014 , 026 , 038 , 050 , 062 , 074 , 086 , 098];
for i=1:8,    m(26+i,g4(i)) = k;   end
g5 = 144+g1;
for i=1:9,    m(34+i,g5(i)) = k;   end
g6 = 144+g2;
for i=1:8,    m(43+i,g6(i)) = k;   end
g7 = 144+g3;
for i=1:9,    m(51+i,g7(i)) = k;   end
g8 = 144+g4;
for i=1:8,    m(60+i,g8(i)) = k;   end
```

```
function m = bloquev7_p10(k)
m = zeros(68,288);
g1 = [111 , 115 , 119 , 123 , 127 , 131 , 135 , 139 , 143];
for i=1:9,    m(i,g1(i)) = k;   end
g2 = [112 , 116 , 120 , 124 , 128 , 132 , 136 , 140];
for i=1:8,    m(9+i,g2(i)) = k;   end
g3 = [113 , 117 , 121 , 125 , 129 , 133 , 137 , 141 , 144];
for i=1:9,    m(17+i,g3(i)) = k;   end
g4 = [114 , 118 , 122 , 126 , 130 , 134 , 138 , 142];
for i=1:8,    m(26+i,g4(i)) = k;   end
g5 = 144+g1;
for i=1:9,    m(34+i,g5(i)) = k;   end
g6 = 144+g2;
for i=1:8,    m(43+i,g6(i)) = k;   end
g7 = 144+g3;
for i=1:9,    m(51+i,g7(i)) = k;   end
g8 = 144+g4;
for i=1:8,    m(60+i,g8(i)) = k;   end
```

```
function mfinal = abajito_p10(ed,ic1,ic2,hll_11,hcp_11,hll_12,hcp_12,...
    hll_21,hcp_21,hll_22,hcp_22,u1zero,u1menos1,u2zero,u2menos1,esi,...
    du1,du2,k,u1r,u2r,hy1,hy2,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,cal_nom)
mfinal = zeros(28,724);
for i=1:20,   mfinal(i,i) = ed(i,1);   end
for i=01:10,   mfinal(i,20+i) = ic1(i,1);   end
for i=11:20,   mfinal(i,20+i) = ic2(i-10,1);   end
mfinal(21,41) = +du1(1);   mfinal(21,42) = -du1(1);
mfinal(22,43) = +du1(2);   mfinal(22,44) = -du1(2);
mfinal(23,45) = +du2(1);   mfinal(23,46) = -du2(1);
mfinal(24,47) = +du2(2);   mfinal(24,48) = -du2(2);
mfinal(25,49) = k/u1r(1);   mfinal(26,50) = k/u1r(2);
mfinal(27,51) = k/u2r(1);   mfinal(28,52) = k/u2r(2);
bny1nu1 = abajony1_p10(hll_11,hcp_11,u1zero); % 10 X 144
bny1nu2 = abajony1_p10(hll_12,hcp_12,u2zero); % 10 X 144
bny2nu1 = abajony2_p10(hll_21,hcp_21,u1zero); % 10 X 144
bny2nu2 = abajony2_p10(hll_22,hcp_22,u2zero); % 10 X 144
p1 = [bny1nu1 , bny1nu2];   p2 = [bny2nu1 , bny2nu2];   p3 = [p1 ; p2];
mfinal(01:20,053:340) = p3;
bvar = abajovp_p10(hy1,hy2,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,cal_nom);
```

```
mfinal(01:20,341:696) = bvar;
mfinal(01:28,697:724) = esi*eye(28);
```

```
function m = abajony1_p10(hll,hcp,uzero)
m = zeros(10,144);
m(01,001) = hcp(1,2)*uzero;   m(01,002) = hll(1,1);   m(02,003) = hll(1,2);
m(01,009) = hcp(1,1);   m(02,010) = hcp(2,2);   m(02,015) = hll(1,1);
m(03,016) = hll(1,2);   m(02,021) = hcp(1,1);   m(03,022) = hcp(2,2);
m(03,027) = hll(1,1);   m(04,028) = hll(1,2);   m(03,033) = hcp(1,1);
m(04,034) = hcp(2,2);   m(04,039) = hll(1,1);   m(05,040) = hll(1,2);
m(04,045) = hcp(1,1);   m(05,046) = hcp(2,2);   m(05,051) = hll(1,1);
m(06,052) = hll(1,2);   m(05,057) = hcp(1,1);   m(06,058) = hcp(2,2);
m(06,063) = hll(1,1);   m(07,064) = hll(1,2);   m(06,069) = hcp(1,1);
m(07,070) = hcp(2,2);   m(07,075) = hll(1,1);   m(08,076) = hll(1,2);
m(07,081) = hcp(1,1);   m(08,082) = hcp(2,2);   m(08,087) = hll(1,1);
m(09,088) = hll(1,2);   m(08,093) = hcp(1,1);   m(09,094) = hcp(2,2);
m(09,099) = hll(1,1);   m(10,100) = hll(1,2);   m(09,103) = hcp(1,1);
m(10,104) = hcp(2,2);   m(10,107) = hll(1,1);   m(10,109) = hcp(1,1);
m(02,111) = hcp(1,2);   m(03,115) = hcp(1,2);   m(04,119) = hcp(1,2);
m(05,123) = hcp(1,2);   m(06,127) = hcp(1,2);   m(07,131) = hcp(1,2);
m(08,135) = hcp(1,2);   m(09,139) = hcp(1,2);   m(10,143) = hcp(1,2);
```

```
function m = abajony2_p10(hll,hcp,uzero)
m = zeros(10,144);   m(01,005) = hcp(1,2)*uzero;   m(01,006) = hll(1,1);
m(02,007) = hll(1,2);   m(01,012) = hcp(1,1);   m(02,013) = hcp(2,2);
m(02,018) = hll(1,1);   m(03,019) = hll(1,2);   m(02,024) = hcp(1,1);
m(03,025) = hcp(2,2);   m(03,030) = hll(1,1);   m(04,031) = hll(1,2);
m(03,036) = hcp(1,1);   m(04,037) = hcp(2,2);   m(04,042) = hll(1,1);
m(05,043) = hll(1,2);   m(04,048) = hcp(1,1);   m(05,049) = hcp(2,2);
m(05,054) = hll(1,1);   m(06,055) = hll(1,2);   m(05,060) = hcp(1,1);
m(06,061) = hcp(2,2);   m(06,066) = hll(1,1);   m(07,067) = hll(1,2);
m(06,072) = hcp(1,1);   m(07,073) = hcp(2,2);   m(07,078) = hll(1,1);
m(08,079) = hll(1,2);   m(07,084) = hcp(1,1);   m(08,085) = hcp(2,2);
m(08,090) = hll(1,1);   m(09,091) = hll(1,2);   m(08,096) = hcp(1,1);
m(09,097) = hcp(2,2);   m(09,101) = hll(1,1);   m(10,102) = hll(1,2);
m(09,105) = hcp(1,1);   m(10,106) = hcp(2,2);   m(10,108) = hll(1,1);
m(10,110) = hcp(1,1);   m(02,113) = hcp(1,2);   m(03,117) = hcp(1,2);
m(04,121) = hcp(1,2);   m(05,125) = hcp(1,2);   m(06,129) = hcp(1,2);
m(07,133) = hcp(1,2);   m(08,137) = hcp(1,2);   m(09,141) = hcp(1,2);
m(10,144) = hcp(1,2);
```

```
function m = abajovp_p10(hy1,hy2,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,c_nom)
v1 = bloquev1a_p10(...
    hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom);
v2 = bloquev2a_p10(...
    hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom);
v3 = bloquev3a_p10(...
    hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom);
v4 = bloquev4a_p10(...
    hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom);
v7 = bloquev7a_p10(...
    hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom);
m = [ v1 , v2 , v3 , v4 , v7 ];
```

```
function m = bloquev1a_p10(hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom)
m = zeros(20,76);
m(01:10,01:10) =     vhll_11(1,1)*eye(10);
m(02:10,11:19) = hy1*vhll_11(1,1)*eye(09);
m(11:20,20:29) =     vhll_21(1,1)*eye(10);
m(12:20,30:38) = hy2*vhll_21(1,1)*eye(09);
```

```
m(01:10,39:48) =     vhll_12(1,1)*eye(10);
m(02:10,49:57) = hy1*vhll_12(1,1)*eye(09);
m(11:20,58:67) =     vhll_22(1,1)*eye(10);
m(12:20,68:76) = hy2*vhll_22(1,1)*eye(09);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev2a_p10(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,76);
m(01:10,01:10) =     vhcp_11(1,1)*eye(10);
m(02:10,11:19) = hy1*vhcp_11(1,1)*eye(09);
m(11:20,20:29) =     vhcp_21(1,1)*eye(10);
m(12:20,30:38) = hy2*vhcp_21(1,1)*eye(09);
m(01:10,39:48) =     vhcp_12(1,1)*eye(10);
m(02:10,49:57) = hy1*vhcp_12(1,1)*eye(09);
m(11:20,58:67) =     vhcp_22(1,1)*eye(10);
m(12:20,68:76) = hy2*vhcp_22(1,1)*eye(09);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev3a_p10(hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom)
m = zeros(20,68);
m(02:10,01:09) =     vhll_11(1,2)*eye(9);
m(03:10,10:17) = hy1*vhll_11(1,2)*eye(8);
m(12:20,18:26) =     vhll_21(1,2)*eye(9);
m(13:20,27:34) = hy2*vhll_21(1,2)*eye(8);
m(02:10,35:43) =     vhll_12(1,2)*eye(9);
m(03:10,44:51) = hy1*vhll_12(1,2)*eye(8);
m(12:20,52:60) =     vhll_22(1,2)*eye(9);
m(13:20,61:68) = hy2*vhll_22(1,2)*eye(8);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev4a_p10(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,68);
m(02:10,01:09) =     vhcp_11(2,2)*eye(9);
m(03:10,10:17) = hy1*vhcp_11(2,2)*eye(8);
m(12:20,18:26) =     vhcp_21(2,2)*eye(9);
m(13:20,27:34) = hy2*vhcp_21(2,2)*eye(8);
m(02:10,35:43) =     vhcp_12(2,2)*eye(9);
m(03:10,44:51) = hy1*vhcp_12(2,2)*eye(8);
m(12:20,52:60) =     vhcp_22(2,2)*eye(9);
m(13:20,61:68) = hy2*vhcp_22(2,2)*eye(8);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev7a_p10(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,68);
m(02:10,01:09) =     vhcp_11(1,2)*eye(9);
m(03:10,10:17) = hy1*vhcp_11(1,2)*eye(8);
m(12:20,18:26) =     vhcp_21(1,2)*eye(9);
m(13:20,27:34) = hy2*vhcp_21(1,2)*eye(8);
m(02:10,35:43) =     vhcp_12(1,2)*eye(9);
```

```
m(03:10,44:51) = hy1*vhcp_12(1,2)*eye(8);
m(12:20,52:60) =     vhcp_22(1,2)*eye(9);
m(13:20,61:68) = hy2*vhcp_22(1,2)*eye(8);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

Table I.5: List of programs to calculate $\mathbf{M}$ for MIMO case example with $M = 3$

Program Name	Description
dabmdentro.m	Calculate $\mathbf{M}$ with $n_u = 2$, $n_y = 2$ and $M = 3$
arribita.m	Internal Program $\mathbf{M}_{12}$
arriba1.m	Internal program $\mathbf{M}_{12}$
bloquev1.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
bloquev2.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
bloquev3.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
bloquev4.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
bloquev5.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
bloquev6.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
bloquev7.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
bloquev8.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
bloquev9.m	Internal program $\mathbf{M}_{21}$ ($\mathbf{M}_{\text{C1CCD}}$)
abajito.m	Internal program $\mathbf{M}_{21}$
abajony1.m	Internal program $\mathbf{M}_{21}$
abajony2.m	Internal program $\mathbf{M}_{21}$
abajovp.m	Internal program $\mathbf{M}_{21}$
bloquev1a.m	Internal program $\mathbf{M}_{21}$
bloquev2a.m	Internal program $\mathbf{M}_{21}$
bloquev3a.m	Internal program $\mathbf{M}_{21}$
bloquev4a.m	Internal program $\mathbf{M}_{21}$
bloquev5a.m	Internal program $\mathbf{M}_{21}$
bloquev6a.m	Internal program $\mathbf{M}_{21}$
bloquev7a.m	Internal program $\mathbf{M}_{21}$
bloquev8a.m	Internal program $\mathbf{M}_{21}$
bloquev9a.m	Internal program $\mathbf{M}_{21}$

```
function m = dabmdentro(esi,k,ed,u,h11_ll,h12_ll,h21_ll,h22_ll,...
    h11_cp,h12_cp,h21_cp,h22_cp,ic1,ic2,hy1,hy2,u1zero,u1menos1,...
    u2zero,u2menos1,du1,du2,u1r,u2r,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,vtc,cal_nom)
marriba = arribita(k,u(1:10,1),u(11:20,1),u1zero,u2zero); % 508 X 36
menmedio = [bloquev1(k) ; bloquev2(k) ; bloquev3(k) ; bloquev4(k) ; ...
    bloquev5(k) ; bloquev6(k) ; bloquev7(k) ; bloquev8(k) ; bloquev9(k) ];
mabajo = abajito(ed,ic1,ic2,h11_ll,h11_cp,h12_ll,h12_cp,h21_ll,h21_cp,...
    h22_ll,h22_cp,u1zero,u1menos1,u2zero,u2menos1,esi,du1,du2,...
    k,u1r,u2r,hy1,hy2,vhll_11,vhcp_11,vhll_12,vhcp_12,vhll_21,vhcp_21,...
    vhll_22,vhcp_22,cal_nom); % 38 X 1140
m = zeros(1140,1140);
```

```
m(0001 : 0508 , 1105 : 1140) = marriba;
m(0509 : 1104 , 0065 : 0508) = menmedio;
m(1105 : 1140 ,      :     ) = mabajo;
m(1106,022) = m(1106,022) + (1/k)*hy1*m(1105,021:1104)*m(021:1104,1105);
m(1107,023) = m(1107,023) + (1/k)*hy1*m(1106,021:1104)*m(021:1104,1106);
m(1108,024) = m(1108,024) + (1/k)*hy1*m(1107,021:1104)*m(021:1104,1107);
m(1109,025) = m(1109,025) + (1/k)*hy1*m(1108,021:1104)*m(021:1104,1108);
m(1110,026) = m(1110,026) + (1/k)*hy1*m(1109,021:1104)*m(021:1104,1109);
m(1111,027) = m(1111,027) + (1/k)*hy1*m(1110,021:1104)*m(021:1104,1110);
m(1112,028) = m(1112,028) + (1/k)*hy1*m(1111,021:1104)*m(021:1104,1111);
m(1113,029) = m(1113,029) + (1/k)*hy1*m(1112,021:1104)*m(021:1104,1112);
m(1114,030) = m(1114,030) + (1/k)*hy1*m(1113,021:1104)*m(021:1104,1113);
m(1116,032) = m(1116,032) + (1/k)*hy2*m(1115,021:1104)*m(021:1104,1115);
m(1117,033) = m(1117,033) + (1/k)*hy2*m(1116,021:1104)*m(021:1104,1116);
m(1118,034) = m(1118,034) + (1/k)*hy2*m(1117,021:1104)*m(021:1104,1117);
m(1119,035) = m(1119,035) + (1/k)*hy2*m(1118,021:1104)*m(021:1104,1118);
m(1120,036) = m(1120,036) + (1/k)*hy2*m(1119,021:1104)*m(021:1104,1119);
m(1121,037) = m(1121,037) + (1/k)*hy2*m(1120,021:1104)*m(021:1104,1120);
m(1122,038) = m(1122,038) + (1/k)*hy2*m(1121,021:1104)*m(021:1104,1121);
m(1123,039) = m(1123,039) + (1/k)*hy2*m(1122,021:1104)*m(021:1104,1122);
m(1124,040) = m(1124,040) + (1/k)*hy2*m(1123,021:1104)*m(021:1104,1123);
```

```
function mfinal = arribita(k,u1,u2,u1zero,u2zero)
mfinal = zeros(508,36);  mfinal(001:020,01:20) = k*eye(20);
mfinal(021:040,01:20) = k*eye(20);  mfinal(041,21) = k*u1zero;
mfinal(042,21) = k*u1(1);  mfinal(043,22) = k*u1(1);
mfinal(044,22) = k*u1(2);  mfinal(045,23) = k*u1(2);
mfinal(046,23) = k*u1(3);  mfinal(047,24) = k*u1(3);
mfinal(048,24) = k*u1(4);  mfinal(049,25) = k*u2zero;
mfinal(050,25) = k*u2(1);  mfinal(051,26) = k*u2(1);
mfinal(052,26) = k*u2(2);  mfinal(053,27) = k*u2(2);
mfinal(054,27) = k*u2(3);  mfinal(055,28) = k*u2(3);
mfinal(056,28) = k*u2(4);  mfinal(057,29) = k*u1(1);
mfinal(058,30) = k*u1(2);  mfinal(059,31) = k*u1(3);
mfinal(060,32) = k*u1(4);  mfinal(061,33) = k*u2(1);
mfinal(062,34) = k*u2(2);  mfinal(063,35) = k*u2(3);
mfinal(064,36) = k*u2(4);  mfinal(065:286,01:20) = arriba1(k,u1);
mfinal(287:508,01:20) = arriba1(k,u2);
```

```
function m = arriba1(k,u)
m = zeros(222,20);
m(001,01) = k*u(01); m(002,01) = k*u(01); m(003,01) = k*u(01);
m(004,02) = k*u(01); m(005,02) = k*u(01); m(006,03) = k*u(01);
m(008,11) = k*u(01); m(009,11) = k*u(01); m(010,11) = k*u(01);
m(011,12) = k*u(01); m(012,12) = k*u(01); m(013,13) = k*u(01);
m(015,01) = k*u(01)*u(01); m(016,02) = k*u(01)*u(01);
m(017,03) = k*u(01)*u(01); m(019,11) = k*u(01)*u(01);
m(020,12) = k*u(01)*u(01); m(021,13) = k*u(01)*u(01);
m(023,02) = k*u(02); m(024,02) = k*u(02); m(025,03) = k*u(02);
m(026,04) = k*u(02); m(028,12) = k*u(02); m(029,12) = k*u(02);
m(030,13) = k*u(02); m(031,14) = k*u(02); m(033,02) = k*u(02)*u(02);
m(034,03) = k*u(02)*u(02); m(035,04) = k*u(02)*u(02);
m(037,12) = k*u(02)*u(02); m(038,13) = k*u(02)*u(02);
m(039,14) = k*u(02)*u(02); m(041,03) = k*u(03); m(042,04) = k*u(03);
m(043,05) = k*u(03); m(045,13) = k*u(03); m(046,14) = k*u(03);
m(047,15) = k*u(03); m(049,03) = k*u(03)*u(03);
m(050,04) = k*u(03)*u(03); m(051,05) = k*u(03)*u(03);
m(053,13) = k*u(03)*u(03); m(054,14) = k*u(03)*u(03);
m(055,15) = k*u(03)*u(03); m(057,04) = k*u(04);
m(058,05) = k*u(04); m(059,06) = k*u(04); m(061,14) = k*u(04);
m(062,15) = k*u(04); m(063,16) = k*u(04); m(065,04) = k*u(04)*u(04);
```

```
m(066,05) = k*u(04)*u(04); m(067,06) = k*u(04)*u(04);
m(069,14) = k*u(04)*u(04); m(070,15) = k*u(04)*u(04);
m(071,16) = k*u(04)*u(04); m(073,05) = k*u(05); m(074,06) = k*u(05);
m(075,07) = k*u(05); m(077,15) = k*u(05); m(078,16) = k*u(05);
m(079,17) = k*u(05); m(081,05) = k*u(05)*u(05);
m(082,06) = k*u(05)*u(05); m(083,07) = k*u(05)*u(05);
m(085,15) = k*u(05)*u(05); m(086,16) = k*u(05)*u(05);
m(087,17) = k*u(05)*u(05); m(089,06) = k*u(06); m(090,07) = k*u(06);
m(091,08) = k*u(06); m(093,16) = k*u(06); m(094,17) = k*u(06);
m(095,18) = k*u(06); m(097,06) = k*u(06)*u(06);
m(098,07) = k*u(06)*u(06); m(099,08) = k*u(06)*u(06);
m(101,16) = k*u(06)*u(06); m(102,17) = k*u(06)*u(06);
m(103,18) = k*u(06)*u(06); m(105,07) = k*u(07);
m(106,08) = k*u(07); m(107,09) = k*u(07); m(109,17) = k*u(07);
m(110,18) = k*u(07); m(111,19) = k*u(07); m(113,07) = k*u(07)*u(07);
m(114,08) = k*u(07)*u(07); m(115,09) = k*u(07)*u(07);
m(117,17) = k*u(07)*u(07); m(118,18) = k*u(07)*u(07);
m(119,19) = k*u(07)*u(07); m(121,08) = k*u(08);
m(122,09) = k*u(08); m(123,10) = k*u(08); m(124,18) = k*u(08);
m(125,19) = k*u(08); m(126,20) = k*u(08); m(127,08) = k*u(08)*u(08);
m(128,09) = k*u(08)*u(08); m(129,10) = k*u(08)*u(08);
m(130,18) = k*u(08)*u(08); m(131,19) = k*u(08)*u(08);
m(132,20) = k*u(08)*u(08); m(133,09) = k*u(09); m(134,10) = k*u(09);
m(135,19) = k*u(09); m(136,20) = k*u(09); m(137,09) = k*u(09)*u(09);
m(138,10) = k*u(09)*u(09); m(139,19) = k*u(09)*u(09);
m(140,20) = k*u(09)*u(09); m(141,10) = k*u(10); m(142,20) = k*u(10);
m(143,10) = k*u(10)*u(10); m(144,20) = k*u(10)*u(10);
m(145,02) = k*u(01)*u(02); m(146,03) = k*u(01)*u(02);
m(148,12) = k*u(01)*u(02); m(149,13) = k*u(01)*u(02);
m(151,03) = k*u(02)*u(03); m(152,04) = k*u(02)*u(03);
m(154,13) = k*u(02)*u(03); m(155,14) = k*u(02)*u(03);
m(157,04) = k*u(03)*u(04); m(158,05) = k*u(03)*u(04);
m(160,14) = k*u(03)*u(04); m(161,15) = k*u(03)*u(04);
m(163,05) = k*u(04)*u(05); m(164,06) = k*u(04)*u(05);
m(166,15) = k*u(04)*u(05); m(167,16) = k*u(04)*u(05);
m(169,06) = k*u(05)*u(06); m(170,07) = k*u(05)*u(06);
m(172,16) = k*u(05)*u(06); m(173,17) = k*u(05)*u(06);
m(175,07) = k*u(06)*u(07); m(176,08) = k*u(06)*u(07);
m(178,17) = k*u(06)*u(07); m(179,18) = k*u(06)*u(07);
m(181,08) = k*u(07)*u(08); m(182,09) = k*u(07)*u(08);
m(184,18) = k*u(07)*u(08); m(185,19) = k*u(07)*u(08);
m(187,09) = k*u(08)*u(09); m(188,10) = k*u(08)*u(09);
m(189,19) = k*u(08)*u(09); m(190,20) = k*u(08)*u(09);
m(191,10) = k*u(09)*u(10); m(192,20) = k*u(09)*u(10);
m(193,03) = k*u(01)*u(03); m(195,13) = k*u(01)*u(03);
m(197,04) = k*u(02)*u(04); m(199,14) = k*u(02)*u(04);
m(201,05) = k*u(03)*u(05); m(203,15) = k*u(03)*u(05);
m(205,06) = k*u(04)*u(06); m(207,16) = k*u(04)*u(06);
m(209,07) = k*u(05)*u(07); m(211,17) = k*u(05)*u(07);
m(213,08) = k*u(06)*u(08); m(215,18) = k*u(06)*u(08);
m(217,09) = k*u(07)*u(09); m(219,19) = k*u(07)*u(09);
m(221,10) = k*u(08)*u(10); m(222,20) = k*u(08)*u(10);
```

```
function m = bloquev1(k)
m = zeros(76,444);
g1 = [003 , 024 , 041 , 057 , 073 , 089 , 105 , 121 , 133 , 141];
for i=1:10, m(i,g1(i)) = k;  end
g2 = [005 , 025 , 042 , 058 , 074 , 090 , 106 , 122 , 134];
for i=1:9,  m(10+i,g2(i)) = k;  end
g3 = [010 , 029 , 045 , 061 , 077 , 093 , 109 , 124 , 135 , 142];
for i=1:10,  m(19+i,g3(i)) = k;  end
g4 = [012 , 030 , 046 , 062 , 078 , 094 , 110 , 125 , 136];
```

```
for i=1:9,  m(29+i,g4(i)) = k;  end
g5 = 222+g1;  for i=1:10,  m(38+i,g5(i)) = k;  end
g6 = 222+g2;  for i=1:9,  m(48+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:10,  m(57+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:9,  m(67+i,g8(i)) = k;  end
```

```
function m = bloquev2(k)
m = zeros(76,444);
g1 = [015 , 033 , 049 , 065 , 081 , 097 , 113 , 127 , 137 , 143];
for i=1:10,  m(i,g1(i)) = k;  end
g2 = [016 , 034 , 050 , 066 , 082 , 098 , 114 , 128 , 138];
for i=1:9,  m(10+i,g2(i)) = k;  end
g3 = [019 , 037 , 053 , 069 , 085 , 101 , 117 , 130 , 139 , 144];
for i=1:10,  m(19+i,g3(i)) = k;   end
g4 = [020 , 038 , 054 , 070 , 086 , 102 , 118 , 131 , 140];
for i=1:9,  m(29+i,g4(i)) = k;  end
g5 = 222+g1;  for i=1:10,  m(38+i,g5(i)) = k;  end
g6 = 222+g2;  for i=1:9,  m(48+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:10,  m(57+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:9,  m(67+i,g8(i)) = k;  end
```

```
function m = bloquev3(k)
m = zeros(68,444);
g1 = [005 , 025 , 042 , 058 , 074 , 090 , 106 , 122 , 134];
for i=1:9,  m(i,g1(i)) = k;  end
g2 = [006 , 026 , 043 , 059 , 075 , 091 , 107 , 123];
for i=1:8,  m(9+i,g2(i)) = k;  end
g3 = [012 , 030 , 046 , 062 , 078 , 094 , 110 , 125 , 136];
for i=1:9,  m(17+i,g3(i)) = k;  end
g4 = [013 , 031 , 047 , 063 , 079 , 095 , 111 , 126];
for i=1:8,  m(26+i,g4(i)) = k;  end
g5 = 222+g1;  for i=1:9,  m(34+i,g5(i)) = k;  end
g6 = 222+g2;  for i=1:8,  m(43+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:9,  m(51+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:8,  m(60+i,g8(i)) = k;  end
```

```
function m = bloquev4(k)
m = zeros(68,444);
g1 = [016 , 034 , 050 , 066 , 082 , 098 , 114 , 128 , 138];
for i=1:9,  m(i,g1(i)) = k;  end
g2 = [017 , 035 , 051 , 067 , 083 , 099 , 115 , 129];
for i=1:8,  m(9+i,g2(i)) = k;  end
g3 = [020 , 038 , 054 , 070 , 086 , 102 , 118 , 131 , 140];
for i=1:9,  m(17+i,g3(i)) = k;  end
g4 = [021 , 039 , 055 , 071 , 087 , 103 , 119 , 132];
for i=1:8,  m(26+i,g4(i)) = k;  end
g5 = 222+g1;  for i=1:9,  m(34+i,g5(i)) = k;  end
g6 = 222+g2;  for i=1:8,  m(43+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:9,  m(51+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:8,  m(60+i,g8(i)) = k;  end
```

```
function m = bloquev5(k)
m = zeros(60,444);
g1 = [006 , 026 , 043 , 059 , 075 , 091 , 107 , 123];
for i=1:8,  m(i,g1(i)) = k;  end
g2 = [007 , 027 , 044 , 060 , 076 , 092 , 108];
for i=1:7,  m(8+i,g2(i)) = k;  end
g3 = [013 , 031 , 047 , 063 , 079 , 095 , 111 , 126];
for i=1:8,  m(15+i,g3(i)) = k;  end
g4 = [014 , 032 , 048 , 064 , 080 , 096 , 112];
for i=1:7,  m(23+i,g4(i)) = k;  end
```

```
g5 = 222+g1;  for i=1:8,  m(30+i,g5(i)) = k;  end
g6 = 222+g2;  for i=1:7,  m(38+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:8,  m(45+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:7,  m(53+i,g8(i)) = k;  end
```

```
function m = bloquev6(k)
m = zeros(60,444);
g1 = [017 , 035 , 051 , 067 , 083 , 099 , 115 , 129];
for i=1:8,  m(i,g1(i)) = k;  end
g2 = [018 , 036 , 052 , 068 , 084 , 100 , 116];
for i=1:7,  m(8+i,g2(i)) = k;  end
g3 = [021 , 039 , 055 , 071 , 087 , 103 , 119 , 132];
for i=1:8,  m(15+i,g3(i)) = k;  end
g4 = [022 , 040 , 056 , 072 , 088 , 104 , 120];
for i=1:7,  m(23+i,g4(i)) = k;  end
g5 = 222+g1;  for i=1:8,  m(30+i,g5(i)) = k;  end
g6 = 222+g2;  for i=1:7,  m(38+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:8,  m(45+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:7,  m(53+i,g8(i)) = k;  end
```

```
function m = bloquev7(k)
m = zeros(68,444);
g1 = [145 , 151 , 157 , 163 , 169 , 175 , 181 , 187 , 191];
for i=1:9,  m(i,g1(i)) = k;  end
g2 = [146 , 152 , 158 , 164 , 170 , 176 , 182 , 188];
for i=1:8,  m(9+i,g2(i)) = k;  end
g3 = [148 , 154 , 160 , 166 , 172 , 178 , 184 , 189 , 192];
for i=1:9,  m(17+i,g3(i)) = k;  end
g4 = [149 , 155 , 161 , 167 , 173 , 179 , 185 , 190];
for i=1:8,  m(26+i,g4(i)) = k;  end
g5 = 222+g1;  for i=1:9,  m(34+i,g5(i)) = k;  end
g6 = 222+g2;  for i=1:8,  m(43+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:9,  m(51+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:8,  m(60+i,g8(i)) = k;  end
```

```
function m = bloquev8(k)
m = zeros(60,444);
g1 = [193 , 197 , 201 , 205 , 209 , 213 , 217 , 221];
for i=1:8,  m(i,g1(i)) = k;  end
g2 = [194 , 198 , 202 , 206 , 210 , 214 , 218];
for i=1:7,  m(8+i,g2(i)) = k;  end
g3 = [195 , 199 , 203 , 207 , 211 , 215 , 219 , 222];
for i=1:8,  m(15+i,g3(i)) = k;  end
g4 = [196 , 200 , 204 , 208 , 212 , 216 , 220];
for i=1:7,  m(23+i,g4(i)) = k;  end
g5 = 222+g1;  for i=1:8,  m(30+i,g5(i)) = k;  end
g6 = 222+g2;  for i=1:7,  m(38+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:8,  m(45+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:7,  m(53+i,g8(i)) = k;  end
```

```
function m = bloquev9(k)
m = zeros(60,444);
g1 = [146 , 152 , 158 , 164 , 170 , 176 , 182 , 188];
for i=1:8,  m(i,g1(i)) = k;  end
g2 = [147 , 153 , 159 , 165 , 171 , 177 , 183];
for i=1:7,  m(8+i,g2(i)) = k;  end
g3 = [149 , 155 , 161 , 167 , 173 , 179 , 185 , 190];
for i=1:8,  m(15+i,g3(i)) = k;  end
g4 = [150 , 156 , 162 , 168 , 174 , 180 , 186];
for i=1:7,  m(23+i,g4(i)) = k;  end
g5 = 222+g1;  for i=1:8,  m(30+i,g5(i)) = k;  end
```

```
g6 = 222+g2;  for i=1:7,  m(38+i,g6(i)) = k;  end
g7 = 222+g3;  for i=1:8,  m(45+i,g7(i)) = k;  end
g8 = 222+g4;  for i=1:7,  m(53+i,g8(i)) = k;  end
```

```
function mfinal = abajito(ed,ic1,ic2,hll_11,hcp_11,hll_12,hcp_12,...
    hll_21,hcp_21,hll_22,hcp_22,u1zero,u1menos1,u2zero,u2menos1,esi,...
    du1,du2,k,u1r,u2r,hy1,hy2,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,cal_nom)
mfinal = zeros(36,1140);
for i=1:20,  mfinal(i,i) = ed(i,1);  end
for i=1:10,  mfinal(i,20+i) = ic1(i,1);  end
for i=11:20,  mfinal(i,20+i) = ic2(i-10,1);  end
mfinal(21,41) = +du1(1);  mfinal(21,42) = -du1(1);
mfinal(22,43) = +du1(2);  mfinal(22,44) = -du1(2);
mfinal(23,45) = +du1(3);  mfinal(23,46) = -du1(3);
mfinal(24,47) = +du1(4);  mfinal(24,48) = -du1(4);
mfinal(25,49) = +du2(1);  mfinal(25,50) = -du2(1);
mfinal(26,51) = +du2(2);  mfinal(26,52) = -du2(2);
mfinal(27,53) = +du2(3);  mfinal(27,54) = -du2(3);
mfinal(28,55) = +du2(4);  mfinal(28,56) = -du2(4);
mfinal(29,57) = k/u1r(1);  mfinal(30,58) = k/u1r(2);
mfinal(31,59) = k/u1r(3);  mfinal(32,60) = k/u1r(4);
mfinal(33,61) = k/u2r(1);  mfinal(34,62) = k/u2r(2);
mfinal(35,63) = k/u2r(3);  mfinal(36,64) = k/u2r(4);
bny1nu1 = abajony1(hll_11,hcp_11,u1zero,u1menos1); % 10 X 222
bny1nu2 = abajony1(hll_12,hcp_12,u2zero,u2menos1); % 10 X 222
bny2nu1 = abajony2(hll_21,hcp_21,u1zero,u1menos1); % 10 X 222
bny2nu2 = abajony2(hll_22,hcp_22,u2zero,u2menos1); % 10 X 222
p1 = [bny1nu1 , bny1nu2];  p2 = [bny2nu1 , bny2nu2];  p3 = [p1 ; p2];
mfinal(01:20,065:508) = p3;
bvar = abajovp(hy1,hy2,vhll_11,vhcp_11,vhll_12,vhcp_12,vhll_21,vhcp_21,...
    vhll_22,vhcp_22,cal_nom);  mfinal(01:20,509:1104) = bvar;
mfinal(01:36,1105:1140) = esi*eye(36);
```

```
function m = abajony1(hll,hcp,uzero,umenos1)
m = zeros(10,222);
m(01,001) = hcp(1,3)*umenos1;  m(01,002) = hcp(1,2)*uzero;
m(01,003) = hll(1,1);  m(02,004) = hcp(2,3)*uzero;  m(02,005) = hll(1,2);
m(03,006) = hll(1,3);  m(01,015) = hcp(1,1);  m(02,016) = hcp(2,2);
m(03,017) = hcp(3,3);  m(02,023) = hcp(1,3)*uzero;  m(02,024) = hll(1,1);
m(03,025) = hll(1,2);  m(04,026) = hll(1,3);  m(02,033) = hcp(1,1);
m(03,034) = hcp(2,2);  m(04,035) = hcp(3,3);  m(03,041) = hll(1,1);
m(04,042) = hll(1,2);  m(05,043) = hll(1,3);  m(03,049) = hcp(1,1);
m(04,050) = hcp(2,2);  m(05,051) = hcp(3,3);  m(04,057) = hll(1,1);
m(05,058) = hll(1,2);  m(06,059) = hll(1,3);  m(04,065) = hcp(1,1);
m(05,066) = hcp(2,2);  m(06,067) = hcp(3,3);  m(05,073) = hll(1,1);
m(06,074) = hll(1,2);  m(07,075) = hll(1,3);  m(05,081) = hcp(1,1);
m(06,082) = hcp(2,2);  m(07,083) = hcp(3,3);  m(06,089) = hll(1,1);
m(07,090) = hll(1,2);  m(08,091) = hll(1,3);  m(06,097) = hcp(1,1);
m(07,098) = hcp(2,2);  m(08,099) = hcp(3,3);  m(07,105) = hll(1,1);
m(08,106) = hll(1,2);  m(09,107) = hll(1,3);  m(07,113) = hcp(1,1);
m(08,114) = hcp(2,2);  m(09,115) = hcp(3,3);  m(08,121) = hll(1,1);
m(09,122) = hll(1,2);  m(10,123) = hll(1,3);  m(08,127) = hcp(1,1);
m(09,128) = hcp(2,2);  m(10,129) = hcp(3,3);  m(09,133) = hll(1,1);
m(10,134) = hll(1,2);  m(09,137) = hcp(1,1);  m(10,138) = hcp(2,2);
m(10,141) = hll(1,1);  m(10,143) = hcp(1,1);  m(02,145) = hcp(1,2);
m(03,146) = hcp(2,3);  m(03,151) = hcp(1,2);  m(04,152) = hcp(2,3);
m(04,157) = hcp(1,2);  m(05,158) = hcp(2,3);  m(05,163) = hcp(1,2);
m(06,164) = hcp(2,3);  m(06,169) = hcp(1,2);  m(07,170) = hcp(2,3);
m(07,175) = hcp(1,2);  m(08,176) = hcp(2,3);  m(08,181) = hcp(1,2);
m(09,182) = hcp(2,3);  m(09,187) = hcp(1,2);  m(10,188) = hcp(2,3);
```

```
m(10,191) = hcp(1,2);  m(03,193) = hcp(1,3);  m(04,197) = hcp(1,3);
m(05,201) = hcp(1,3);  m(06,205) = hcp(1,3);  m(07,209) = hcp(1,3);
m(08,213) = hcp(1,3);  m(09,217) = hcp(1,3);  m(10,221) = hcp(1,3);
```

```
function m = abajony2(hll,hcp,uzero,umenos1)
m = zeros(10,222);  m(01,008) = hcp(1,3)*umenos1;
m(01,009) = hcp(1,2)*uzero;  m(01,010) = hll(1,1);
m(02,011) = hcp(2,3)*uzero;  m(02,012) = hll(1,2);  m(03,013) = hll(1,3);
m(01,019) = hcp(1,1);  m(02,020) = hcp(2,2);  m(03,021) = hcp(3,3);
m(02,028) = hcp(1,3)*uzero;  m(02,029) = hll(1,1);  m(03,030) = hll(1,2);
m(04,031) = hll(1,3);  m(02,037) = hcp(1,1);  m(03,038) = hcp(2,2);
m(04,039) = hcp(3,3);  m(03,045) = hll(1,1);  m(04,046) = hll(1,2);
m(05,047) = hll(1,3);  m(03,053) = hcp(1,1);  m(04,054) = hcp(2,2);
m(05,055) = hcp(3,3);  m(04,061) = hll(1,1);  m(05,062) = hll(1,2);
m(06,063) = hll(1,3);  m(04,069) = hcp(1,1);  m(05,070) = hcp(2,2);
m(06,071) = hcp(3,3);  m(05,077) = hll(1,1);  m(06,078) = hll(1,2);
m(07,079) = hll(1,3);  m(05,085) = hcp(1,1);  m(06,086) = hcp(2,2);
m(07,087) = hcp(3,3);  m(06,093) = hll(1,1);  m(07,094) = hll(1,2);
m(08,095) = hll(1,3);  m(06,101) = hcp(1,1);  m(07,102) = hcp(2,2);
m(08,103) = hcp(3,3);  m(07,109) = hll(1,1);  m(08,110) = hll(1,2);
m(09,111) = hll(1,3);  m(07,117) = hcp(1,1);  m(08,118) = hcp(2,2);
m(09,119) = hcp(3,3);  m(08,124) = hll(1,1);  m(09,125) = hll(1,2);
m(10,126) = hll(1,3);  m(08,130) = hcp(1,1);  m(09,131) = hcp(2,2);
m(10,132) = hcp(3,3);  m(09,135) = hll(1,1);  m(10,136) = hll(1,2);
m(09,139) = hcp(1,1);  m(10,140) = hcp(2,2);  m(10,142) = hll(1,1);
m(10,144) = hcp(1,1);  m(02,148) = hcp(1,2);  m(03,149) = hcp(2,3);
m(03,154) = hcp(1,2);  m(04,155) = hcp(2,3);  m(04,160) = hcp(1,2);
m(05,161) = hcp(2,3);  m(05,166) = hcp(1,2);  m(06,167) = hcp(2,3);
m(06,172) = hcp(1,2);  m(07,173) = hcp(2,3);  m(07,178) = hcp(1,2);
m(08,179) = hcp(2,3);  m(08,184) = hcp(1,2);  m(09,185) = hcp(2,3);
m(09,189) = hcp(1,2);  m(10,190) = hcp(2,3);  m(10,192) = hcp(1,2);
m(03,195) = hcp(1,3);  m(04,199) = hcp(1,3);  m(05,203) = hcp(1,3);
m(06,207) = hcp(1,3);  m(07,211) = hcp(1,3);  m(08,215) = hcp(1,3);
m(09,219) = hcp(1,3);  m(10,222) = hcp(1,3);
```

```
function m = abajovp(hy1,hy2,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,c_nom)
v1 = bloquev1a(hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom); % 20 X 76
v2 = bloquev2a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom); % 20 X 76
v3 = bloquev3a(hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom); % 20 X 68
v4 = bloquev4a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom); % 20 X 68
v5 = bloquev5a(hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom); % 20 X 60
v6 = bloquev6a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom); % 20 X 60
v7 = bloquev7a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom); % 20 X 68
v8 = bloquev8a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom); % 20 X 60
v9 = bloquev9a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom); % 20 X 60
m = [v1 , v2 , v3 , v4 , v5 , v6 , v7 , v8 , v9];
```

```
function m = bloquev1a(hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom)
m = zeros(20,76);
m(01:10,001:010) =     vhll_11(1,1)*eye(10);
m(02:10,011:019) = hy1*vhll_11(1,1)*eye(09);
m(11:20,020:029) =     vhll_21(1,1)*eye(10);
m(12:20,030:038) = hy2*vhll_21(1,1)*eye(09);
m(01:10,039:048) =     vhll_12(1,1)*eye(10);
m(02:10,049:057) = hy1*vhll_12(1,1)*eye(09);
m(11:20,058:067) =     vhll_22(1,1)*eye(10);
m(12:20,068:076) = hy2*vhll_22(1,1)*eye(09);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev2a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,76);
m(01:10,001:010) =     vhcp_11(1,1)*eye(10);
m(02:10,011:019) = hy1*vhcp_11(1,1)*eye(09);
m(11:20,020:029) =     vhcp_21(1,1)*eye(10);
m(12:20,030:038) = hy2*vhcp_21(1,1)*eye(09);
m(01:10,039:048) =     vhcp_12(1,1)*eye(10);
m(02:10,049:057) = hy1*vhcp_12(1,1)*eye(09);
m(11:20,058:067) =     vhcp_22(1,1)*eye(10);
m(12:20,068:076) = hy2*vhcp_22(1,1)*eye(09);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev3a(hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom)
m = zeros(20,68);
m(02:10,001:009) =     vhll_11(1,2)*eye(9);
m(03:10,010:017) = hy1*vhll_11(1,2)*eye(8);
m(12:20,018:026) =     vhll_21(1,2)*eye(9);
m(13:20,027:034) = hy2*vhll_21(1,2)*eye(8);
m(02:10,035:043) =     vhll_12(1,2)*eye(9);
m(03:10,044:051) = hy1*vhll_12(1,2)*eye(8);
m(12:20,052:060) =     vhll_22(1,2)*eye(9);
m(13:20,061:068) = hy2*vhll_22(1,2)*eye(8);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev4a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,68);
m(02:10,001:009) =     vhcp_11(2,2)*eye(9);
m(03:10,010:017) = hy1*vhcp_11(2,2)*eye(8);
m(12:20,018:026) =     vhcp_21(2,2)*eye(9);
m(13:20,027:034) = hy2*vhcp_21(2,2)*eye(8);
m(02:10,035:043) =     vhcp_12(2,2)*eye(9);
m(03:10,044:051) = hy1*vhcp_12(2,2)*eye(8);
m(12:20,052:060) =     vhcp_22(2,2)*eye(9);
m(13:20,061:068) = hy2*vhcp_22(2,2)*eye(8);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev5a(hy1,hy2,vhll_11,vhll_12,vhll_21,vhll_22,c_nom)
m = zeros(20,60);
m(03:10,001:008) =     vhll_11(1,3)*eye(8);
m(04:10,009:015) = hy1*vhll_11(1,3)*eye(7);
m(13:20,016:023) =     vhll_21(1,3)*eye(8);
m(14:20,024:030) = hy2*vhll_21(1,3)*eye(7);
m(03:10,031:038) =     vhll_12(1,3)*eye(8);
m(04:10,039:045) = hy1*vhll_12(1,3)*eye(7);
m(13:20,046:053) =     vhll_22(1,3)*eye(8);
m(14:20,054:060) = hy2*vhll_22(1,3)*eye(7);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev6a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,60);
m(03:10,001:008) =     vhcp_11(3,3)*eye(8);
m(04:10,009:015) = hy1*vhcp_11(3,3)*eye(7);
m(13:20,016:023) =     vhcp_21(3,3)*eye(8);
m(14:20,024:030) = hy2*vhcp_21(3,3)*eye(7);
m(03:10,031:038) =     vhcp_12(3,3)*eye(8);
m(04:10,039:045) = hy1*vhcp_12(3,3)*eye(7);
m(13:20,046:053) =     vhcp_22(3,3)*eye(8);
m(14:20,054:060) = hy2*vhcp_22(3,3)*eye(7);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev7a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,68);
m(02:10,001:009) =     vhcp_11(1,2)*eye(9);
m(03:10,010:017) = hy1*vhcp_11(1,2)*eye(8);
m(12:20,018:026) =     vhcp_21(1,2)*eye(9);
m(13:20,027:034) = hy2*vhcp_21(1,2)*eye(8);
m(02:10,035:043) =     vhcp_12(1,2)*eye(9);
m(03:10,044:051) = hy1*vhcp_12(1,2)*eye(8);
m(12:20,052:060) =     vhcp_22(1,2)*eye(9);
m(13:20,061:068) = hy2*vhcp_22(1,2)*eye(8);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev8a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,60);
m(03:10,001:008) =     vhcp_11(1,3)*eye(8);
m(04:10,009:015) = hy1*vhcp_11(1,3)*eye(7);
m(13:20,016:023) =     vhcp_21(1,3)*eye(8);
m(14:20,024:030) = hy2*vhcp_21(1,3)*eye(7);
m(03:10,031:038) =     vhcp_12(1,3)*eye(8);
m(04:10,039:045) = hy1*vhcp_12(1,3)*eye(7);
m(13:20,046:053) =     vhcp_22(1,3)*eye(8);
m(14:20,054:060) = hy2*vhcp_22(1,3)*eye(7);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

```
function m = bloquev9a(hy1,hy2,vhcp_11,vhcp_12,vhcp_21,vhcp_22,c_nom)
m = zeros(20,60);
m(03:10,001:008) =     vhcp_11(2,3)*eye(8);
m(04:10,009:015) = hy1*vhcp_11(2,3)*eye(7);
m(13:20,016:023) =     vhcp_21(2,3)*eye(8);
m(14:20,024:030) = hy2*vhcp_21(2,3)*eye(7);
m(03:10,031:038) =     vhcp_12(2,3)*eye(8);
m(04:10,039:045) = hy1*vhcp_12(2,3)*eye(7);
m(13:20,046:053) =     vhcp_22(2,3)*eye(8);
m(14:20,054:060) = hy2*vhcp_22(2,3)*eye(7);
if c_nom == 1
    m(01:09,:) = 0*m(01:09,:);
    m(11:19,:) = 0*m(11:19,:);
end
```

Table I.6: List of programs to calculate μ-test

Program Name	Description
main_mu.m	Main program used to calculate the cost function
che_int_bis.m	Internal program (calcute bounds for k_{ssv})
fkssv.m	Finds k_{ssv} such that $\mu_{\mathbf{\Delta}}(\mathbf{M}) \geq k_{\text{ssv}}$
dasigno.m	Internal Program
res_bratz_vf0.m	Internal Program

```
% Program used to calculate the Control Law for the
% Robust Volterra series based NMPC

% Information required for the program

% fv        = initial guess value for k_ssv
% blk       = matrix describing the uncertainty structure (used by matlab
%             function "mussv")
% ed        = vector of feedback for ny=1 and ny=2
% kuig      = vector of initial guesses for the manipulated variables
% h11_ll    = nominal value of the linear Volterra series coefficients for
%             ny=1 and nu=1
% h12_ll    = nominal value of the linear Volterra series coefficients for
%             ny=1 and nu=2
% h21_ll    = nominal value of the linear Volterra series coefficients for
%             ny=2 and nu=1
% h22_ll    = nominal value of the linear Volterra series coefficients for
%             ny=2 and nu=2
% h11_cp    = nominal value of the nonlinear Volterra series coefficients
%             for ny=1 and nu=1
% h12_cp    = nominal value of the nonlinear Volterra series coefficients
%             for ny=1 and nu=2
% h21_cp    = nominal value of the nonlinear Volterra series coefficients
%             for ny=2 and nu=1
% h22_cp    = nominal value of the nonlinear Volterra series coefficients
%             for ny=2 and nu=2
% ic1       = vector of initial conditions for ny=1
% ic2       = vector of initial conditions for ny=2
% hy1       = autoregressive Volterra series coefficient for ny=1
% hy2       = autoregressive Volterra series coefficient for ny=2
% u1zero    = manipulated variable value at u(k-1) for nu=1
% u1menos1  = manipulated variable at sambling instant u(k-2) for nu=1
% u2zero    = manipulated variable value at u(k-1) for nu=2
% u2menos1  = manipulated variable at sambling instant u(k-2) for nu=2
% du1       = vector of manipulated variable weighting movements for nu=1
% du2       = vector of manipulated variable weighting movements for nu=2
% u1r       = vector of manipulated variables bounds for nu=1
% u2r       = vector of manipulated variables bounds for nu=2
% vh11_ll   = uncertain value of the linear Volterra series coefficients
%             for ny=1 and nu=1
% vh12_ll   = uncertain value of the linear Volterra series coefficients
%             for ny=1 and nu=2
% vh21_ll   = uncertain value of the linear Volterra series coefficients
%             for ny=2 and nu=1
% vh22_ll   = uncertain value of the linear Volterra series coefficients
%             for ny=2 and nu=2
% vh11_cp   = uncertain value of the nonlinear Volterra series coefficients
%             for ny=1 and nu=1
% vh12_cp   = uncertain value of the nonlinear Volterra series coefficients
%             for ny=1 and nu=2
% vh21_cp  = uncertain value of the nonlinear Volterra series coefficients
%             for ny=2 and nu=1
% vh22_cp  = uncertain value of the nonlinear Volterra series coefficients
%             for ny=2 and nu=2
% vtc      = vector of terminal conditions, value at y(k+p) for
%             ny=1 and ny=2

esi = 1E-5;
cal_nom = 0;

opmu = 'd'; % Options for the Matlab program mussv
%              (possible options "fd", "fdU")
f_mu = @fminsearch; % Matlab optimization function
%                      (possible options "fminsearch", "fminunc", "fmincon")

opvar = optimset(f_mu);
opvar.Display = 'iter';

%Preliminary search for k_ssv

[c_l,li,ls] = che_int_bis(fv,blk,esi,ed,kuig,...
    h11_ll,h12_ll,h21_ll,h22_ll,h11_cp,h12_cp,h21_cp,h22_cp,ic1,ic2,...
    hy1,hy2,u1zero,u1menos1,u2zero,u2menos1,du1,du2,u1r,u2r,...
    vhll_11,vhcp_11,vhll_12,vhcp_12,vhll_21,vhcp_21,vhll_22,vhcp_22,...
    vtc,cal_nom,opmu);

if c_l == 0
    error('Under␣the␣current␣conditions␣the␣problem␣is␣unfeasible')
else
    factorcillo = 1.00;
end

[umu,rek] = f_mu(@(ku)fkssv(li,ls,blk,esi,ed,ku,...
    h11_ll,h12_ll,h21_ll,h22_ll,h11_cp,h12_cp,h21_cp,h22_cp,...
    ic1,ic2,hy1,hy2,u1zero,u1menos1,u2zero,u2menos1,du1,du2,u1r,u2r,...
    vhll_11,vhcp_11,vhll_12,vhcp_12,vhll_21,vhcp_21,vhll_22,vhcp_22,...
    vtc,cal_nom,opmu),factorcillo*kuig,opvar);
```

```
function [cal,li,ls] = che_int_bis(iguess,blk,esi,ed,uu,...
    h11_ll,h12_ll,h21_ll,h22_ll,h11_cp,h12_cp,h21_cp,h22_cp,...
    ic1,ic2,hy1,hy2,u1zero,u1menos1,u2zero,u2menos1,du1,du2,u1r,u2r,...
    vhll_11,vhcp_11,vhll_12,vhcp_12,vhll_21,vhcp_21,vhll_22,vhcp_22,...
    vtc,cal_nom,opmu)
if iguess == 0,     iguess = 1E-5;   end
for inj = 1:2
    if inj == 1
        gfzero = [ iguess - iguess*0.99 , iguess + iguess*0.99 ];
    elseif inj == 2
        gfzero = [1E-5 1E-4 1E-3 1E-2 1E-1 1E0 1E1];
    end
    ef = zeros(length(gfzero),1);   cou = 0;   cal = 0;
    for i=gfzero
        cou = cou + 1;
        ef(cou,1) = sign(...
            res_bratz_vf0(blk,esi,i,ed,uu,...
            h11_ll,h12_ll,h21_ll,h22_ll,h11_cp,h12_cp,h21_cp,h22_cp,...
            ic1,ic2,hy1,hy2,u1zero,u1menos1,u2zero,u2menos1,...
            du1,du2,u1r,u2r,vhll_11,vhcp_11,vhll_12,vhcp_12,...
            vhll_21,vhcp_21,vhll_22,vhcp_22,vtc,cal_nom,opmu));
        if cou > 1
            si = ef(cou,1) * ef(cou-1,1);
            if si < 0
                li = gfzero(1,cou-1);   ls = gfzero(1,cou);
                cal = 1;   fprintf('li␣=␣%1.7f␣,␣ls␣=␣%1.7f␣\n',li,ls)
                break
            end
        end
    end
    if cal == 1,
        break
    end
end
if cal == 0
    fprintf('Unfeasible␣problem␣under␣the␣current␣conditions␣\n')
end
```

```
function kcillo = fkssv(t_lower,t_upper,blk,esi,ed,uu,h11_ll,h12_ll,...
    h21_ll,h22_ll,h11_cp,h12_cp,h21_cp,h22_cp,ic1,ic2,hy1,hy2,...
    u1zero,u1menos1,u2zero,u2menos1,du1,du2,u1r,u2r,vhll_11,vhcp_11,...
    vhll_12,vhcp_12,vhll_21,vhcp_21,vhll_22,vhcp_22,vtc,cal_nom,opmu)
tit = clock;  tol = 1e-3;  kc = 0;  t_works = t_upper;
while abs((t_upper - t_lower))>tol
    kc = kc + 1;
    t_test = (t_upper+t_lower)/2;
    sol = da_signo(t_test,blk,esi,ed,uu,h11_ll,h12_ll,h21_ll,h22_ll,...
        h11_cp,h12_cp,h21_cp,h22_cp,ic1,ic2,hy1,hy2,u1zero,u1menos1,...
        u2zero,u2menos1,du1,du2,u1r,u2r,vhll_11,vhcp_11,vhll_12,vhcp_12,...
        vhll_21,vhcp_21,vhll_22,vhcp_22,vtc,cal_nom,opmu);
    if sol ~= 0
        t_lower = t_test;
    else
        t_upper = t_test;
        t_works = t_test;
    end
    if kc > 20
        fprintf('Iteration␣number␣reached␣kfssv␣\n')
        t_upper = t_lower;
    end
end
kcillo = t_works;
```

```
function s_i = da_signo(k_ssv,blk,esi,ed,uu,h11_ll,h12_ll,h21_ll,h22_ll,...
    h11_cp,h12_cp,h21_cp,h22_cp,ic1,ic2,hy1,hy2,u1zero,u1menos1,...
    u2zero,u2menos1,du1,du2,u1r,u2r,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,vtc,cal_nom,opmu)
ef = res_bratz_vf0(blk,esi,k_ssv,ed,uu,h11_ll,h12_ll,h21_ll,h22_ll,...
    h11_cp,h12_cp,h21_cp,h22_cp,ic1,ic2,hy1,hy2,u1zero,u1menos1,...
    u2zero,u2menos1,du1,du2,u1r,u2r,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,vtc,cal_nom,opmu);
if ef >= 0
    s_i = 0;
else
    s_i = 1;
end
```

```
function c = res_bratz_vf0(blk,esi,k,ed,kuig,h11_ll,h12_ll,h21_ll,h22_ll,...
    h11_cp,h12_cp,h21_cp,h22_cp,ic1,ic2,hy1,hy2,u1zero,u1menos1,...
    u2zero,u2menos1,du1,du2,u1r,u2r,vhll_11,vhcp_11,vhll_12,vhcp_12,...
    vhll_21,vhcp_21,vhll_22,vhcp_22,vtc,cal_nom,opmu)
ts = clock;  u = zeros(20,1);  m = length(kuig)/2;
switch m
    case 1
```

```
        u(01:10) = kuig(1);  u(11:20) = kuig(2);
    case 2
        u(01:02) = kuig(1:2);  u(03:10) = kuig(2);
        u(11:12) = kuig(3:4);  u(13:20) = kuig(4);
    case 3
        u(01:03) = kuig(1:3);  u(04:10) = kuig(3);
        u(11:13) = kuig(4:6);  u(14:20) = kuig(6);
    case 4
        u(01:04) = kuig(1:4);  u(05:10) = kuig(4);
        u(11:14) = kuig(5:8);  u(15:20) = kuig(8);
end
%This corresponds to the example when M=3
mb = dabmdentro(esi,k,ed,u,h11_ll,h12_ll,h21_ll,h22_ll,h11_cp,h12_cp,...
    h21_cp,h22_cp,ic1,ic2,hy1,hy2,u1zero,u1menos1,u2zero,u2menos1,...
    du1,du2,u1r,u2r,vhll_11,vhcp_11,vhll_12,vhcp_12,vhll_21,vhcp_21,...
    vhll_22,vhcp_22,vtc,cal_nom); %This corresponds to the example when M=3
bn = mussv(mb,blk,opmu);  c = k-bn(1,1);
```

Table I.7 show the programs that was used to generate the normalized input sequence and the programs that convert the normalized input sequence to absolute units for the empirical models, i.e, state-affine and Volterra series.

Table I.7: List of programs to generate the input sequence used for the identification process

Program Name	Description
c_sequence.m	Program to calculate the points for the normalized sequence
man_var_values	Program to convert the normalized input sequences to absolute units for the state-affine model
man_var_values_volterra_siso	Program to convert the normalized input sequence to absolute units for the SISO Volterra series model
man_var_values_volterra_mimo	Program to convert the normalized input sequences to absolute units for the MIMO Volterra series model

```
function data_seq_to_use = c_sequence(nran,repetir)

% nran      number of initial points
% repetir   number of sequence value repetitions in the final sequence

%Only 3 levels are considered

level_1 = -1;
level_2 =  0;
level_3 = +1;

probability_1 = 1/3;
probability_2 = 1/3;
```

```
probability_3 = 1/3;

seq_levels = [level_1 , level_2 , level_3];
pro_levels = [probability_1 , probability_2 , probability_3];

data_seq = randsrc(nran,2,[seq_levels ; pro_levels]);

data_seq(1,:) = 0;

data_seq_to_use = zeros(repetir*length(data_seq),2);

for i=1:length(data_seq)
    data_seq_to_use(repetir*(i-1)+1:repetir*i,1) = data_seq(i,1);
    data_seq_to_use(repetir*(i-1)+1:repetir*i,2) = data_seq(i,2);
end

figure
subplot(2,2,1) , plot(data_seq(:,1))
ylim([-1.1 1.1]) , title('MV_1')
subplot(2,2,2) , plot(data_seq(:,2))
ylim([-1.1 1.1]) , title('MV_2')
subplot(2,2,3) , plot(data_seq_to_use(:,1))
ylim([-1.1 1.1]) , title('MV_1␣with␣repetitions')
subplot(2,2,4) , plot(data_seq_to_use(:,2))
ylim([-1.1 1.1]) , title('MV_2␣with␣repetitions')
```

```
function mv = man_var_values_sa(s_data,uss,sin)

% This function takes:
% a) the normalized data (s_data),
% b) the value of the manipulated variables at the steady state for a
%    specific value of S_in (uss),
% c) the average value of s_in, which is used to calculate the 2 scaling
%    factors (pv1mv and pv2mv)
% This function returns the real value of the manipulated variables that
% will be used to identify the process

ro = size(s_data);
mv = zeros(ro,2); % preallocation

% The scaling factors were used as follows:
% Sin = 2.0, pv1mv = 0.1, pv2mv = 0.1
% Sin = 2.5, pv1mv = 0.04, pv2mv = 0.04
% Sin = 3.0 , pv1mv = 0.015, pv2mv = 0.015

if sin == 2.0
    pv1mv = 0.1;
    pv2mv = 0.1;
elseif sin == 2.5
    pv1mv = 0.04;
    pv2mv = 0.04;
elseif sin == 3.0
    pv1mv = 0.015;
    pv2mv = 0.015;
else
    error('Incorrect␣option')

mv(:,1) = ( uss(1,1) * pv1mv * s_data(:,1) ) + uss(1,1);
mv(:,2) = ( uss(2,1) * pv2mv * s_data(:,2) ) + uss(2,1);
```

```
function mv = man_var_values_volterra_siso(s_data,u_ss)
```

```
% This function takes:
% a) the normalized data (s_data),
% b) the value of the manipulated variable at the steady state (u_ss) for a
%    specific value of \Beta
% This function returns the real value of the manipulated variable that
% will be used to identify the process

ro = size(s_data);

pv1 = 9/14;

mv = 0.075*pv1*s_data(:,1) + u_ss(1,1);
```

```
function mv = man_var_values_volterra_mimo(s_data,uss)

% This function takes:
% a) the normalized data (s_data),
% b) the value of the manipulated variables at the steady state for a
%    specific value of q_2, Cv and n (uss),
% This function returns the real value of the manipulated variables that
% will be used to identify the process

ro = size(s_data);

mv = zeros(ro,2); % preallocation

pv1mv = 0.025;
pv2mv = 0.025;

mv(:,1) = ( uss(1,1) * pv1mv * s_data(:,1) ) + uss(1,1);
mv(:,2) = ( uss(2,1) * pv2mv * s_data(:,2) ) + uss(2,1);
```