

**Robust Control Design of Gain-scheduled Controllers
for Nonlinear Processes**

by

Jiaying Gao

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, 2004

©Jiaying Gao 2004

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the chemical or biochemical industry most processes are modeled by nonlinear equations. It is of a great significance to design high-performance nonlinear controllers for efficient control of these nonlinear processes to achieve closed-loop system's stability and high performance. However, there are many difficulties which hinder the design of such controllers due mainly to the process nonlinearity. In this work, comprehensive design procedures based on robust control have been proposed to efficiently deal with the design of gain-scheduled controllers for nonlinear systems.

Since all the design procedures proposed in this work rely strongly on the process model, the first difficulty addressed in this thesis is the identification of a relatively simple model of the nonlinear processes under study. The nonlinearity of the processes makes it often difficult to obtain a first-principles model which can be used for analysis and design of the controller. As a result, relatively simple empirical models, Volterra series model and state-affine model, are chosen in this work to represent the nonlinear process for the design of controllers.

The second major difficulty is that although the nonlinear models used in this thesis are easy to identify, the analysis of stability and performance for such models using nonlinear control theory is not straightforward. Instead, it is proposed in this study to investigate the stability and performance using a robust control approach. In this approach, the nonlinear model is approximated by a nominal linear model combined with a mathematical description of model error to be referred to, in this work, as model uncertainty. In the current work it was assumed that the main source of uncertainty with respect to the nominal linear model is due to the system nonlinearity. Then, in this study, robust control theoretical tools have been especially developed and applied for the design of gain-scheduled Proportional-Integral (PI) control and gain-scheduled Model Predictive Control (MPC).

Gain-scheduled controllers are chosen because for nonlinear processes operated over a wide range of operation, gain-scheduling has proven to be a successful control design technique (Bequette, 1997) for nonlinear processes. To guarantee the closed-loop system's robust stability and performance with the designed controllers, a systematic approach has been proposed for the design of robust gain-scheduled controllers for nonlinear processes. The design procedure is based on robust stability and performance conditions proposed in this work. For time-varying uncertain parameters, robust stability and performance conditions using fixed Lyapunov functions and parameter-dependent Lyapunov functions, were used. Then, comprehensive procedures for the design and optimization of robust gain-scheduled PI and MPC controllers tuning parameters based on the robust stability and performance tests are then proposed.

Since the closed-loop system represented by the combination of a state-affine process model and the gain-scheduled controller is found to have an affine dependence on the uncertain parameters, robust stability and performance conditions can be tested by a finite number of Linear Matrix Inequalities (LMIs). Thus, the final problems are numerically solvable.

One of the inherent problems with robust control is that the design is conservative. Two approaches have been proposed in this work to reduce the conservatism. The first one is based on parameter-dependent Lyapunov functions, and it is applied when the rate of change of the time-varying uncertainty parameters is *a priori* available. The second one is based on the relaxation of an input-saturation factor defined in the thesis to deal with the issue of actuator saturation.

Finally, to illustrate the techniques discussed in the thesis, robust gain-scheduled PI and MPC controllers are designed for a continuous stirred tank reactor (CSTR) process. A simple MIMO example with two inputs and two outputs controlled by a multivariable gain-scheduled MPC controller is also discussed to illustrate the applicability of the methods to multivariable situations. All the designed controllers are simulated and the

simulations show that the proposed design procedures are efficient in designing and comparing robust gain-scheduled controllers for nonlinear processes.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Hector M. Budman. During the past four years, Hector has been instrumental in ensuring my academic, professional and financial wellbeing. In every sense, none of this work would have been possible without him. It has been a great privilege to work with him.

Many thanks also go to my committee members, Professors Thomas Duever, Daniel Davison, Peter Douglas, Professor Ali Elkamel, and James McLellan. Thank you for your constructive and valuable comments, suggestions and contributions to this thesis.

Far too many people to mention individually have assisted and mentored me in so many ways during my study and work at the University of Waterloo. They all have my sincere gratitude. In particular, I would like to thank Professors Alexander Penlidis, Jenő M. Scharer, and Park Reily.

I would like to express my sincere appreciation to my friends, Xi Zhang, Weimin Guo, Mingsen Zhang, Yong Gu, and many other friends in Waterloo. With their friendship and support, the four years of work have been years of fun and pleasure. I am also indebted to all of my colleagues and officemates, and friends in the process control group, especially to Shijin Lou, Jesse Huebsch, Deborah, Jessada Jitjareonchai, Greer Painter, Clement Lee, Luigi D'Agnillo, for all the help and the interesting discussions in the lab.

My deepest thank-you goes to my wonderful parents, Aizhen Shen and Wanchun Gao, for always being there when I needed them most, and never once complaining about how infrequently I visit. They deserve far more credit than I can ever give them. No matter where I am, their love always accompanies me and encourages me. My final, and most heartfelt, acknowledgment goes to my husband Jian Ma. His deep love and patience have turned my journey through graduate school into a possibility, and a joy. For all that, and for everything I am given by them, they have my everlasting love.

Table of Contents

Abstract	iii
Acknowledgements	vi
Table of Contents	vii
List of Tables	x
List of Figures	xi
1 Introduction	1
1.1 Empirical Modeling	1
1.2 Robust Control	2
1.3 Objectives and Novelties	4
1.4 Outline of the Work	5
2 Literature Review	8
2.1 Empirical Modeling of Nonlinear Processes	8
2.1.1 Volterra series model.....	10
2.1.2 State-affine model.....	11
2.2 Robustness Analysis	12
2.2.1 Quadratic Lyapunov functions.....	13
2.2.2 Parameter-dependent Lyapunov functions.....	15
2.2.3 Linear matrix inequalities (LMIs) in control.....	15
2.2.4 Structured singular value (SSV) analysis.....	18
2.3 Gain-scheduled Controller Design	21
2.4 Robustness of MPC	24
3 Uncertain Dynamical Systems	29
3.1 Model Uncertainty	30
3.2 Uncertain State-space Models	32
3.2.1 Affine Parameter-dependent models.....	32
3.2.2 Quantification of model uncertainty.....	35
3.3 Linear-Fractional Models of Uncertainty	36
3.4 Model Identification Methodology	37
3.4.1 Volterra series models.....	37
3.4.2 State-affine models.....	39
3.5 Case Study	42
3.5.1 Nonlinear process: CSTR.....	42
3.5.2 Volterra series models of the CSTR.....	43
3.5.3 State-affine models of the CSTR.....	46
4 Robust Stability and Robust Performance Analysis ..	55
4.1 Linear Matrix Inequalities (LMIs)	56
4.1.1 LMIs and LMI problems.....	56
4.1.2 Well-posedness issues.....	59

4.1.3	Semi-definite $B(x)$ in GEVP problems	60
4.2	Quadratic Lyapunov Functions	61
4.2.1	Quadratic Lyapunov stability (QLS)	65
4.2.2	Quadratic Lyapunov H_∞ performance (QLP)	67
4.3	Parameter-dependent Lyapunov Functions	71
4.3.1	Time-varying uncertain parameters	74
4.3.2	Constant uncertain parameters	78
4.4.1	Review of the SSV concept	81
4.4.2	Generation of an $M - \Delta$ LFT	88
4.4.3	RS and RP conditions for time-varying uncertainty	90
5	Robust Gain-Scheduled PI Controller	97
5.1	Gain-scheduled PI Controller	98
5.1.1	Closed-loop system	99
5.1.2	Input-saturation	103
5.1.3	Modeling error	106
5.2	Design and Optimization using Quadratic Lyapunov Functions	108
5.2.1	Design of robust gain-scheduled PI controller	108
5.2.2	Optimization of robust gain-scheduled PI controllers	109
5.2.3	Relaxation of the input-saturation factor ψ	111
5.3	Design based on SSV Analysis	113
5.3.1	Generation of an $M - \Delta$ LFT: simple case	113
5.3.2	Generation of an $M - \Delta$ LFT: with modeling error	115
5.3.3	Design of robust gain-scheduled PI controllers: SSV analysis	118
5.4	CSTR Case Study	120
5.4.1	Design and optimization using quadratic Lyapunov functions	122
5.4.2	Reducing conservatism of the quadratic design and optimization	130
5.4.3	Design based on the SSV analysis	144
5.5	Conclusions	146
6	Robust Gain-scheduled MPC	149
6.1	Unconstrained MPC Control Law	151
6.1.1	Model prediction based on step response models	151
6.1.2	Unconstrained SISO MPC control law	155
6.1.3	Unconstrained MIMO MPC control law	159
6.1.4	Unconstrained MIMO MPC control law in state-space form	162
6.2	Design and Optimization using Quadratic Lyapunov Functions	168
6.2.1	Selection of MPC tuning parameters	168

6.2.2	Design and optimization of gain-scheduled MPC.....	169
6.3	<i>Case Study Results and Conclusions</i>	173
6.3.1	Design results for SISO processes.....	173
6.3.2	Design results for MIMO processes.....	180
6.3.3	Conclusions.....	189
7	Conclusions and Future Work	192
7.1	<i>Conclusions</i>	192
7.2	<i>Future work</i>	198
7.2.1	Reducing conservatism of the gain-scheduled MPC design.....	199
7.2.2	Reducing conservatism of the robustness analysis.....	204
	References	213
8	Appendix A: Nomenclature for CSTR	219
9	Appendix B: MATLAB Code	220
9.1	<i>Model Identification</i>	220
9.2	<i>Gain-scheduled PI Controllers Design</i>	225
9.2.1	Quadratic Lyapunov functions.....	225
9.2.2	Parameter-dependent Lyapunov functions.....	231
9.2.3	Structured Singular Value.....	235
9.3	<i>Gain-scheduled MPC Controllers Design</i>	239
9.3.1	SISO processes.....	239
9.3.2	MIMO processes.....	250
10	Appendix C: Nomenclature	265
11	Appendix D: Defense Presentation Slides	268

List of Tables

Table 3.1 Example of Behavior matrix $\mathbf{B}(f)$	40
Table 3.2 Volterra kernels ($M = 9$).....	45
Table 3.3 1 st -order Volterra kernels and corresponding Behavior matrix entries.....	47
Table 3.4 Volterra kernels and corresponding Behavior matrix entries	52
Table 4.1 Commuting $\mathbf{D} - \mathbf{\Lambda}$ pairs	86
Table 5.1 Optimization design results	128
Table 5.2 Input-saturation factor lower bound for controllers on stability limit	141
Table 5.3 Relaxation of input-saturation factor bound and conservatism reduction	142
Table 6.1 Gain-scheduled MPC controller optimization (SISO).....	175
Table 6.2 Gain-scheduled MPC controller simulation (SISO).....	176
Table 6.3 Compare: Gain-scheduled PI and Gain-scheduled MPC.....	179
Table 6.4 MPC controller analysis (comparing LMPC-M and GSMPC-M).....	182
Table 6.5 MPC controller simulation (comparing LMPC-M and GSMPC-M).....	183
Table 6.6 MPC controller optimization	187
Table 7.1 Simulation results of two MPC controllers.....	203
Table 8.1 Nomenclature for exothermic CSTR	219
Table 9.1 MATLAB files for model identification.....	220
Table 9.2 MATLAB files for Gain-scheduled PI design: fixed Lyapunov functions	225
Table 9.3 MATLAB files for Gain-scheduled PI design:parameter-dependent Lyapunov functions.....	231
Table 9.4 MATLAB files for Gain-scheduled PI design: SSV	235
Table 9.5 MATLAB files for Gain-scheduled MPC design: SISO	239
Table 9.6 MATLAB files for Gain-scheduled MPC design: MIMO.....	250

List of Figures

Figure 2.1 Complex disc covering real interval.....	20
Figure 3.1 General $\mathbf{M} - \Delta$ LFT framework	37
Figure 3.2 Open loop simulation	43
Figure 3.3 Input/output data.....	44
Figure 3.4 CSTR process output (solid line) and Volterra series model output (dotted line)	46
Figure 3.5 CSTR process output (solid line) and state-affine model output (dotted line)	53
Figure 4.1 General $\mathbf{M} - \Delta$ LFT framework	83
Figure 4.2 Equivalent $\mathbf{M} - \Delta$ framework (Equation (4.74))	89
Figure 4.3 Equivalent $\mathbf{M} - \Delta$ framework (Equation (4.77))	90
Figure 4.4 Equivalent $\mathbf{M} - \Delta$ framework (equation (4.82)).....	90
Figure 4.5 Equivalent scaled and transformed loops for robust stability	93
Figure 4.6 Equivalent scaled and transformed loops for robust performance	94
Figure 5.1 Open-loop properties of CSTR ($x_c = [-10 \quad 40]$).....	121
Figure 5.2 Stability and performance regions (to the left of the lines) of linear PI controller parameters.	123
Figure 5.3 Performance region (inside the lines) of gain-scheduled PI controller parameters.	125
Figure 5.4 Stability region (inside the lines) of gain-scheduled PI controller parameters.	126
Figure 5.5 Closed-loop simulations of state-affine model (lower two curves).....	129
Figure 5.6 Stability region (to the left of the lines) of linear PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).	132
Figure 5.7 Performance region (to the left of the lines) of linear PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).	134
Figure 5.8 Stability region (inside the lines) of gain-scheduled PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).	136
Figure 5.9 Performance region (inside the lines) of gain-scheduled PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).	137
Figure 5.10 Stability regions (to the left of the lines) of linear PI controller parameters (comparing: without input-saturation (solid) and with $\psi \in [0.4 \quad 1]$ (dotted)).	138
Figure 5.11 Performance regions (to the left of the lines) of linear PI controller parameters (comparing: without input-saturation (solid) and with $\psi \in [0.4 \quad 1]$ (dotted)).....	139
Figure 5.12 Stability regions (to the left of the lines) of linear PI controller parameters (comparing: without input-saturation (solid), with $\psi \in [0.4 \quad 1]$ (dotted) and with $\psi \in [0.6203 \quad 1]$ (dashed)).	140

Figure 5.13 Performance regions (to the left of the lines) of linear PI controller parameters (comparing: without input-saturation (solid), with $\psi \in [0.4 \ 1]$ (dotted) and with $\psi \in [0.6203 \ 1]$ (dashed)).....	141
Figure 5.14 Stability region (inside the lines) of gain-scheduled PI controller parameters (comparing: quadratic Lyapunov approach (solid) and SSV approach (dotted))... ..	144
Figure 5.15 Performance region (inside the lines) of gain-scheduled PI controller parameters (comparing: quadratic Lyapunov approach (solid) and SSV approach (dotted)).....	145
Figure 6.1 Step response of a process.....	151
Figure 6.2 Model predictive control problem.....	156
Figure 6.3 Vertices of the parameter box to be tested for robust stability.....	175
Figure 6.4 GSMPC5-1 (solid line) and GSMPC5-2 (dotted line) simulation	177
Figure 6.5 Disturbance signal used for the results in Table 6.2.....	178
Figure 6.6 GSPI (dotted line) and GSMPC3 (solid line) simulation	179
Figure 6.7 LMPC-M (solid line) and GSMPC-M (dotted line) simulation ($t = [1,100]$)	184
Figure 6.8 LMPC-M (solid line) and GSMPC-M (dotted line) simulation ($t = [1,500]$)	185
Figure 6.9 Disturbance signal used for the simulation	185
Figure 6.10 LMPC-M-OPT (solid line) and GSMPC-M-OPT (dotted line) simulation	187
Figure 6.11 LMPC-M-OPT (solid line) and GSMPC-M-OPT (dotted line) simulation	188
Figure 7.1 Step response ($u_1 \in [0,1], u_2 \in [-0.8,1]$).....	201
Figure 7.2 Step response ($u_1 \in [-1,0], u_2 \in [-1,-0.8]$).....	202
Figure 7.3 GS-MPC-1 (solid line) and GS-MPC-2 (dotted line) simulation.....	202
Figure 7.4 Disturbance signal used for the simulation results in Figure 7.3 and Table 7.1	203
Figure 7.5 Parameter box.....	211

1 Introduction

Chemical or biochemical processes are in general highly nonlinear especially when operated over a wide range of operating conditions. The nonlinearity is generally related to reaction kinetics or nonlinearity of physical properties. Therefore, there is a strong motivation to control these processes with nonlinear controllers. However, there are not many general design procedures to deal with this task, and there are many difficulties to design such controllers because of the systems nonlinearity.

For model-based control design problems for highly nonlinear processes, the first difficulty is to obtain a good simple model of the processes under study. Relatively simple empirical models can be identified from process input/output data. Different techniques such as Volterra series or nonlinear auto-regressive moving average models (NARMA) have been used to identify reduced-order empirical models of the process.

The second major difficulty is that although the nonlinear models used in this thesis are easy to identify, the analysis of stability and performance for such models using nonlinear control theory is not straightforward. Since the state-affine models used in this work can be easily approximated by a nominal linear part and model uncertainty, robust control theory is a natural choice to analyze this type of models. This research deals with the application of robust control theory for the design of control techniques such as gain-scheduling control, Proportional-Integral (PI) control, and Model Predictive Control (MPC). Methods for quantifying the model uncertainty from experimental data are shown. Then, the corresponding controllers are designed to provide robust stability and performance in the presence of model/plant mismatch.

1.1 Empirical Modeling

In general, the design of high performance controllers requires accurate mathematical modeling of the nonlinear processes to be regulated. Two types of nonlinear models may

be considered: 1-first principle models, i.e., models based on mass and energy balances; and 2-empirical models.

In many cases it is difficult to find proper first principle models due to, for example, the fact that kinetic properties are very difficult to identify or may change as a function of the operating conditions and thus, it is difficult to come up with the correct model. Even in cases where the kinetic properties are known accurately, the development of first principle models may be impractical for model based control if the model requires a large number of differential equations with a significant number of dynamic states. These models may also include a significant number of parameters that may be very costly to identify.

For the above reasons, an attractive alternative is to use relatively simple and compact empirical models obtained directly from measured input/output data. Examples of nonlinear empirical models are NARMA models, Volterra series models and state-affine models. This work uses an algorithm that produces nonlinear state-affine models from process input/output data, through an intermediate-step identification of the Volterra series models.

1.2 Robust Control

This study proposes the design of a robust controller based on a nonlinear state-affine model of a nonlinear process. The main subject of this work deals with the analysis and design of the nonlinear closed-loop control system. In the design, given a nonlinear plant to be controlled and some closed-loop specifications, the task is to construct a controller such that the closed-loop system meets the desired characteristics.

Linear control is a mature subject with a wide variety of powerful design methodologies and a long history of successful industrial applications. However, there is an active interest in the development and applications of nonlinear control methodologies. Many reasons can be cited for this:

1. Improvement of existing control systems: Linear control methods rely on the key assumption of small range operation for the linear model to be valid. When the required operation range is large, a linear controller is likely to perform very poorly or be unstable, because the nonlinearities in the system can not be properly compensated for. Nonlinear controllers, on the other hand, may handle the nonlinearities in a larger range of operation.

2. Analysis of hard nonlinearities: Another assumption of linear control is that the system model is indeed linearizable. However, in control systems, there are many nonlinearities whose discontinuous nature does not allow linear approximation. These so-called “hard nonlinearities” (Slotine and Li, 1991) such as saturation and dead-zones, are often found in control engineering. Their effects cannot be derived from linear methods, and nonlinear analysis techniques must be developed to predict a system’s performance in the presence of these inherent nonlinearities.

3. Dealing with model uncertainties: In designing linear controllers it is usually necessary to assume that the parameters of the system model are reasonably well known. However, many control problems involve uncertainties in the model parameters. This may be due to slow time-variation of the parameters or parameter dependence on conditions. A linear controller based on inaccurate values of the model parameters may exhibit significant performance degradation or even instability. Nonlinearities can be intentionally introduced into the controller so that model uncertainties can be tolerated. Two classes of nonlinear controllers for this purpose are robust controllers and adaptive controllers.

Robust controllers are the focus of the current study. As a result, a comprehensive methodology to design robust gain-scheduled PI controllers and robust gain-scheduled MPC controllers, is presented here in this work. Our methodology comprises an identification step from plant data and the finding of the optimal model parameters. The objective is to propose a methodology that will be easy and fast to apply in industrial applications. Conditions which guarantee robust stability and performance are formulated

as a finite set of Linear Matrix Inequalities (LMIs) and hence, the resulting problem is numerically tractable.

1.3 Objectives and Novelties

In summary, the objective of this work is to propose a comprehensive design procedure for gain-scheduled controllers, which can guarantee the robust stability and performance of the closed-loop systems. The fundamental basis of this work is the state-affine model previously developed and used in the work of Budman and Knapp (2000, 2001). By representing a nonlinear process with this model, it is possible to quantify the system uncertainty from the process nonlinearity, which is a function of the current input variable only.

The traditional gain-scheduling design technique is based on local linearization of the nonlinear processes, and it has proven to be a successful design methodology in many engineering applications (Bequette, 1997). However, in the absence of a sound theoretical analysis, these designs come with no guarantees of robust stability, performance or even nominal stability of the overall gain-scheduled design (Shamma and Athans, 1990). This work presents such an analysis for one type of nonlinear gain-scheduled control system. This gain-scheduled control system is novel in that it is based on the process input, different from the gain-scheduled designs in the literature based on the process output or system reference trajectory (Shamma and Athans, 1990). The gain-scheduling PI controller parameters are changing as a continuous function of the scheduling variable, i.e., the process input, instead of being switched at discrete values.

A methodology is proposed for the design and optimization of the robust gain-scheduled controllers. Conditions which guarantee robust performance are proposed in this work as extensions of the previous work on robust stability by Budman and Knapp (2001) formulated as a finite set of Linear Matrix Inequalities (LMIs). The work by Budman and Knapp (2001) was only applied to the design of traditional gain-scheduled PI controllers, which satisfy closed-loop robust stability. In this work, both the robust stability and the

robust performance conditions are applied for the design of the novel continuous gain-scheduled PI controllers. The resulting problem formulated as a finite set of LMIs is numerically tractable. Issues of modeling error and input-saturation are explicitly incorporated into the analysis.

Additionally and in contrast with the nonlinear MPC controllers used in the literature (Chen and Allgower (1998)), gain-scheduled MPC controllers, based on the discretization of the operation range, are proposed in this work. The design of the gain-scheduled MPC controllers consists of optimizing the controller parameters, specifically the input weights, and of scheduling the step response matrix of the MPC controller, based on the input variables.

The inherent conservatism of robustness analysis results in smaller ranges of controller parameters that satisfy the design criteria and consequently in degraded performance. Two approaches, the use of parameter-dependent Lyapunov function and relaxation of the input saturation factor bounds, are proposed to reduce the conservatism of the controller design. The parameter-dependent Lyapunov function has been proposed in the literature (Gahinet, Apkarian and Chilali in 1994), and applied for continuous systems. In this work, it is extended to the robustness analysis of discrete-time systems. The relaxation of the input-saturation factor is proposed in this work, and to our knowledge it has not been reported in the literature.

Finally, several alternatives to reduce the conservatism of the analysis, are proposed at the end of the work as subjects for future study.

1.4 Outline of the Work

The thesis is organized in chapters as follows:

Chapter 1. provides an overview of the work.

Chapter 2. includes a complete literature review, which covers the available techniques to approach the problem of empirical modeling and robust stability and performance analysis. Robustness issues with respect to gain-scheduled PI and MPC controllers are also specifically reviewed.

Chapter 3. reviews identification methods of nonlinear processes using state-affine models from its input/output data. The representation method of the state-affine model as affine parameter-dependent systems is illustrated, and the approach for quantifying the model uncertainty is also given.

Chapter 4. presents the most fundamental analysis tools used in this work, including the approach based on Lyapunov function and the approach using structured singular value (SSV) analysis. For the first approach, the concept of quadratic Lyapunov stability and its extension to stability and performance analysis of nonlinear systems are reviewed. Several techniques, based on parameter-dependent Lyapunov functions, are proposed in this work for reducing the conservatism of the quadratic Lyapunov stability and performance tests. For the second approach, the original SSV theory in frequency-domain is first given, and then the development of the SSV extensions to deal with nonlinear and time-varying uncertainty is summarized. Finally, the two approaches are compared.

Chapter 5. proposes the gain-scheduled PI controller design methodology. First, a gain-scheduled PI controller structure is proposed which schedules the tuning parameters as a continuous function of the manipulated variable. The systematic robust design approach is then proposed to generate regions of the gain-scheduled PI controller parameters in the parameter space, which guarantee the closed-loop systems' robust stability and performance. The design is based on the satisfaction of the robust stability and performance conditions. The robustness conditions include the quadratic Lyapunov stability and performance conditions, the less conservative parameter-dependent Lyapunov function proposed in this thesis, and the SSV extensions. An analytical approach calculating the input-saturation factor bounds is also first developed in this work and applied to reduce the conservatism of the design.

Chapter 6. proposes the gain-scheduled MPC controller design methodology. A systematic robust design approach is proposed to compute proper input weights in the objective function of the MPC controllers, which guarantee the closed-loop systems' robust stability and performance. The design is based on the global satisfaction of the robust stability and performance conditions, and in this chapter, only the conditions based on the quadratic Lyapunov functions are used. This approach is different from the gain-scheduled designs proposed before in the literature. Modifications have been developed in this work to improve the traditional linear MPC formulation, which are very important to the design approach proposed here. First, explicit incorporation of plant uncertainty into the optimization objective function is realized by using the state-affine model for process output approximation. Second, to calculate the model based control actions, the step-response matrix is modified such that it changes according to the manipulated variable, to compensate for the system nonlinearity. Finally, extensions have been made for the design of MPC controllers for multi-input-multi-output (MIMO) processes.

Chapters 1.-5. concentrate on single-input-single-output (SISO) nonlinear systems represented in discrete-time form. Both SISO and MIMO nonlinear systems are considered in Chapter 6. The SISO case study example selected for this work is a typical chemical engineering process, a continuously stirred tank reactor (CSTR). Results on the CSTR process are presented in the different chapters. The theorems and approaches developed in this work can also be applied to other types of nonlinear chemical systems as explained in the thesis. The CSTR was selected as a case study example due, on the one hand, to the simplicity of its mathematical representation with only two dynamical states, i.e. reactor temperature and reactor concentration, and on the other hand, to its inherent nonlinearity. For the MIMO case study in Chapter 6, a simple 2x2 system is selected for simplicity, which is in the form of a state-affine model. If a real process is selected, it will be represented by this 2x2 state-affine model in the end, for the application of the approached proposed in this work.

2 Literature Review

The emphasis of this PhD work has been in developing accurate models of nonlinear processes and the development of stable and robust controllers for these processes. One of the main areas of interest in this field is robust control using nonlinear empirical models. This chapter will discuss some of the important work that has been performed in this field in recent years. Section 2.1 will discuss work that has been conducted in the area of nonlinear modeling based on input/output data. In section 2.2, major contributions to the analysis of robust stability and performance will be reviewed. Section 2.3 will focus on the gain-scheduling design approach. In section 2.4, the work on robustness of MPC controllers for nonlinear and uncertain processes is reviewed.

2.1 *Empirical Modeling of Nonlinear Processes*

Two key problems arise during the design stage of a robust controller for a nonlinear process:

1. Accurate mechanistic models are often difficult to obtain especially since many of the parameters are poorly known;
2. Even when mechanistic models are available, it is not trivial to quantify from them uncertainty bounds for robustness analysis purposes.

For example, Doyle et al. (1989, 1990) using a structured singular value approach, designed a robust linear controller for an exothermic CSTR. The method requires that:

- A mechanistic first-principles model of the process is available;
- An optimization procedure is carried out to find bounds on the perturbations representing nonlinearities of the model.

Unfortunately, in many situations, a mechanistic model of a nonlinear chemical process is not readily available from first principles. For instance, for biological reactors, the reaction kinetics are often unknown or very difficult to measure. Additionally, the optimization procedure proposed by Doyle to calculate uncertainty bounds was not trivial and may become very difficult when the model contains a large number of states.

A viable alternative to mechanistic modeling is to develop nonlinear empirical models directly from experimental input/output data. Using persistent-excitation signals based on the rules developed by Nowak and Van Veen (1994), limited experimental effort is required to identify nonlinear empirical models. One disadvantage with the use of empirical models is that they may have a structure, which is not totally correct to describe the actual nonlinear process, making it difficult to extrapolate the model predictions for operating conditions beyond the experimental data used for model training. Despite this, nonlinear empirical models still find a wide application in the field of nonlinear model-based control.

Based on the above considerations, in the current work, nonlinear empirical models were used. Examples of nonlinear empirical models are nonlinear auto-regressive moving average models (NARMA) (Haber, 1990; Hernandez, 1993), Volterra series models (e.g., Nowak, 1994) and state-affine models (Dang Van Mien, 1984; Diaz, 1988). However, NARMA and Volterra series models are not directly suitable for robust stability and performance analysis due to the dependence of the output on past inputs and outputs raised to different powers and in different product combinations. If all these products and high-order terms will be accounted for as model uncertainty in a robustness analysis, a very conservative design may result. On the other hand, it was found in a previous work by Knapp and Budman (2000) that nonlinear state-affine models initially proposed by Sontag (1978) and represented by equation (2.1) are ideally suited for the robustness analysis. The form of the model is as follows:

$$\begin{aligned}\mathbf{x}(t+1) &= \mathbf{F}(u(t))\mathbf{x}(t) + \mathbf{G}(u(t)) \\ y(t) &= \mathbf{H}(u(t))\mathbf{x}(t)\end{aligned}\tag{2.1}$$

where \mathbf{x} is the state vector and $\mathbf{F}(\cdot), \mathbf{G}(\cdot), \mathbf{H}(\cdot)$ are polynomial matrices, for example, $\mathbf{F}(u(t)) = \mathbf{F}_0 + \mathbf{F}_1 u + \mathbf{F}_2 u^2 + \dots$. These models have the distinct advantage that the nonlinear terms, which are assumed to be the source of model mismatch with respect to a nominal linear model, have a clear structure and are a polynomial function of the current inputs $u(t)$ only. This fact greatly facilitates the calculation of the uncertainty bounds since the inputs have *a priori* known limits due to, e.g., actuator limits or economic constraint considerations. Then, for the purpose of robustness analysis, a minimal state-affine realization in the form of equation (2.1) may produce less conservative results, regarding stability and performance compared to other nonlinear modeling techniques. It has been shown that nonlinear state-affine models can be synthesized from a Volterra series (Sontag, 1978). This will be further reviewed in the current work.

Empirical modeling of nonlinear processes has been a topic of much research for many years and several types of models have been reported. Only the nonlinear models relevant to this thesis will be discussed in the sequel, i.e., Volterra series models and state-affine models.

2.1.1 Volterra series model

A Volterra series model relates the output of a process to a polynomial of past inputs. Volterra theory is a generalization of the linear convolution integral approach often applied to linear, time-invariant systems. The theory states that any time-invariant, nonlinear system can be modeled as an infinite sum of multidimensional convolution integrals of increasing order. This method is the generalization of an impulse response for linear processes.

Sandberg (1992) showed that for a large class of systems, a truncated Volterra series provides a uniform approximation to the infinite Volterra series on a hyper-volume of bounded inputs. The Volterra series model is attractive because it is a straightforward generalization of the linear system description. Specifically, the parameters of the model

are linearly related to the output, hence the identification of the parameters is a linear regression problem that can be solved by standard least squares regression.

A complete review of Volterra series models can be found in the textbook of Schetzen (1989). In the present work, the identification of Volterra series model was conducted as an intermediate step towards the identification of a state-affine model. This identification procedure is explained in Chapter 3.

Nowak and Van Veen (1994) identified an input signal that provides persistent excitation (PE) for nonlinear Volterra series approximation using a least squares method. They showed that deterministic pseudo-random multilevel sequences (PRMS) are persistently exciting for a truncated Volterra series of polynomial order N only if the sequences take on $(N+1)$ or more distinct levels. In the current work, this input signal is used to generate input/output data for Volterra series model identification. Marmarelis (1978) gave definitions and properties of PRMS for reference.

2.1.2 State-affine model

Sontag (1978) studied a general type of input/output nonlinear relation known as a response map, which specified how past values of the input affect the present output of the system. Using the response map description, Sontag developed a very general realization theory for a class of nonlinear systems called state-affine system, i.e. systems that are affine in the state variables but are nonlinear with respect to the inputs. This system is represented by equation (2.1). The theoretical proofs and realization algorithms, in Sontag's work, offered a basis for the subsequent research work on discrete-time state-affine model realization. The idea behind Sontag's work is to find a minimal state-affine realization departing from a Volterra series model, which can be identified from input/output data. Sontag's algorithm is provided in Chapter 3 of this work for reference.

Based on Sontag's algorithm, for nonlinear processes, Knapp and Budman (1999, 2000, 2001) used an empirical state-affine model extracted from a Volterra series model for

robustness analysis of a nonlinear system under linear PI control. The state-affine model is of the form:

$$\begin{aligned} \mathbf{x}(t+1) &= \{\mathbf{F}_0 + \sum_{i=1}^{n-1} \mathbf{F}_i u(t)^i\} \mathbf{x}(t) + \{\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} u(t)^i\} u(t) \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t) \end{aligned} \quad (2.2)$$

where $\mathbf{F}_i, \mathbf{G}_i, \mathbf{H}_i$ are matrices of model coefficients, $\mathbf{x}(t)$ are the process states, $y(t)$ is the output, $u(t)$ is the manipulated variable. The model given in equation (2.2) can be easily identified from input/output data as explained by Budman and Knapp (2000, 2001).

The nonlinear terms with respect to the nominal linear model, assumed to be the main source of the uncertainty, are directly related to powers of the input. Since in practice the inputs are bounded by known limits, it is easy to quantify the uncertainty bounds. Thus, the optimization procedures such as the one proposed by Doyle (1989, 1990) to calculate the uncertainty bounds can be avoided, facilitating the application of the technique to systems with a large number of states. Sontag's algorithm (1978) calculated the model in equation (2.2) from a Volterra series model directly identified from input/output data. Thus, a first-principles model is not necessary.

2.2 Robustness Analysis

There are many options to consider when choosing a control strategy for a process, but, regardless of which control strategy is implemented, the controller will generally be designed based on a simplified model of the process. These models generally have varying degrees of accuracy, which do not take into account all model behavior. Controllers designed based on these models are desired to be robust in the presence of model uncertainty or model inaccuracy. A-posteriori robustness analysis is then necessary to validate the design and obtain guarantees of stability and performance in the face of plant uncertainty. In the literature, a variety of tools are available to assess robust stability and performance.

This section reviews most of the available Lyapunov-based analysis techniques: quadratic stability and performance analysis, and tests involving parameter-dependent Lyapunov functions, and structured singular value (SSV) analysis. Since all of these tests are based on sufficient conditions, they are only useful when they succeed in establishing finite and feasible robust stability and performance bounds.

2.2.1 Quadratic Lyapunov functions

The most useful and general approach for studying the stability of nonlinear control systems is the theory introduced in the late 19th century by the Russian mathematician Alexandr Mikkailovich Lyapunov. Lyapunov's work, *The General Problem of Motion Stability*, first published in 1982, includes two methods for stability analysis, the linearization method and the direct method. The linearization method draws conclusions about a nonlinear system's local stability around an equilibrium point from the stability properties of its linear approximation. The second method, referred to as the direct method, is not restricted to infinitesimal localized motion, and determines the stability properties of a nonlinear system by constructing a scalar "energy-like" function for the system and examining the function's time variations. The details of these two methods are summarized in many books, e.g., Slotine and Li (1991). Today, Lyapunov's linearization method is the basic theoretical analysis method for linear control. The Lyapunov's direct method has become the most important tool for nonlinear system analysis and design. Together, the linearization and the direct method constitute the so-called Lyapunov stability theory. The objective of this section is to review the application of Lyapunov stability theory in the analysis and design of nonlinear control systems.

Lyapunov's direct method and its extensions to performance analysis are applied in this work to uncertain time-varying systems. In the literature, linear matrix inequalities (LMIs) based tests have been derived to assess closed-loop robust stability and robust performance. LMIs problems are convex and efficient polynomial-time optimization algorithms are available to solve them, e.g., MathWorks MATLAB. The stability and

performance tests can be formulated as a finite set of LMIs and hence, the resulting problem is numerically tractable.

Gahinet and Apkarian (1994) have solved continuous and discrete-time H_∞ control problems via elementary manipulations on LMIs. A LMIs-based parameterization of all suboptimal H_∞ controllers has been given, including reduced-order controllers. Gahinet's work has also been based on quadratic Lyapunov functions for stability and performance analysis.

Budman and Knapp (2001) proposed the use of empirical state-affine model to design robust controllers. A novel methodology was proposed for the analysis of robust stability of a nonlinear process under Proportional-Integral (PI) control. This methodology has the advantage that it is based solely on empirical models. The state-affine model is nonlinear with respect to the manipulated variables. This model in combination with a linear PI controller results in a closed-loop model that can be shown to lie in a polytope of matrices. This allows for the formulation of a Lyapunov stability test in terms of a finite set of LMIs. The stability analysis has been used in their work to produce regions of stability in the PI controller parameters space. This technique has also been applied to test the stability of the closed-loop system with a simple traditional gain-scheduled PI controller. The analysis has been based only on robust stability while no robust performance has been considered in that work.

Gao and Budman (2003) have extended the robust stability analysis (Budman and Knapp, 2001) by considering robust performance for the design of a novel class of gain-scheduled PI controllers. The tuning coefficients of the controller used by Gao and Budman are continuous linear functions of the manipulated variable and these linear functions are defined in terms of four parameters only, whereas in the work of Budman and Knapp (2001) the switching of the controller parameters was effected at finite discrete values. A PI controller structure was selected because it is widely accepted in chemical process control practice. Subsequently, Gao's work addresses the optimization of these parameters. The parameterization of the proposed controller in terms of a

relatively small number of parameters greatly facilitates the optimization step. These new results are the core of the present thesis and will be shown in later chapters of this thesis.

2.2.2 Parameter-dependent Lyapunov functions

Quadratic Lyapunov stability and H_∞ performance tests guarantee stability and performance in the presence of uncertain parameters without considering the parameter rate of change. As a result, compared to the case when this information is taken into account in the design, these tests can be very conservative for time-varying parameters, thus affecting the efficiency of the design (Gahinet, Apkarian and Chilali, 1994; Gao and Budman, 2003).

To reduce conservatism in such cases, the notion of parameter-dependent Lyapunov functions was proposed by Gahinet, Apkarian and Chilali (1994). That is, for Lyapunov functions $V(t) = \boldsymbol{\eta}(t)^T \mathbf{P}(\boldsymbol{\delta}_t) \boldsymbol{\eta}(t)$, the Lyapunov matrix $\mathbf{P}(\boldsymbol{\delta}_t)$ is no longer constant, but it is now a function of $\boldsymbol{\delta}_t$. In their work, it was shown that by imposing additional constraints on the parameter-dependent Lyapunov functions, the calculation of a Lyapunov matrix of the form:

$$\mathbf{P}(\boldsymbol{\delta}_t) = \mathbf{P}_0 + \mathbf{P}_1 \delta_{1,t} + \mathbf{P}_2 \delta_{2,t} + \cdots + \mathbf{P}_n \delta_{n,t} \quad (2.3)$$

can be formulated into a LMIs problem for the unknown matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$. The resulting test is therefore numerically tractable while always less conservative than quadratic tests based on fixed Lyapunov matrices, i.e. $\mathbf{P} = \mathbf{P}_0$ in equation (2.3) because there are more parameters available for optimization.

2.2.3 Linear matrix inequalities (LMIs) in control

LMIs based techniques have emerged as powerful design tools in control engineering. Boyd and et al. (1994) have given a good introduction to LMIs concepts. It has been

shown that a wide variety of problems arising in system and control theory can be reduced to a convex optimization problem involving LMIs. Since these resulting optimization problems can be solved numerically very efficiently using recently developed interior-point methods (Boyd and et al., 1994), the resulting LMIs formulation is an attractive form of solution to complex problems. In comparison, the more conventional approach is to seek an analytic or frequency-domain solution to the matrix inequalities. In summary, three factors make LMIs techniques appealing:

1. A variety of design specifications and constraints can be expressed as LMIs.
2. Once formulated in LMIs, a problem can be solved using efficient numerical convex optimization algorithms available in MATLAB.
3. The main strength of LMIs formulations is the ability to combine various design constraints or objectives, with no analytical solutions in terms of matrix equations, in a numerically tractable manner (Wang and Balakrishnan, 1999; Budman and Knapp, 2001).

Many control problems and design specifications can be formulated as LMIs conditions, especially for Lyapunov-based analysis and design (Apkarian, 1995; Watanabe, 1996; Sivrioglu and Nonami, 1996). Packard et al. (1991) have given a collection of robust control problems that may be formulated in terms of LMIs. This is also true for optimal LQG control, H_∞ control, etc. Further applications of LMIs arise in estimations, identification, optimal design, matrix scaling problems, and so on.

To show the principles underlying the LMIs based design, the following two LMIs formulations of typical design objectives are shown here, while they are further detailed in later chapters in this work.

Stability: the stability of the dynamical system

$$\boldsymbol{\eta}(t+1) = \mathbf{A}\boldsymbol{\eta}(t), \boldsymbol{\eta}(0) = \boldsymbol{\eta}_0 \quad (2.4)$$

is given based on Lyapunov by the following problem:

$$\text{Find } \mathbf{P} > \mathbf{0}, \mathbf{P} = \mathbf{P}^T \text{ such that } \mathbf{A}^T \mathbf{P} \mathbf{A} - \mathbf{P} < \mathbf{0} \quad (2.5)$$

This can be generalized to the case where \mathbf{A} is assumed to vary within a polytope of matrices. Specifically,

$$\mathbf{A} = q_1 \mathbf{A}_1 + \dots + q_K \mathbf{A}_K, q_i > 0 \quad \sum_i q_i = 1 \quad (2.6)$$

where $\mathbf{A}_1, \dots, \mathbf{A}_K$ are fixed. Here the q_i 's are the coefficients of a convex decomposition of \mathbf{A} over the set $\{\mathbf{A}_1, \dots, \mathbf{A}_K\}$ of vertices of the polytope. A sufficient condition for the asymptotic stability of this system is the feasibility of a set of LMIs as follows:

$$\text{Find } \mathbf{P} > \mathbf{0}, \mathbf{P} = \mathbf{P}^T \text{ such that } \mathbf{A}_i^T \mathbf{P} \mathbf{A}_i - \mathbf{P} < \mathbf{0}, i = 1, \dots, K \quad (2.7)$$

RMS gain: for a stable system as follows:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{e}(t) \end{bmatrix} &= \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{v}(t) \end{bmatrix} \\ \boldsymbol{\eta}(0) &= \boldsymbol{\eta}_0 \end{aligned} \quad (2.8)$$

The random-mean-squares (RMS) gain is the largest input/output gain γ , $\|\mathbf{e}\|_{L_2} < \gamma \|\mathbf{v}\|_{L_2}$, over all bounded inputs. This gain is the global minimum of the following linear objective minimization problem:

$$\begin{aligned}
& \min_{\mathbf{P} > \mathbf{0}, \mathbf{P} = \mathbf{P}^T} \gamma^2 & (2.9) \\
& \text{subject to } \begin{bmatrix} \mathbf{A}^T \mathbf{P} \mathbf{A} - \mathbf{P} & \mathbf{A}^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A} & \mathbf{B}^T \mathbf{P} \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0}
\end{aligned}$$

Efficient interior-point methods (Boyd and et al., 1994) are available to solve the LMIs problems. In this work, the MATLAB LMIs Toolbox is used for the formulation and for the solution of the LMIs.

2.2.4 Structured singular value (SSV) analysis

A second technique to analyze the robustness of a closed-loop system is based on the structured singular value, referred to as μ , of a matrix. The basic concepts and results are summarized and reviewed by Doyle and Packard (1988, 1989).

For the case of linear model/plant mismatch, i.e. linear model with linear time-invariant perturbation, the model uncertainty has been handled most efficiently by the SSV approach (Packard and Doyle, 1987). The authors based the robust stability test on the closed-loop system's state-space representation, which can be decomposed into a linear-fractional transformation (LFT) structure made of a nominal constant part and an uncertain part representing changes in operating conditions. Necessary and sufficient conditions are obtained which guarantee stability and performance levels for the perturbed system, based on the bounds calculated from the above uncertain part. However, the stability and performance conditions with structured uncertainty reduce to computing μ for constant matrices $G(j\omega)$ and then taking *sup* over $j\omega$.

Suppose an uncertainty structure has only full blocks, and the perturbations are modeled as linear and time-invariant. It is shown by Doyle and Packard (1987) that the frequency domain μ test can conceptually be reduced to a single constant matrix μ test, but the actual uncertainty structure must be augmented with a repeated scalars block of size

equal to the state dimension. The key idea there is that the Laplace transform variable s , when appropriately transformed to the unit disk $|z| \leq 1$, can itself be interpreted as an additional block of repeated scalars in an augmented structure, replacing the search over the $j\omega$ term in the frequency domain. In view of the counter example given by Packard and Doyle (1988), it is likely that in this case of the augmented uncertainty structure, the upper bound of μ will not equal μ , and the conclusions will be conservative. Instead, for the original full block structure uncertainty, a frequency domain upper bound test is appropriate, since it has been found that for the frequency domain test, μ and the upper bound are very close. However, it is important to realize that the frequency domain test only gives conclusions about linear time-invariant perturbations. It is also applicable to time-invariant parameter-dependent systems by first deriving an equivalent LFT representation.

If the uncertainties are nonlinear and/or time-varying, then in general, the frequency domain tests are not valid. The upper bound approaches based on constant matrix operations proposed by Packard and Doyle (1988) handle this type of uncertainty, and the motivation which led to their development was the relationship between μ and the upper bound. These new advances in SSV theory allowed the application of the results to a class of time-varying and nonlinear models. Doyle, Packard and Morari (1989) have applied this technique to the calculation of margins of robust stability and robust performance for a nonlinear CSTR model, which is represented as a dynamical system with cone-bounded nonlinearities.

Doyle, Packard and Morari (1989) identified these bounds and designed a robust linear controller for the CSTR using extensions of SSV results to handle a class of time-varying and nonlinear systems. In Doyle's work, a first-principles model of the process is developed, from which conic bounds on the nonlinearities are found. Assuming that the model uncertainty is entirely due to the nonlinearities of the process, it was possible to use these conic bounds to describe the process with a linear nominal model augmented by a suitable uncertainty structure. Sufficient robust stability and robust performance

conditions for time-varying complex uncertainty are given, while the authors also mentioned that less conservative results are possible by considering real variations in the uncertainty, as the uncertainty for the CSTR model is more accurately described by real perturbations. Suppose that a real parameter k is assumed to be constant but uncertain, and the value of k is modeled to lie in an interval with a real uncertainty δ , shown by the dark line inside the disc in Figure 2.1, as follows:

$$k \in [0.8, 1.6] \Rightarrow k \in \{1.2 + 0.4\delta : \delta \in \mathbf{R}, |\delta| \leq 1\} \quad (2.10)$$

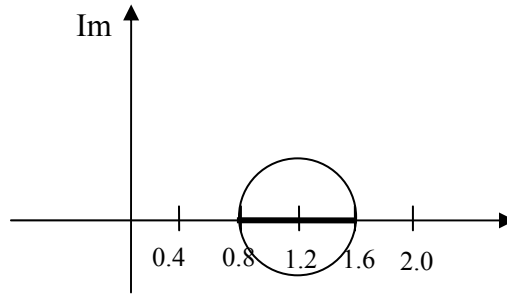


Figure 2.1 Complex disc covering real interval

However, the value of k is modeled to be the disc in Figure 2.1 with a complex uncertainty δ as follows:

$$k \in [0.8, 1.6] \Rightarrow k \in \{1.2 + 0.4\delta : \delta \in \mathbf{C}, |\delta| \leq 1\} \quad (2.11)$$

In general, using discs instead of intervals to model real uncertain parameters leads to more conservative robustness properties.

The identification of the bounds shown by Doyle (1989) is not trivial and requires careful observation of the nonlinearities to be bounded and the solution of an optimization problem to calculate the conic sectors. Additionally, Doyle's analysis (1989, 1990) was based on a mechanistic model, which for many processes is often not available.

To deal with these issues, Knapp and Budman (2000) used state-affine models identified from input/output data to design robust linear controllers for the nonlinear CSTR based on SSV robust stability analysis. A key advantage in using this state-affine model is that the model uncertainty, related to the system nonlinearity, is a function of the inputs only. This greatly facilitates the quantification of model uncertainty since bounds on the inputs are generally known, e.g., due to saturation limits of process actuators. One difficulty in using SSV analysis is that it is currently not clear how to integrate hard constraints on actuators into the test. This can be accomplished by using a LMIs formulation based on Lyapunov stability theory as shown by Knapp and Budman (2001).

Another problem is that the function $\mu_{\Delta}(\mathbf{M})$ is not necessarily a continuous function when all of the perturbation blocks are real. This mathematical fact is pointed out by Barmish and et al. (1990), and an example is given where the robustness margin to real parameter uncertainty changes abruptly for infinitesimal changes in the problem data. Also, in the Barmish and et al. (1990) example, the structured singular value of the frequency response exhibits discontinuities across frequency. What is the significance of these issues on $\mu_{\Delta}(\mathbf{M})$? The discontinuities can cause problems in the convergence of the lower bound algorithm. For problems with purely real uncertainty, the lower bound algorithm may converge to a value which is significantly lower than $\mu_{\Delta}(\mathbf{M})$ itself, or may not even converge at all. This could be a serious problem, but usually it is not, because almost all problems have a full complex block associated with a robust performance specification. It turns out that if a $\mu_{\Delta}(\mathbf{M})$ problem has at least one complex block that counts, then the function $\mu_{\Delta}(\mathbf{M})$ will be continuous at the problem data. Sometimes, though, a robust stability calculation for an uncertain system with only real uncertainties is needed.

2.3 Gain-scheduled Controller Design

Gain-scheduling is a common engineering practice used to control nonlinear plants in a variety of engineering applications. Bequette (1997) reviewed the traditional gain-scheduled process control. A typical gain-scheduled design procedure for nonlinear plants is as follows:

1. The designer selects several operating points which span the range of operation of the process.
2. At each of these operating points, the designer constructs a linear time-invariant approximation of the plant and designs a linear compensator for the linearized plant model.
3. In between operating points, the parameters or gains of the compensators are then interpolated, or scheduled, thus resulting in a global compensator applicable to the whole window of operation.

Since the local designs are based on linear time-invariant approximations to the plant, the designer may be able to guarantee that at each operating point, the feedback system has the needed feedback properties, such as stability and performance of the local linear model. However, since the actual system is nonlinear, the overall gain-scheduled system may not satisfy the stability and performance margins for the actual nonlinear process. In other words, one typically cannot assess *a priori* the guaranteed stability and performance properties of this traditional gain-scheduled design. Rather, any such properties have to be inferred from extensive computer simulations (Shamma and Athans, 1990).

In addition to simulations, gain-scheduled designs are guided by heuristic rules-of-thumb (Shamma and Athans, 1990). The two most fundamental guidelines are:

1. The scheduling variable should vary slowly.
2. The scheduling variable should be related to the plant's model nonlinearities.

These guidelines are simply reminders that the local operating point designs were based on linear time-invariant approximations to the actual plant. Thus, these approximations must be sufficiently accurate if one expects the local feedback properties to carry over to the overall gain-scheduled system.

Shamma and Athans (1990) analyzed two types of nonlinear gain-scheduled systems: 1) controller scheduling along a reference trajectory; and 2) controller scheduling based on the plant output. In each case, sufficient conditions were given which guarantee that the overall gain-scheduled system will retain the feedback properties of the local designs. These conditions formalize the rules-of-thumb, and again, the most fundamental idea behind the analysis is that the original designs are based on local linear time-invariant approximations of a nonlinear plant.

However, Shamma and Athans (1990, 1991) revealed certain limitations of this traditional gain-scheduling approach. More explicitly, the guidelines of “varying slowly” and “capturing the plant’s nonlinearity” in fact place fundamental limitations on the achievable performance of current gain-scheduling practices.

The case of the restriction to slow variations most likely is due to the nature of the scheduling algorithms. More precisely, the scheduling of controller gains is such that good performance may be expected for any fixed interpolated operating condition. However, performance may deteriorate rapidly as one experiences rapid changes throughout the range of operating conditions. Shamma and Athans (1992) analyzed the potential hazards of the traditional gain-scheduled designs, and pointed out that without a modification of the gain-scheduling design procedure, the aforementioned fundamental limitations will remain. If the possibility of fast parameter variations is not addressed in the design process, then guaranteed properties of the overall design cannot be established. The limitation of capturing the nonlinearities can be addressed through the appropriate selection of the scheduling variables.

In contrast with Shamma and Athans' work (1990, 1991 and 1992), in the current study, the scheduling variable is chosen to be the manipulated variable, which is able to capture the nonlinearity because it will be shown to be the sole source of the process nonlinearity in the state-affine models used along this work. A novel robust gain-scheduling design approach, which is different from the traditional gain-scheduling approach will be presented in this paper. This robust gain-scheduling design will be applied for the design of the widely-used PI controllers.

2.4 Robustness of MPC

MPC techniques widely used in the chemical industry are those based on the optimization of a quadratic objective function involving the error between the set-point and the predicted outputs. The success of linear MPC (LMPC) algorithms in industry has led to various extensions to handle nonlinear systems. Chen and Allgower (1998b) reviewed a number of nonlinear MPC (NLMPC) schemes, that address issues related to nominal or robust closed-loop stability.

For example, Mutha, Cluett and Penlidis (1997) designed a NLMPC algorithm to handle control nonaffine systems, i.e., nonlinear in the manipulated variable. The algorithm is based on a reinterpretation of the prediction equation as a Taylor series expansion. The key feature of this algorithm lies in the use of a process output prediction that accounts for changes in process dynamics as a function of the operating point as well as of the magnitude of the process input change.

Due to the presence of nonlinearities, a system behaves differently for different operating conditions. Closed-loop stability can be achieved by a suitable tuning of MPC design parameters such as prediction horizon, control horizon, and weighting matrices. However, the tuning for stability often can not deliver satisfactory performance for various different operating points. Thus, guaranteed stability and performance, independent of the choice of the operating point, is of great interest not only in theory, but also for practitioners.

Chen, Scherer and Allgower (1997) proposed a robust NLMPC scheme that can be conceptually viewed as a combination of NLMPC and nonlinear H_∞ control. This approach potentially combines the strengths of both methods, thus, the designed controllers have guaranteed robust stability and achieved good disturbance rejection in the face of input constraints. The major obstacle of this approach is the high on-line computational demand which prevents the industrial application of this method. The designed controllers are not able to optimize the performance index in terms of disturbance rejection.

A computationally attractive nonlinear MPC scheme for open-loop stable systems was proposed by Chen and Allgower (1998a), for the problem of stability. The open-loop optimal control problem was formulated as minimizing a finite horizon cost plus a terminal penalty term subject to nonlinear system dynamics and constraints. The terminal penalty term forces the system states at the end of the horizon to lie in a prescribed region around the system equilibrium point. The authors reported that there was no performance improvement introduced by this proposed algorithm.

However, a fundamental question that was not addressed by existing MPC-based control techniques, linear or nonlinear, is their robustness to model uncertainty and noise. Most known formulations of MPC minimize, on-line at each sampling step, a nominal objective function, using a single linear model (LMPC) or a nonlinear model (NLMPC) to predict the future plant behavior. Feedback, in the form of plant measurement at the next time step, is expected to account for the plant/model mismatch. Needless to say, such control systems that provide optimal performance for particular model may perform poorly when implemented on a physical system that is not exactly described by the model (Zheng and Morari, 1993). This section gives an overview of the attempts in the literature to provide MPC with some robustness guarantees in the presence of model uncertainty.

Broadly, the existing literature on robustness in MPC can be summarized as follows:

1. Analysis of robustness properties of MPC.

By using a contraction mapping theorem, Zafiriou (1990) derived a set of sufficient conditions for nominal and robust stability of MPC. Because the conditions are difficult to check, he also stated some necessary conditions associated to these sufficient conditions.

Gencilli and Nikolaou (1993) gave sufficient conditions for robust closed-loop stability and investigated robust performance of dynamic matrix control (DMC) systems with hard input/soft output constraints. The authors considered an ℓ_1 -norm performance index, a terminal state condition as a state constraint, and used an impulse-response model with bounds on the variations of the coefficients. They derived a robustness test in terms of simple inequalities to be satisfied. This simplicity is largely lost in the extension to the MIMO case.

Zanovello and Budman (1999) proposed a model predictive control algorithm which deals with soft constraints. The issues of nominal and robust stability of the control system were assessed offline. Robust stability was assessed using a structured singular value (μ) test. The model uncertainty was obtained from several step tests performed on the system around different operating conditions. The dimension of the problem studied by the authors is very large, so a frequency-domain μ test was used, which has the advantage of reducing the dimensions. However, the calculation has to be repeated at each frequency in the relevant range, and the uncertainty was assumed to be time-invariant.

2. Robust synthesis of MPC.

The basic philosophy in the literature for optimizing the performance of MPC-based design algorithms that explicitly account for plant uncertainty is to modify the on-line 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; Zheng and Morari, 1993).

Min-max robust MPC was first proposed by Campo and Morari (1987), and further developed by Zheng and Morari (1993), for SISO plants with finite impulse response (FIR), given uncertainty bounds on the impulse response coefficients. Kothare et al. (1996) applies this min-max formulation for polytopic/multi-model and structured feedback uncertainty. However, this approach has a few drawbacks. The first one is computational: solving the min-max problem for a family of plants is computationally much more demanding than solving it for a nominal plant. The second one is that the control action may be excessively conservative. To simplify the computational complexity, one must choose simplistic, albeit unrealistic, model uncertainty descriptions, e.g., fewer impulse response coefficients.

Another problem is the fact that, the above methods inherently assume that by solving the min-max problem to obtain a sequence of future inputs and then implementing the first one and repeating the computation at the next sampling point, one is guaranteed robust stability and performance, provided that a sufficiently long horizon is used in the objective function. However, feedback from an uncertain plant exists in reality and it is not taken into account in the formulation of the optimization problem, which is an open-loop minimization of the objective function over all possible plants. This fact can result in performance deterioration and instability of the actual closed-loop system. The problems cannot possibly be satisfactorily addressed without considering the problem in its proper nonlinear framework. Zafiriou (1990) argued that instead of augmenting the objective functions to account for robustness, an action that dramatically increases the computational load and at the same time produces no rigorous robustness guarantees, one should study the problem accounting for its nonlinear nature, i.e., obtain conditions that guarantee nominal and robust stability and performance and tune the parameters of the original MPC optimization problem accordingly.

The robust gain-scheduled MPC design approach proposed in this work addresses some of the above problems efficiently. The state-affine model, which depends nonlinearly on the manipulated variable u , is used to generate the process predictions. As a result, these

output predictions take into account explicitly the model uncertainty and approximate the feedback from the uncertain plant. In this work, to avoid the nonlinear optimization formulation, it is proposed to do predictions with step response models as done for the linear case. However, to account for the process nonlinearity, instead of using one step response model, a family of step response models will be defined for different sub-ranges based on the values of the manipulated variable u . This approach results in a simple gain-scheduled MPC strategy, which has not been reported in the literature to the author's knowledge. The key advantage is that in this work, global closed-loop stability and performance will be tested instead of testing only the local closed-loop stability and performance as proposed by practitioners for the traditional gain-scheduling approach. In addition, the input weight will be assumed as the tuning parameter scheduling against u .

3 Uncertain Dynamical Systems

Two classes of uncertain dynamical systems are of particular relevance to this work, which are affine parameter-dependent models and linear-fractional models. In this work, a nonlinear process will be represented by a state-affine model, which depends nonlinearly on the manipulated variable. If the process nonlinearity is treated as uncertainty, the uncertainty can then be quantified from the nonlinear terms of the state-affine model. The process nonlinearity, i.e., the process uncertainty, is shown to be a polynomial function of the current input only and this facilitates the calculation of the uncertainty bounds.

The state-affine model has the form of parameter-dependent systems with affine parameter-dependence on the uncertain parameters. This allows for the formulation of a Lyapunov stability and performance test in terms of a finite set of Linear Matrix Inequalities (LMIs). The state-affine model can also be transformed into linear-fractional models, such that the robustness analysis based on SSV approach can also be applied.

This chapter is organized as follows. In section 3.1, model uncertainty is briefly reviewed. One class of uncertain dynamical systems, i.e., parameter-dependent models, is introduced in section 3.2. Methods of quantifying uncertainty are developed from the state-affine model in section 3.2. The other class of uncertain dynamical systems, i.e., linear-fractional models, is discussed in section 3.4. The identification algorithm of the state-affine model from process input/output data, through an intermediate step of Volterra series model identification, is summarized in section 3.5. The case study process, a CSTR, is introduced in section 3.5. A state-affine model is identified for the nonlinear CSTR. This state-affine model will be used throughout the work to illustrate the different theoretical developments proposed in this thesis. The uncertainty expression and its bounds are also given in this section.

3.1 Model Uncertainty

The notion of uncertain dynamical systems is central to robust control theory. For control design purposes, the possibly complex behavior of dynamical systems is often approximated by models of relatively low complexity. The difference between a process model and the true physical system behavior is called model uncertainty. A key cause of uncertainty is the imperfect knowledge of some parameters of the system, or their variability due to changes in operating conditions, fouling, etc. Note that model uncertainty should be distinguished from variable exogenous actions such as disturbances or measurement noise.

The current work focuses on the class of dynamical systems that can be approximated by linear models, which in combination with a model uncertainty description, may represent the behavior of the real system. When deriving the nominal linear model and estimating the uncertainty, two fundamental principles must be remembered:

- Uncertainty should be small where high performance is desired, i.e., there is a tradeoff between performance and robustness.
- The more information one has about the uncertainty, e.g., phase, structure, time invariance, etc., the higher the achievable performance will be.

There are two major classes of uncertainty:

- Dynamical uncertainty, which consists of dynamical components not accounted for by the linear model due to e.g., nonlinear behavior as well as variations in the dynamical behavior during operation.
- Parameter uncertainty, which stems from imperfect knowledge of the physical parameter values, or from variations of these parameters during operation. Examples of physical parameters include stiffness and damping coefficients in mechanical systems,

aerodynamical coefficients in flying devices, capacitors and inductors in electric circuits, etc.

Some important characteristics of uncertainty include whether it is linear or nonlinear, and whether it is time-invariant or time-varying. Model uncertainty is generally a combination of dynamical and parametric uncertainty, and may arise at different points in the control loop. For instance, there may be dynamical uncertainty on the system actuators, and parametric uncertainty on some sensor calibration coefficients.

Two representations of model uncertainty can be used in robust control designs:

- Uncertain state-space models. This representation is relevant for systems described by dynamical equations with uncertain and/or time-varying coefficients.
- Linear-fractional representation of uncertainty. Here the uncertain system is described as an interconnection of known LTI (linear time-invariant) systems with uncertain components called “uncertainty blocks”. Each uncertainty block $\Delta_i(\cdot)$ represents a family of systems of which only a few characteristics are known. For instance, the only available information about $\Delta_i(\cdot)$ may be that it is a time-invariant nonlinearity with gain less than 0.01.

Determining factors in the choice of representation include the available model, e.g., state-space equations, frequency-domain models, etc., and the analysis or synthesis tool to be used. In the current work, state-affine models are obtained to represent the physical nonlinear process. This model has the form of uncertain state-space models, and is suitable for the robust stability and performance conditions based on quadratic Lyapunov functions. The state-affine model can also be transformed into linear-fractional models, such that the robustness analysis based on SSV approach can also be applied.

3.2 Uncertain State-space Models

The nonlinear process is assumed to be modeled by a state-space nonlinear model. The resulting state-space equations typically involve physical parameters whose values are only approximately known, as well as approximations of nonlinear or more complex phenomena. In other words, the system is described by an uncertain state-space model:

$$\begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{e}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{v}(t) \end{bmatrix} \quad (3.1)$$

where the state-space matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ depend affinely on uncertain and/or time-varying parameters, or vary in some bounded sets of the space of matrices. The class of parameter-dependent models is of particular relevance to this work and it is discussed next.

3.2.1 Affine Parameter-dependent models

Affine parameter-dependent models (PDS) are of the following form:

$$\begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{e}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}(\boldsymbol{\delta}_t) & \mathbf{B}(\boldsymbol{\delta}_t) \\ \mathbf{C}(\boldsymbol{\delta}_t) & \mathbf{D}(\boldsymbol{\delta}_t) \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{v}(t) \end{bmatrix} \quad (3.2)$$

where $\mathbf{A}(\cdot), \mathbf{B}(\cdot), \mathbf{C}(\cdot), \mathbf{D}(\cdot)$ are known functions of some uncertain parameter vector $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \dots, \delta_n) \in \mathbf{R}^n$. This work focuses on analyzing the stability and performance of parameter-dependent models with an affine dependence on the parameter vector $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \dots, \delta_n)$, that is, a PDS where:

$$\mathbf{A}(\boldsymbol{\delta}) = \mathbf{A}_0 + \mathbf{A}_1 \delta_1 + \mathbf{A}_2 \delta_2 \dots + \mathbf{A}_n \delta_n \quad (3.3)$$

Affine parameter-dependent models are well-suited for Lyapunov-based analysis and synthesis and can be also easily converted to linear-fractional uncertainty models for Structured Singular Value (SSV) based analysis.

In this work, a state-affine model is identified to represent the nonlinear process, and the nonlinearity of the process is considered as the main source of model uncertainty. The development of the uncertainty description to account for nonlinearity of the state-affine model is illustrated in the sequel. For nonlinear processes, Budman and Knapp (2000, 2001) proposed the use of state-affine models as follows:

$$\begin{aligned}\mathbf{x}(t+1) &= \{\mathbf{F}_0 + \sum_{i=1}^{n-1} \mathbf{F}_i u(t)^i\} \mathbf{x}(t) + \{\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} u(t)^i\} u(t) \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t)\end{aligned}\tag{3.4}$$

where $\mathbf{F}_i, \mathbf{G}_i, \mathbf{H}_i$ are model coefficients, $\mathbf{x}(t)$ are the process states, $y(t)$ is the output, $u(t)$ is the manipulated variable. This model given in equation (3.4) can be easily identified from input/output data using the methodology proposed by Sontag (1978), and Budman and Knapp (2000, 2001).

For a process given by the state-affine model (3.4), it is valid to assume that in a small neighborhood of a pre-selected nominal operating point, i.e., for $|u(t)| \ll 1$, the process can be accurately modeled by the linear part of the state-affine model given as follows:

$$\begin{aligned}\mathbf{x}(t+1) &= \mathbf{F}_0 \mathbf{x}(t) + \mathbf{G}_1 u(t) \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t)\end{aligned}\tag{3.5}$$

The uncertainty of the system will be assumed to be the difference between the nonlinear model given by equation (3.4) and the nominal linear model defined by equation (3.5). It is also assumed that all of the uncertainty in the state-affine model is due to the time-

varying nonlinearity of the state-affine model around this operating point. It is therefore possible to describe the model uncertainty perturbation $\delta_{i,t}$ in the following form:

$$\delta_{i,t} = u(t)^i \quad (3.6)$$

Equation (3.6) represents the key advantage of using the state-affine model given by equation (3.4) to model the system. Generally it is not trivial to quantify the uncertainty $\delta_{i,t}$ from mechanistic first-principles models (Doyle, 1990). In our case, since $\delta_{i,t}$ is equal to the powers of the current input, it can be easily quantified. Each input in a process is generally bounded between a lower limit and an upper limit known during the design stage due to, for example, actuator constraints or economic considerations. Specifically, according to equation (3.4), each parameter $\delta_{i,t}$ ranges between *a priori* known extreme values $\underline{\delta}_i$ and $\overline{\delta}_i$ as follows:

$$u(t) \in [\underline{u} \quad \overline{u}] \rightarrow \delta_{i,t} \in [\underline{\delta}_i, \overline{\delta}_i] \quad (3.7)$$

In summary, the major motivation for representing a nonlinear process with the state-affine model given by equation (3.4) derives from the fact that, the uncertainty is shown to be a function of the current input only, and it can be easily quantified with known bounds. The nonlinearity of the process is treated as model uncertainty and thus a robust control design approach can be applied.

Rewriting equation (3.4) using the uncertainty expression (3.6) gives:

$$\begin{aligned} \mathbf{x}(t+1) &= \{\mathbf{F}_0 + \sum_{i=1}^{n-1} \mathbf{F}_i \delta_{i,t}\} \mathbf{x}(t) + \{\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t}\} u(t) \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t) \end{aligned} \quad (3.8)$$

Or equivalently by the following parameter-dependent model:

$$\begin{bmatrix} \mathbf{x}(t+1) \\ y(t) \end{bmatrix} = \begin{bmatrix} \mathbf{F}(\boldsymbol{\delta}_t) & \mathbf{G}(\boldsymbol{\delta}_t) \\ \mathbf{H}_0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ u(t) \end{bmatrix} \quad (3.9)$$

where $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \dots, \delta_{n-1}) \in \mathbf{R}^{n-1}$ is the uncertain parameter vector, $\mathbf{F}(\cdot), \mathbf{G}(\cdot)$ are polynomial matrices which depend affinely on the uncertain parameters, for example, $\mathbf{F}(\boldsymbol{\delta}_t) = \mathbf{F}_0 + \mathbf{F}_1\delta_1 + \mathbf{F}_2\delta_2 + \dots + \mathbf{F}_{n-1}\delta_{n-1}$.

3.2.2 Quantification of model uncertainty

Parameter uncertainty can be quantified based on the range of parameter values and possibly based on the rates of parameter variation. The parameter uncertainty range can be described as a hyper-rectangle in the parameter space. This corresponds to the case where each uncertain and/or time-varying parameter ranges between two empirically determined limits. Specifically, according to equation (3.6), each parameter $\delta_{i,t}$ ranges between *a priori* known extreme values $\underline{\delta}_i$ and $\bar{\delta}_i$, i.e., $u(t) \in [\underline{u} \ \bar{u}] \rightarrow \delta_{i,t} \in [\underline{\delta}_i, \bar{\delta}_i]$.

If $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \dots, \delta_n) \in \mathbf{R}^n$ is the vector of all uncertain parameters of $\boldsymbol{\delta}_t$, equation (3.7) delimits a hyper-rectangle of the parameter space \mathbf{R}^n called the parameter box. In the sequel, \mathbf{W} denotes the 2^n vertices or corners of this parameter box as follows:

$$\mathbf{W} := \{(\omega_1, \omega_2, \dots, \omega_n) : \omega_i \in [\underline{\delta}_i, \bar{\delta}_i]\} \quad (3.10)$$

Similarly, it is assumed that the rate of variation $\Delta\boldsymbol{\delta}_t$ is well defined at all time-intervals and satisfies:

$$\Delta\delta_{i,t} = \delta_{i,t+1} - \delta_{i,t}, \Delta\delta_{i,t} \in [\underline{v}_i \ \bar{v}_i] \quad (3.11)$$

where $\underline{v}_i, \bar{v}_i$ are *a priori* known lower and upper bounds on this rate of variation. To handle the time-varying case with less conservatism when the knowledge of parameter-variation is available, the rates $\Delta\delta_t = (\Delta\delta_{1,t}, \Delta\delta_{2,t}, \dots, \Delta\delta_{K,t}) \in \mathbf{R}^n$ are considered as additional time-varying uncertain parameters. As a whole, the vector $\Delta\delta_t$ evolves in a n -dimensional hyper-rectangle whose vertices are in the set:

$$\mathbf{S} := \{(\tau_1, \tau_2, \dots, \tau_n) : \tau_i \in [\underline{v}_i, \bar{v}_i]\} \quad (3.12)$$

3.3 Linear-Fractional Models of Uncertainty

For systems with both dynamical and parametric uncertainty, a general representation of uncertainty is the linear-fractional model of Figure 3.1. In this linear fractional transformation (LFT) representation, the linear time-invariant (LTI) system $\mathbf{M} \in \mathbf{C}^{n \times n}$ represents all the known LTI components including the controller, the nominal models of the systems, sensors, and actuators. The input vector \mathbf{d} includes all external actions on the system, i.e., disturbance, noise and reference signal, and the vector \mathbf{e} consists of all output signals generated by the system. The uncertainty block $\Delta = \text{diag}(\Delta_1, \dots, \Delta_n)$, which satisfies $\bar{\sigma}(\Delta_i) \leq 1$, is a norm-bounded LTI uncertainty with some prescribed structure. $\bar{\sigma}$ denotes the maximum singular value of a matrix. Each uncertainty block Δ_i accounts for one particular source of uncertainty, e.g., neglected dynamics, nonlinearity, uncertainty parameters, etc. The diagonal structure of Δ reflects how each uncertainty component Δ_i enters the loop and affects the overall behavior of the true system.

In linear-fractional uncertainty models, each block Δ_i of $\Delta = \text{diag}(\Delta_1, \dots, \Delta_n)$ is a dynamical system characterized by the following aspects:

- The dynamical nature: linear time-invariant or time-varying, nonlinear;

- The dimensions and structure: full block or repeated scalars block $\Delta_i = \delta_i \mathbf{I}$. Scalars blocks are used to represent uncertainty parameters.
- Whether Δ_i is a complex or real-values matrix
- Quantitative information such as norm bounds.

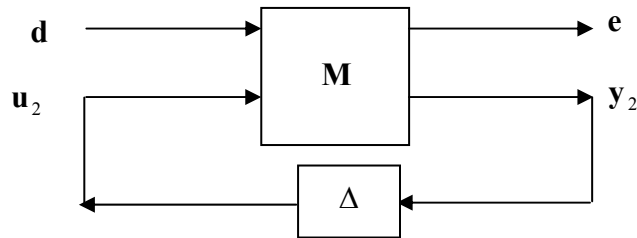


Figure 3.1 General $\mathbf{M} - \Delta$ LFT framework

For systems with linear time-invariant linear-fractional uncertainty, SSV analysis investigates the robust stability and performance of. SSV approach is also applicable to parameter-dependent systems based on an equivalent LFT representation. The general procedure to derive a linear-fractional model of an uncertain state-space model is illustrated in section 4.4.2. A state-affine model can also be transformed into a linear-fractional model, and the procedure is given in section 5.3.1.

3.4 Model Identification Methodology

3.4.1 Volterra series models

The algorithm (Sontag, 1978) used in this work to find state-affine models is based on an intermediate step where a Volterra series model is identified. Thus, an algorithm to find a Volterra series must be explained first. A Volterra series model relates the output of a process to a polynomial of past inputs. A Volterra series model has the form:

$$\begin{aligned}
y(t) = & h_0 + \sum_{i=1}^{\infty} h_i u(t-i) + \sum_{i=1}^{\infty} \sum_{j=i}^{\infty} h_{ij} u(t-i)u(t-j) + \\
& \sum_{i=1}^{\infty} \sum_{j=i}^{\infty} \sum_{k=j}^{\infty} h_{ijk} u(t-i)u(t-j)u(t-k) + \dots
\end{aligned} \tag{3.13}$$

where h_0, h_i, h_{ij} are the 0th-order, 1st-order, 2nd-order Volterra kernels. The maximum power of the inputs on the right hand side of equation (3.13) is referred to as the nonlinearity order n , Assuming that there is no immediate response of the manipulated input, the coefficient h_0 is zero. If the series is truncated to finite M time steps into the past, it is possible to estimate the Volterra kernels from input/output data. A Volterra series model which has only 2nd-order terms is given as follows:

$$y(t) = \sum_{i=1}^M h_i u(t-i) + \sum_{i=1}^M \sum_{j=i}^M h_{ij} u(t-i)u(t-j) + \eta_t \tag{3.14}$$

where $\{y(t)\}$ is the observed output sequence associated with the input sequence $\{u(t)\}$. $\{\eta_t\}$ is an observation noise sequence that is independent of the input.

Suppose that the system output is being observed beginning at time t and data are collected over an observation period of $\tau > 0$. Then, the vector of outputs, $\mathbf{Y}_t = [y(t), \dots, y(t + \tau)]^T$ is related to the input by:

$$\begin{aligned}
\mathbf{Y}_t &= \mathbf{X}_t \boldsymbol{\theta} + \boldsymbol{\eta}_t \\
\mathbf{X}_t &= [\mathbf{x}_t^T, \dots, \mathbf{x}_{t+\tau}^T]^T \\
\mathbf{x}_t &= [u(t-1) \quad \dots \quad u(t-M) \quad u(t-1)u(t-1) \quad \dots \quad u(t-M)u(t-M)] \\
\boldsymbol{\theta} &= [h_1 \quad \dots \quad h_M \quad h_{11} \quad \dots \quad h_{MM}]^T
\end{aligned} \tag{3.15}$$

where $\boldsymbol{\eta}_t$ is a vector containing samples of the observation noise sequence, \mathbf{X}_t is the data matrix, and $\boldsymbol{\theta}$ is the parameter vector. If \mathbf{X}_t is full rank, then the least squares estimate of the parameters is given by:

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}_t^T \mathbf{X}_t)^{-1} \mathbf{X}_t^T \mathbf{Y}_t \quad (3.16)$$

In case of $E(\boldsymbol{\eta}_t) = 0$, i.e., the noise has zero-mean, the estimate given by equation (3.16) is an un-biased estimate.

3.4.2 State-affine models

Once the Volterra kernels are obtained from least squares regression, a generalized Behavior matrix $\mathbf{B}(f)$ must be developed in order to find a state-affine model, where f denotes a finite input/output response. A Behavior matrix is a block matrix constructed as follows. The rows of $\mathbf{B}(f)$ are indexed by $[\mathbf{J}]^+$ and the columns of $\mathbf{B}(f)$ are indexed by $[\mathbf{J}]_+$, where:

$$[\mathbf{J}]^+ = \{0,1,\dots,n,00,01,\dots,0n,10,\dots,nn,000,\dots\} \quad (3.17)$$

$$[\mathbf{J}]_+ = \{1,2,\dots,n,10,11,\dots,1n,20,\dots,nn,100,\dots\} \quad (3.18)$$

where n is the maximum order of the Volterra series. The $\beta\alpha^{th}$ entry (denoted by $b_{\beta\alpha}$, row β , column α) of $\mathbf{B}(f)$ contains the coefficient $a_{\alpha\beta}$ which corresponds to a Volterra kernel in equation (3.13). For example, a_{110} corresponds to b_{011} , i.e., $\beta = 0, \alpha = 11$. To illustrate this index notation, Table 3.1 shows the first few blocks of $\mathbf{B}(f)$ for $n = 2$.

The Volterra kernels are placed in the matrix based upon the Volterra terms to which they correspond. The locations of the nonzero terms in the index $\alpha\beta$ indicate the number of

time steps in the past that each input in the polynomial term represents while the magnitudes of the nonzero values in the index represent the power of the corresponding term. For example, the Volterra kernel h_{1123} , which corresponds to the input term $u^1(t-3)u^1(t-2)u^2(t-1)u^0(t)$, would have a Behavior matrix entry of a_{1120} . Once the Behavior matrix is properly constructed, a state-affine model may be obtained based on the algorithm explained in the sequel.

Table 3.1 Example of Behavior matrix $\mathbf{B}(f)$

	1	2	10	11	12	22	...
0	a_{10}	a_{20}	a_{100}	a_{110}	...		
1	a_{11}	a_{21}	a_{101}	a_{111}	...		
2	a_{12}	a_{22}	a_{102}	a_{112}	...		
00	a_{100}	a_{200}	a_{1000}	a_{1100}	...		
01	a_{101}	a_{201}	a_{1001}	a_{1101}	...		
02	\vdots	\vdots	\vdots	\vdots			
\vdots							

Sontag (1978) proposed an algorithm to find a state-affine model given a properly constructed Behavior matrix. Let $\boldsymbol{\phi}$ be an $m \times m$ nonsingular sub-matrix of $\mathbf{B}(f)$ and let $\alpha_i, i = 1, \dots, m$ denote the rows of $\boldsymbol{\phi}$ and let $\beta_i, i = 1, \dots, m$ denote the columns of $\boldsymbol{\phi}$. A state-affine model may then be determined from the realization:

$$\begin{aligned}
 \mathbf{F}_i &= \boldsymbol{\phi}^{-1} \boldsymbol{\phi}_i, i = 0, \dots, m-1 \\
 \mathbf{G}_i &= \boldsymbol{\phi}^{-1} [a_{i\beta_1}, \dots, a_{i\beta_m}]^T, i = 1, \dots, n \\
 \mathbf{H}_i &= [a_{\alpha_1 i}, \dots, a_{\alpha_m i}], i = 0, \dots, n
 \end{aligned} \tag{3.19}$$

where $\boldsymbol{\phi}_i$ is a sub-matrix of $\mathbf{B}(f)$ with the same rows as $\boldsymbol{\phi}$ but with the columns indexed by $\alpha_1 i, \dots, \alpha_m i$.

The state-affine models are found recursively by the following algorithm:

1. Find a nonzero row of $\mathbf{B}(f)$ and define $\boldsymbol{\varphi}$ and $\boldsymbol{\varphi}_i$, $i = 0, \dots, n$. Set $m = 1$ (model of dimension = 1);
2. Find a state-affine model using equation (3.19);
3. Add a row to $\boldsymbol{\varphi}$ by choosing the next available row of $\mathbf{B}(f)$. Find the rank of $\boldsymbol{\varphi}$. If $\boldsymbol{\varphi}$ is full rank, keep the row and proceed with step 2. If $\boldsymbol{\varphi}$ is not full rank, remove the most recently added row and add the next available row from $\mathbf{B}(f)$. Repeat this procedure until the rank of $\boldsymbol{\varphi}$ increases. This step is necessary to ensure that $\boldsymbol{\varphi}$ is nonsingular and therefore $\boldsymbol{\varphi}^{-1}$ can be calculated.

The result of this algorithm is a model of the form:

$$\begin{aligned} \mathbf{x}(t+1) &= \sum_{i=0}^n \mathbf{F}_i u(t)^i \mathbf{x}(t) + \sum_{i=1}^n \mathbf{G}_i u(t)^i \\ y(t) &= \sum_{i=0}^n \mathbf{H}_i u(t)^i \mathbf{x}(t) \end{aligned} \quad (3.20)$$

Assuming that for a physical system there is no instantaneous response, i.e., the output at time t , $y(t)$, is not affected by the input at time t , $u(t)$, the state-affine model in equation (3.20) can be simplified to:

$$\begin{aligned} \mathbf{x}(t+1) &= \sum_{i=0}^n \mathbf{F}_i u(t)^i \mathbf{x}(t) + \sum_{i=1}^n \mathbf{G}_i u(t)^i \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t) \end{aligned} \quad (3.21)$$

3.5 Case Study

3.5.1 Nonlinear process: CSTR

The issue of model identification and uncertainty quantification is illustrated for a specific process. The case study under investigation is a CSTR with a first-order exothermic reaction. Doyle, Packard and Morari (1989) provided an example of a CSTR with single-input-single-output (SISO) for which the dynamic behavior can be described using the following non-dimensional normalized equations representing the component and energy balances respectively:

$$\begin{aligned}\dot{x}_1 &= -x_1 + D_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\gamma}\right) \\ \dot{x}_2 &= -x_2 + BD_a(1-x_1)\exp\left(\frac{x_2}{1+x_2/\gamma}\right) - \beta(x_2 - x_c) \\ y &= x_1\end{aligned}\tag{3.22}$$

where the states x_1 and x_2 are the dimensionless reactant concentration and reaction temperature respectively, and the input or manipulated variable $u = x_c$ is the dimensionless temperature of the cooling jacket surrounding the reactor. In this work, the reactant concentration x_1 was selected as the controlled variable.

A summary of the variables used in the above equations is summarized in the nomenclature in Appendix A. The process has one stable steady state when $D_a = 0.072$, $B = 1.0$, $\beta = 0.3$ and $\gamma = 20.0$.

Seborg et al. (1989) illustrated that the CSTR is an important nonlinear process because it embodies many of the features of more commonly encountered reaction systems. At the same time, CSTR models, although highly nonlinear, tend to be simpler than models for other types of continuous reactors, such as packed-bed reactors that are modeled by

partial differential equations. Therefore, CSTR's have been used to illustrate new control algorithms for nonlinear systems.

Open-loop simulation of the output x_1 is shown in Figure 3.2 under an initial condition of $[0.4759, 2.9045]$. The CSTR has one stable steady state at $[x_{10}, x_{20}] = [0.0706, 0.0543]$ and a settling time of 9 sampling intervals.

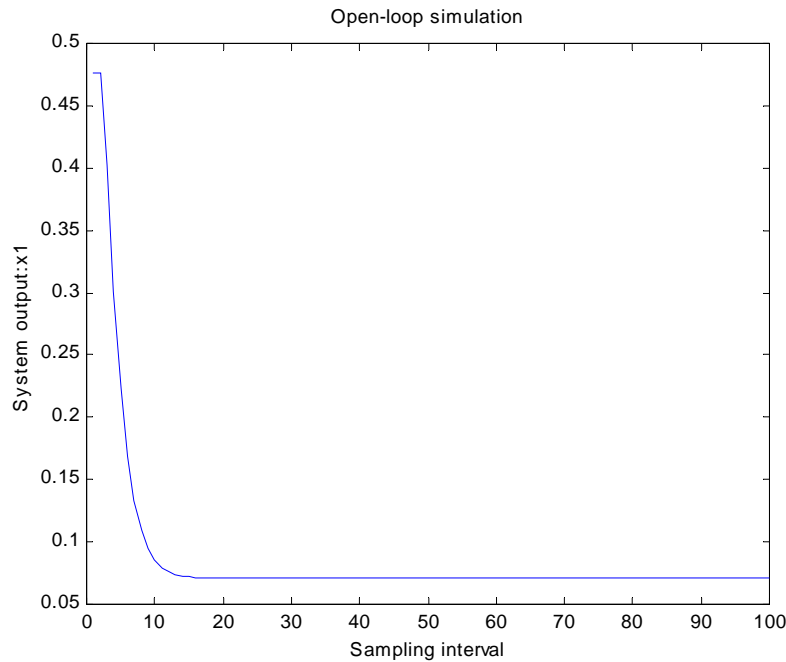


Figure 3.2 Open loop simulation

3.5.2 Volterra series models of the CSTR

A Volterra series model can be identified from the simulated input/output data of the nonlinear CSTR, using the first-principles model given by equation (3.22). Following the guidelines of Novak and Van Veen (1994), an $n + 1$ -level PRMS input is created and the process output y is simulated using equation (3.22). n is the maximum power of the inputs of the Volterra series model given by equation (3.14). As a start, it is chosen to be $n = 2$ for equation (3.14), and it can be increased as necessary based on comparisons of

the model predictions with the actual measured data. For M , it is reasonable to choose it such that it is equal to or larger than the system settling time. Based on this fact, M is set to 9 (time steps) for the CSTR system under study. The model obtained has the following form:

$$y(t) = \sum_{i=1}^M h_i u(t-i) + \sum_{i=1}^M \sum_{j=i}^M h_{ij} u(t-i) u(t-j) + \eta_t \quad (3.14)$$

Since $n = 2$, the PRMS for identification has to have at least 3 levels to guarantee persistent excitation. The 3-level PRMS of inputs $x_c = [5,14,23]$ is generated and shown in Figure 3.3. The output of the system for the PRMS input is predicted using the CSTR equations (3.22). The input and output data are shown in Figure 3.3. The above simulation data are normalized to the range of $[-1,1]$ before being used for identification.

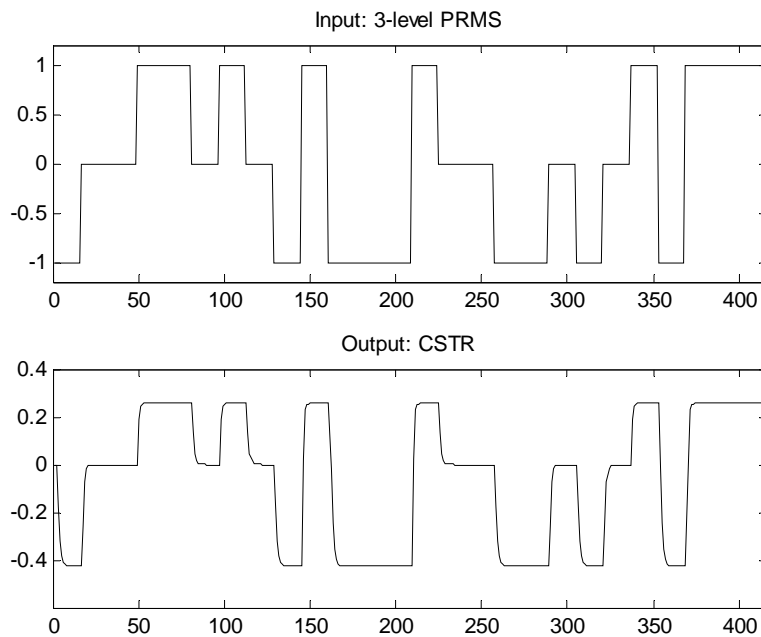


Figure 3.3 Input/output data

Based on least squares algorithm given in equation (3.16), a Volterra series model was obtained with kernels listed in Table 3.2. The number of parameters to be identified is the number of the different elements in Table 3.2, and it is calculated as follows. The 1st-row of Table 3.2 consists of M 1st-order Volterra kernels. The lower $M \times M$ symmetric matrix consists of M elements on the diagonal, and $(\frac{M \times M - M}{2})$ elements in the upper triangle block, i.e., $M + (\frac{M \times M - M}{2})$ 2nd-order Volterra kernels. In summary, the number of parameters to be identified is a nonlinear function of the memory length, and can be calculated from $M + M + (\frac{M \times M - M}{2})$. Thus, when $M = 9$, the total number of parameters is 54.

Table 3.2 Volterra kernels ($M = 9$)

I	1	2	3	4	5	6	7	8	9
h_i	0.4566	0.2837	0.0919	0.0361	0.0105	0.0021	0.0012	0.0006	-0.0008
$h_{i,1}$	0.0782	-0.0064	-0.0367	-0.0058	-0.0017	-0.0010	0.0023	0.0021	-0.0033
$h_{i,2}$		-0.1036	0.0042	-0.0026	-0.0038	-0.0006	-0.0016	-0.0002	0.0013
$h_{i,3}$			-0.0062	-0.0023	0.0039	-0.0084	-0.0017	0.0004	0.0001
$h_{i,4}$				0.0004	-0.0034	0.0176	-0.0030	-0.0038	0.0039
$h_{i,5}$					0.0115	-0.0085	-0.0038	0.0015	-0.0020
$h_{i,6}$						0.0066	0.0015	0.0052	-0.0025
$h_{i,7}$							0.0073	0.0019	-0.0137
$h_{i,8}$								-0.0063	0.0054
$h_{i,9}$									0.0378

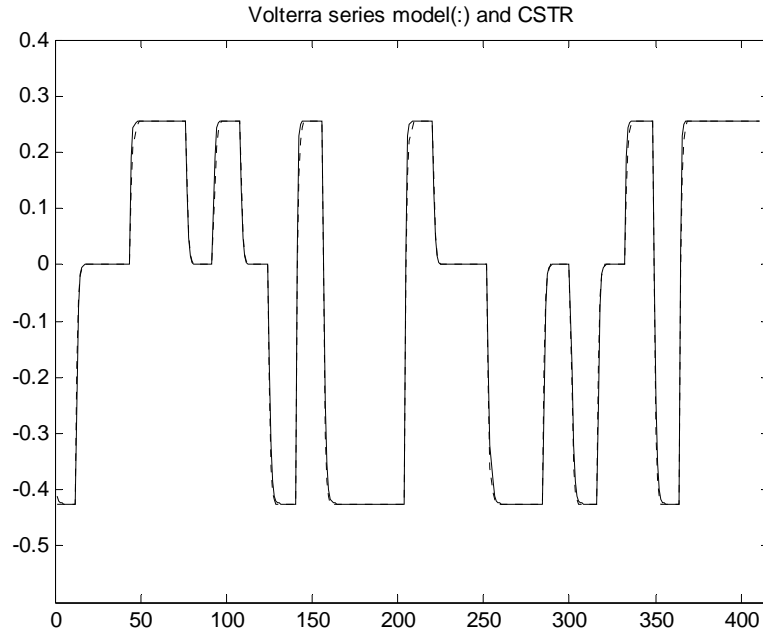


Figure 3.4 CSTR process output (solid line) and Volterra series model output (dotted line)

Simulation result of this Volterra series model is given in Figure 3.4 , and compared to the real CSTR process output. The sum of squared errors is 1.2019, calculated using $\sum_i (y(i) - \hat{y}(i))^2$, which is 0.5% of the sum of squares output. $y(i)$ is the real process output data used for identification, and $\hat{y}(i)$ is the prediction of the identified model for the same input.

3.5.3 State-affine models of the CSTR

Using the algorithm described in section 3.4.2, a state-affine model given by equation (3.4) will be generated for the process CSTR, using the Volterra kernels obtained in the previous section.

3.5.3.1 Behavior matrix of 1st-order Volterra kernels

Initially, to simplify the explanation of the procedure, only the 1st-order Volterra kernels are considered. The 1st-order Volterra kernels with corresponding values given in Table 3.2 and the corresponding Behavior matrix terms are shown in Table 3.3.

Table 3.3 1st-order Volterra kernels and corresponding Behavior matrix entries

Volterra kernel	Values	Behavior matrix entry
h_1	0.4566	a_{10}
h_2	0.2837	a_{100}
h_3	0.0919	a_{1000}
h_4	0.0361	a_{10000}
h_5	0.0105	a_{100000}
h_6	0.0021	$a_{1000000}$
h_7	0.0012	$a_{10000000}$
h_8	0.0006	$a_{100000000}$
h_9	-0.0008	$a_{1000000000}$

It should be noticed that, since there are no 2nd-order or higher order entries in the Behavior matrix when only 1st-order Volterra kernels are considered, there will be many rows and columns of zeros. It is possible to remove these rows and columns without affecting the state-affine modeling algorithm. A Behavior matrix with the rows and columns of zeros removed is therefore:

$$\mathbf{B}(f) = \begin{bmatrix} a_{10} & a_{100} & a_{1000} & a_{10000} & a_{100000} \\ a_{100} & a_{1000} & a_{10000} & a_{100000} & a_{1000000} \\ a_{1000} & a_{10000} & a_{100000} & a_{1000000} & a_{10000000} \\ a_{10000} & a_{100000} & a_{1000000} & a_{10000000} & a_{100000000} \\ a_{100000} & a_{1000000} & a_{10000000} & a_{100000000} & a_{1000000000} \end{bmatrix} \quad (3.23)$$

The indices of the rows of this reduced Behavior matrix, corresponding to the column indices of $a_{\alpha\beta}$, i.e., β , are as follows:

$$[\mathbf{J}]^+ = \{0,00,000,0000,00000\} \quad (3.24)$$

and the indices of the columns of this reduced Behavior matrix, corresponding to the row indices of $a_{\alpha\beta}$, i.e., α , are as follows:

$$[\mathbf{J}]_+ = \{1,10,100,1000,10000\} \quad (3.25)$$

State-affine models can now be found recursively using the algorithm described in section 3.4.2 and illustrated in the sequel for the 1st-order Volterra case, using the data given in Table 3.3.

1. Take a 1×1 nonzero sub-matrix of $\mathbf{B}(f)$:

$$\boldsymbol{\varphi} = a_{10} = 0.4566 \xrightarrow{\text{rank}(\boldsymbol{\varphi})=1} \alpha_1 = 1, \beta_1 = 0 \quad (3.26)$$

Obviously since $a_{10} = 0.4566$ is nonzero, the matrix $\boldsymbol{\varphi}$ is full rank. By substitution of the values from Table 3.3, the state-affine model coefficients are as follows:

$$\boldsymbol{\varphi}_0 = a_{100} \Rightarrow \mathbf{F}_0 = \boldsymbol{\varphi}^{-1} \boldsymbol{\varphi}_0 = a_{100} / a_{10} \quad (3.27)$$

$$\boldsymbol{\varphi}_1 = 0 \Rightarrow \mathbf{F}_1 = 0$$

$$\mathbf{G}_1 = \boldsymbol{\varphi}^{-1}[a_{10}]$$

$$\mathbf{H}_0 = a_{10}$$

$$\boldsymbol{\varphi}_0 = 0.2837 \Rightarrow \mathbf{F}_0 = 0.6213 \quad (3.28)$$

$$\boldsymbol{\varphi}_1 = 0 \Rightarrow \mathbf{F}_1 = 0$$

$$\mathbf{G}_1 = 1$$

$$\mathbf{H}_0 = 0.4566$$

2. To find the model with dimension 2, one row is added to $\boldsymbol{\varphi}$ and the rank of the matrix is computed. In this example,

$$\boldsymbol{\varphi} = \begin{bmatrix} a_{10} & a_{100} \\ a_{100} & a_{1000} \end{bmatrix} = \begin{bmatrix} 0.4566 & 0.2837 \\ 0.2837 & 0.0919 \end{bmatrix} \quad (3.29)$$

$$\underline{\text{rank}(\boldsymbol{\varphi}) = 2} \begin{matrix} \alpha_1 = 1, \alpha_2 = 10 \\ \beta_1 = 0, \beta_2 = 00 \end{matrix}$$

Again the following state-affine model is found:

$$\boldsymbol{\varphi}_0 = \begin{bmatrix} a_{100} & a_{1000} \\ a_{1000} & a_{10000} \end{bmatrix} \Rightarrow \mathbf{F}_0 = \boldsymbol{\varphi}^{-1} \boldsymbol{\varphi}_0 \quad (3.30)$$

$$\boldsymbol{\varphi}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Rightarrow \mathbf{F}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{G}_1 = \boldsymbol{\varphi}^{-1} \begin{bmatrix} a_{10} \\ a_{100} \end{bmatrix}, \mathbf{G}_2 = \boldsymbol{\varphi}^{-1} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{H}_0 = [a_{10} \quad a_{100}]$$

$$\boldsymbol{\varphi}_0 = \begin{bmatrix} 0.2837 & 0.0919 \\ 0.0919 & 0.0361 \end{bmatrix} \Rightarrow \mathbf{F}_0 = \boldsymbol{\varphi}^{-1} \boldsymbol{\varphi}_0 = \begin{bmatrix} 0 & 0.0466 \\ 1 & 0.2489 \end{bmatrix} \quad (3.31)$$

$$\boldsymbol{\varphi}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Rightarrow \mathbf{F}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{G}_1 = \boldsymbol{\varphi}^{-1} \begin{bmatrix} 0.4566 \\ 0.2837 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{G}_2 = \boldsymbol{\varphi}^{-1} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{H}_0 = [0.4566 \quad 0.2837]$$

3. As a final illustration, two more rows are added to the above $\boldsymbol{\varphi}$:

$$\Phi = \begin{bmatrix} a_{10} & a_{100} & a_{1000} & a_{10000} \\ a_{100} & a_{1000} & a_{10000} & a_{100000} \\ a_{1000} & a_{10000} & a_{100000} & a_{1000000} \\ a_{10000} & a_{100000} & a_{1000000} & a_{10000000} \end{bmatrix} \quad (3.32)$$

which is a full rank matrix as follows, after the substitution of the values of the Volterra kernels:

$$\Phi = \begin{bmatrix} 0.4566 & 0.2837 & 0.0919 & 0.0361 \\ 0.2837 & 0.0919 & 0.0361 & 0.0105 \\ 0.0919 & 0.0361 & 0.0105 & 0.0021 \\ 0.0361 & 0.0105 & 0.0021 & 0.0012 \end{bmatrix} \xrightarrow{\text{rank}(\Phi)=4} \quad (3.33)$$

$\alpha_1 = 1, \alpha_2 = 10, \alpha_3 = 100, \alpha_4 = 1000$
 $\beta_1 = 0, \beta_2 = 00, \beta_3 = 000, \beta_4 = 0000$

The matrix Φ_0 is as follows:

$$\Phi_0 = \begin{bmatrix} a_{100} & a_{1000} & a_{10000} & a_{100000} \\ a_{1000} & a_{10000} & a_{100000} & a_{1000000} \\ a_{10000} & a_{100000} & a_{1000000} & a_{10000000} \\ a_{100000} & a_{1000000} & a_{10000000} & a_{100000000} \end{bmatrix} \Rightarrow \quad (3.34)$$

$$\Phi_0 = \begin{bmatrix} 0.2837 & 0.0919 & 0.0361 & 0.0105 \\ 0.0919 & 0.0361 & 0.0105 & 0.0021 \\ 0.0361 & 0.0105 & 0.0021 & 0.0012 \\ 0.0105 & 0.0021 & 0.0012 & 0.0006 \end{bmatrix}$$

and the obtained state-affine model is as follows:

$$\mathbf{F}_0 = \boldsymbol{\Phi}^{-1} \boldsymbol{\Phi}_0 = \begin{bmatrix} 0 & 0 & 0 & -0.0001 \\ 1 & 0 & 0 & 0.0771 \\ 0 & 1 & 0 & -0.1779 \\ 0 & 0 & 1 & 0.1389 \end{bmatrix} \quad (3.35)$$

$$\boldsymbol{\Phi}_1 = \boldsymbol{\Phi}_2 = \boldsymbol{\Phi}_3 = \boldsymbol{\Phi}_4 = \mathbf{0} \Rightarrow \mathbf{F}_1 = \mathbf{F}_2 = \mathbf{F}_3 = \mathbf{F}_4 = \mathbf{0}$$

$$\mathbf{G}_1 = \boldsymbol{\Phi}^{-1} \begin{bmatrix} a_{10} \\ a_{100} \\ a_{1000} \\ a_{10000} \end{bmatrix} = \boldsymbol{\Phi}^{-1} \begin{bmatrix} 0.4566 \\ 0.2837 \\ 0.0919 \\ 0.0361 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{G}_2 = \mathbf{G}_3 = \mathbf{G}_4 = \mathbf{0}$$

$$\mathbf{H}_0 = [a_{10} \quad a_{100} \quad a_{1000} \quad a_{10000}] \\ = [0.4566 \quad 0.2837 \quad 0.0919 \quad 0.0361]$$

3.5.3.2 Behavior matrix of all Volterra kernels

For brevity, part of the Volterra kernels and their corresponding Behavior matrix entries are shown in Table 3.4. The other entries are generated based on the same principles. The resulting Behavior matrix with the rows and columns of zeros removed is as follows:

$$\mathbf{B}(f) = \begin{bmatrix} a_{10} & a_{20} & a_{100} & a_{110} & a_{200} & a_{1000} & a_{1010} & a_{1100} & a_{2000} \\ a_{100} & a_{200} & a_{1000} & a_{1100} & a_{2000} & a_{10000} & a_{10100} & a_{11000} & a_{20000} \\ a_{110} & 0 & a_{1010} & 0 & 0 & a_{10010} & 0 & 0 & 0 \\ a_{1000} & a_{2000} & a_{10000} & a_{11000} & a_{20000} & a_{100000} & a_{101000} & a_{110000} & a_{200000} \\ a_{1010} & 0 & a_{10010} & 0 & 0 & a_{100010} & 0 & 0 & 0 \\ a_{1100} & 0 & a_{10100} & 0 & 0 & a_{100100} & 0 & 0 & 0 \end{bmatrix} \quad (3.36)$$

The Behavior matrix can also be written with the original Volterra kernels as follows:

$$\mathbf{B}(f) = \begin{bmatrix} h_1 & h_{11} & h_2 & h_{12} & h_{22} & h_3 & h_{13} & h_{23} & h_{33} \\ h_2 & h_{22} & h_3 & h_{23} & h_{33} & h_4 & h_{24} & h_{34} & h_{44} \\ h_{12} & 0 & h_{13} & 0 & 0 & h_{14} & 0 & 0 & 0 \\ h_3 & h_{33} & h_4 & h_{34} & h_{44} & h_5 & h_{35} & h_{45} & h_{55} \\ h_{13} & 0 & h_{14} & 0 & 0 & h_{15} & 0 & 0 & 0 \\ h_{23} & 0 & h_{24} & 0 & 0 & h_{25} & 0 & 0 & 0 \end{bmatrix} \quad (3.37)$$

Table 3.4 Volterra kernels and corresponding Behavior matrix entries

Volterra kernels	Values	Behavior matrix entries
h_{11}	0.0782	a_{20}
h_{13}	-0.0367	a_{1010}
h_{23}	-0.0026	a_{1100}
h_{29}	0.0013	$a_{1000000100}$
h_{44}	0.0004	a_{20000}

Based on the same procedures shown in section 3.5.3.1 using the 1st-order Volterra kernels, the state-affine model dimension, i.e., the dimension of the \mathbf{F}_i matrices, was obtained, such that the state-affine model with this dimension will exhibit the minimum sum of squared errors. Simulation results show that dimension 3 is the optimal dimension of the model, and this means that only those Volterra kernels that make up the upper-left 3×3 sub-matrix of the Behavior matrix are used to produce the state-affine model. Since all of the Volterra kernels are contributing to approximate the CSTR system behavior, partial use of the kernels will result in increasing modeling error for the resulting state-affine model.

3.5.3.3 Simulation of the state-affine model

The resulting state-affine model is of dimension 3, with the matrices given by equation (3.39).

$$\begin{aligned} \mathbf{x}(t+1) &= \{\mathbf{F}_0 + \mathbf{F}_1 u(t)\} \mathbf{x}(t) + \{\mathbf{G}_1 + \mathbf{G}_2 u(t)\} u(t) \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t) \end{aligned} \quad (3.38)$$

$$\begin{aligned} \mathbf{F}_0 &= \begin{bmatrix} 0 & -0.1776 & 0.1037 \\ 0 & -0.399 & 0.0597 \\ 1 & 0.0307 & 0.1406 \end{bmatrix} & \mathbf{F}_1 &= \begin{bmatrix} -0.0105 & 0 & -0.1267 \\ -0.1074 & 0 & -0.2776 \\ 0.0018 & 0 & 0.0219 \end{bmatrix} \\ \mathbf{G}_1 &= [1 \ 0 \ 0]^T & \mathbf{G}_2 &= [0 \ 1 \ 0]^T \\ \mathbf{H}_0 &= [0.4566 \ 0.0782 \ 0.2837] \end{aligned} \quad (3.39)$$

Simulation of this state-affine model is shown in Figure 3.5, and compared to the real CSTR process output. The sum of squared errors of the simulation is 2.8066, which is 1% of the sum of squares output. Results were generated using MATABL.

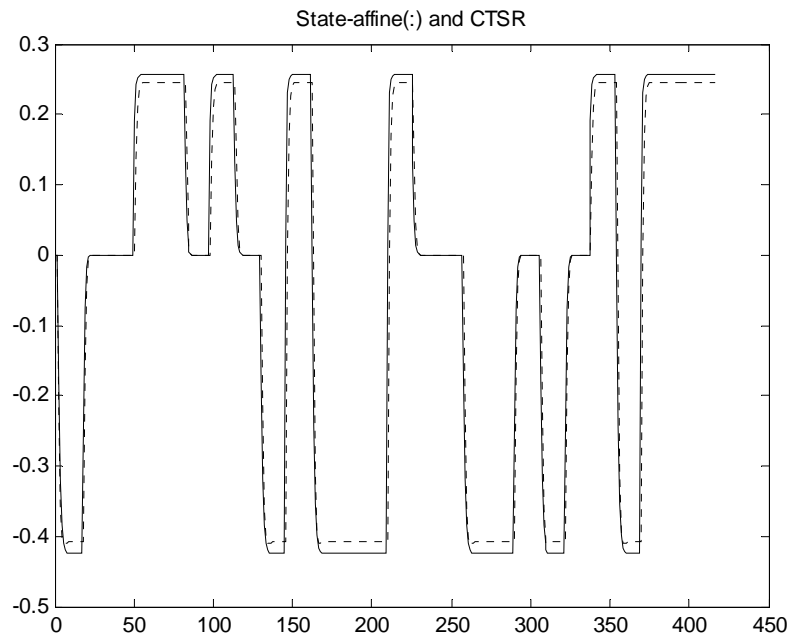


Figure 3.5 CSTR process output (solid line) and state-affine model output (dotted line)

Based on the discussion in section 3.2, the uncertainty of this state-affine model is 1st-order, and shown as follows:

$$\delta_{1,t} = u(t) \quad (3.40)$$

Using the above uncertainty expression, equation (3.38) is rewritten as follows:

$$\begin{aligned} \mathbf{x}(t+1) &= \{\mathbf{F}_0 + \mathbf{F}_1 \delta_{1,t}\} \mathbf{x}(t) + \{\mathbf{G}_1 + \mathbf{G}_2 \delta_{1,t}\} u(t) \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t) \end{aligned} \quad (3.41)$$

This model will be used in later chapters of this work for robustness analysis and design. The uncertain parameter is quantified based on the bounds of the manipulated variable. In section 3.5.2, it has been discussed that the input variable is in the range of $[-1,1]$, and this can also be seen from Figure 3.3. As a result, the uncertain parameter is quantified as follows:

$$u(t) \in [-1 \ 1] \rightarrow \delta_{1,t} \in [-1,1] \quad (3.42)$$

If this model uncertain parameter is the only uncertainty in the system, the two vertices of the parameter box \mathbf{W} are as follows:

$$\mathbf{W} := \{(\omega_1) : \omega_1 \in [-1,1]\} \quad (3.43)$$

4 Robust Stability and Robust Performance Analysis

Control systems are often designed based on a simplified model of the physical plant that often does not take into account complex behaviors such as nonlinear and high order dynamics. The difference between the simplified model and the real process is model uncertainty. A robustness analysis is necessary to validate the design and obtain guarantees of the stability and performance in the face of model uncertainty. In this chapter, two approaches are introduced and compared, with respect to the analysis of robust stability (RS) and robust performance (RP) of the system.

First, RS and RP tests based on quadratic Lyapunov functions and their LMIs formulations will be presented. The theory of LMIs is introduced in detail and three generic LMIs problems are reviewed. RS and RP conditions specific for our problem are formulated based on fixed-parameter Lyapunov functions, and then on parameter-dependent Lyapunov functions. Under the affine parameter-dependence assumption of the parameter-dependent systems introduced in the previous chapter, these conditions are all reduced to a finite set of LMIs, which can be solved using one of the three generic LMIs algorithms. Different approaches have been investigated to reduce the conservatism of the analysis towards more reliable designs. The key novelty of this part of the work is that a set of gain-scheduled PI controller and MPC controller design problems have been formulated based on the robustness conditions proposed in this chapter. These formulations are further explained in details in Chapter 5 and 6.

Second, RS and RP tests based on the extensions of structured singular values (SSV) will be reviewed for nonlinear and/or time-varying uncertainty. SSV analysis investigates the robust stability and performance of systems with linear time-invariant linear-fractional uncertainty. It is also applicable to parameter-dependent systems based on an equivalent linear fractional transformation (LFT) representation. The procedure to partition a system model into an equivalent LFT is required before the application of SSV approach. Gain-scheduled PI controller design problems will be formulated using the RS and RP conditions in Chapter 5.

Finally, the two approaches, i.e., the quadratic Lyapunov approach and the approach based on the extensions of SSV, are compared.

This chapter is organized as follows. In section 4.1, the background knowledge related with the LMIs theory and the LMIs techniques used in this work is summarized for reference. Section 4.2 presents conditions of quadratic stability and performance based on Lyapunov's direct method, and their corresponding LMIs formulation. In section 4.3, to reduce the conservatism of the analysis developed in section 4.2, parameter-dependent Lyapunov functions are introduced, and LMIs-based robust stability and performance tests are formulated. Section 4.4 reviews the SSV analysis and proposes the RS and RP conditions for time-varying uncertainty. In section 4.5, the comparison of the SSV approach with the quadratic Lyapunov approach is presented.

4.1 Linear Matrix Inequalities (LMIs)

4.1.1 LMIs and LMI problems

A linear matrix inequality (LMI) is any expression of the form

$$\mathbf{A}(\mathbf{x}) := \mathbf{A}_0 + x_1 \mathbf{A}_1 + \dots + x_N \mathbf{A}_N < \mathbf{0} \quad (4.1)$$

where

- $\mathbf{x} = [x_1 \ \dots \ x_N]$ is a vector of unknown scalars, also referred to as the decision or the optimization variables.
- $\mathbf{A}_0, \dots, \mathbf{A}_N$ are given symmetric matrices.
- The inequality is negative definite, i.e., the largest eigenvalue of $\mathbf{A}(\mathbf{x})$ is negative, or $\boldsymbol{\eta}^T \mathbf{A}(\mathbf{x}) \boldsymbol{\eta} < \mathbf{0}$ for all nonzero $\boldsymbol{\eta} \in \mathfrak{R}^N$.

Although the form of LMIs given by equation (4.1) may seem to be restrictive, it can represent a wide variety of constraints on \mathbf{x} . It should be noted that the constraints $\mathbf{A}(\mathbf{x}) > \mathbf{0}$ and $\mathbf{A}(\mathbf{x}) < \mathbf{B}(\mathbf{x})$ are special cases of equation (4.1), since they can be rewritten as $-\mathbf{A}(\mathbf{x}) < \mathbf{0}$ and $\mathbf{A}(\mathbf{x}) - \mathbf{B}(\mathbf{x}) < \mathbf{0}$, respectively.

The LMIs in equation (4.1) is a convex constraint on \mathbf{x} since $\mathbf{A}(\mathbf{y}) < \mathbf{0}, \mathbf{A}(\mathbf{z}) < \mathbf{0} \Rightarrow \mathbf{A}(\frac{\mathbf{y} + \mathbf{z}}{2}) < \mathbf{0}$. As a result,

- its solution set, called the feasible set, is a convex subset of \mathfrak{R}^N ;
- finding a solution \mathbf{x} to equation (4.1), if any such solution exists, it is a convex optimization problem.

Convexity has an important consequence: even though equation (4.1) has no analytical solution in general, it can be solved numerically with guarantees of finding a solution when one exists. A system of LMIs constraints can be regarded as a single LMI since

$$\begin{cases} \mathbf{A}_1(\mathbf{x}) < \mathbf{0} \\ \vdots \\ \mathbf{A}_K(\mathbf{x}) < \mathbf{0} \end{cases} \Leftrightarrow \mathbf{A}(\mathbf{x}) := \text{diag}(\mathbf{A}_1(\mathbf{x}), \dots, \mathbf{A}_K(\mathbf{x})) < \mathbf{0}$$

where $\text{diag}(\mathbf{A}_1(\mathbf{x}), \dots, \mathbf{A}_K(\mathbf{x}))$ denotes a block-diagonal matrix with $\mathbf{A}_1(\mathbf{x}), \dots, \mathbf{A}_K(\mathbf{x})$ on its diagonal. Hence multiple LMIs constraints can be imposed on the vector of decision variables \mathbf{x} while preserving convexity.

In most control applications, LMIs do not naturally arise in the canonical form given by equation (4.1), but rather in the following form:

$$\mathbf{L}(\mathbf{X}_1, \dots, \mathbf{X}_n) < \mathbf{R}(\mathbf{X}_1, \dots, \mathbf{X}_n)$$

where $\mathbf{L}(\cdot), \mathbf{R}(\cdot)$ are affine functions of some structured matrix variables $\mathbf{X}_1, \dots, \mathbf{X}_n$. A simple example is the Lyapunov inequality:

$$\mathbf{A}^T \mathbf{X} \mathbf{A} - \mathbf{X} < \mathbf{0}$$

where the unknown \mathbf{X} is a symmetric matrix. Defining $x_1 \dots x_N$ as the independent scalar entries of \mathbf{X} , these LMIs could be rewritten in the form of equation (4.1). Expressing LMIs in a condensed form as follows: $\mathbf{A}(\mathbf{x}) < \mathbf{0}$, in addition to saving notation, may lead to more convenient and efficient computation. This natural form $\mathbf{A}(\mathbf{x}) < \mathbf{0}$ is the approach taken in this work.

The three generic problems that can be formulated in terms of LMIs are as follows:

1. Feasibility problem (FEASP in MATLAB). Finding a solution to the LMIs system

$$\mathbf{A}(\mathbf{x}) < \mathbf{0} \tag{4.2}$$

is called the feasibility problem.

2. The eigenvalue problem. The eigenvalue problem is to minimize the maximum eigenvalue of a matrix that depends affinely on a variable, subject to an LMIs constraint, i.e.,

$$\text{Minimize } \lambda \text{ subject to } \lambda \mathbf{I} - \mathbf{A}(\mathbf{x}) < \mathbf{0}, \mathbf{B}(\mathbf{x}) > \mathbf{0} \tag{4.3}$$

where \mathbf{A}, \mathbf{B} are symmetric matrices that depend affinely on the optimization variable \mathbf{x} . This is also a convex optimization problem. Eigenvalue problems can appear in the equivalent form of minimizing a linear function (MINCX in MATLAB) subject to an LMIs, as follows:

$$\text{Minimize } \mathbf{c}^T \mathbf{x} \text{ subject to } \mathbf{A}(\mathbf{x}) < \mathbf{0} \quad (4.4)$$

3. Generalized eigenvalue minimization problem (GEVP in MATLAB). The GEVP is to minimize the maximum generalized eigenvalue of a pair of matrices that depend affinely on a variable, subject to a LMIs constraint. The general form of a GEVP is:

$$\text{Minimize } \lambda \text{ subject to } \lambda \mathbf{B}(\mathbf{x}) - \mathbf{A}(\mathbf{x}) > \mathbf{0}, \mathbf{B}(\mathbf{x}) > \mathbf{0}, \mathbf{C}(\mathbf{x}) > \mathbf{0} \quad (4.5)$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are symmetric matrices that are affine functions of the optimization variable \mathbf{x} . This problem can also be expressed as follows:

$$\text{Minimize } \lambda_{\max}(\mathbf{A}(\mathbf{x}), \mathbf{B}(\mathbf{x})) \text{ subject to } \mathbf{B}(\mathbf{x}) > \mathbf{0}, \mathbf{C}(\mathbf{x}) > \mathbf{0} \quad (4.6)$$

where $\lambda_{\max}(\mathbf{A}(\mathbf{x}), \mathbf{B}(\mathbf{x}))$ denotes the largest generalized eigenvalue of $\lambda \mathbf{B}(\mathbf{x}) - \mathbf{A}(\mathbf{x}) > \mathbf{0}$, with $\mathbf{B}(\mathbf{x}) > \mathbf{0}$.

4.1.2 Well-posedness issues

The LMIs solvers included in the MATLAB LMIs Toolbox to solve the three generic LMIs problems listed above are based on the interior-point optimization techniques. To compute a feasible solution for these problems, such techniques require that the system of LMIs constraints be strictly feasible, i.e., that the feasible set have a nonempty interior (Boyd and et al., 1994). As a result, the solvers may encounter difficulty when the LMIs constraints are feasible but not strictly feasible. That is, feasible solutions exist for the LMIs of the following form:

$$\text{Find } x \text{ such that } \mathbf{L}(\mathbf{x}) \leq \mathbf{0} \quad (4.7)$$

while no feasible solutions exist for the following strict LMIs:

$$\text{Find } x \text{ such that } \mathbf{L}(\mathbf{x}) < \mathbf{0} \quad (4.8)$$

According to MATLAB LMIs Toolbox manual (Gahinet, Nemirovski and et al., 1995), for feasibility problems, this difficulty is automatically circumvented by the Toolbox function FEASP by reformulating the problem given by equation (4.7) as follows:

$$\text{Minimize } t \text{ subject to } \mathbf{L}(\mathbf{x}) < t \times \mathbf{I} \quad (4.9)$$

In this modified problem, the LMIs constraint is always strictly feasible in x, t and the original LMIs given by equation (4.7) is feasible if and only if the global minimum t_{\min} of equation (4.9) satisfies

$$t_{\min} \leq 0 \quad (4.10)$$

For feasible but not strictly feasible problems, however, the computational effort is typically higher as the FEASP function strives to approach the global optimum corresponding to $t_{\min} = 0$ to a high accuracy.

For the LMIs problems addressed by the LMIs Toolbox functions MINCX and GEVP, non-strict feasibility generally causes the solvers to fail and to return an “infeasibility” diagnosis. Although there is no universal remedy for this difficulty, it is sometimes possible to eliminate underlying algebraic constraints to obtain a strictly feasible problem with fewer variables. Boyd and et al. (1994) have given an algorithm of reducing a set of feasible non-strict LMIs to a set of strictly feasible LMIs.

4.1.3 Semi-definite $\mathbf{B}(\mathbf{x})$ in GEVP problems

Consider the generalized eigenvalue minimization problem

$$\text{Minimize } \lambda \text{ subject to } \lambda \mathbf{B}(\mathbf{x}) - \mathbf{A}(\mathbf{x}) > \mathbf{0}, \mathbf{B}(\mathbf{x}) > \mathbf{0}, \mathbf{C}(\mathbf{x}) > \mathbf{0} \quad (4.5)$$

Technically, the positivity of $\mathbf{B}(\mathbf{x})$ for some $\mathbf{x} \in \mathfrak{R}^N$ is required for the well-posedness of the LMIs problem and the applicability of the interior-point methods. Hence, problems with the following $\mathbf{B}(\mathbf{x})$:

$$\mathbf{B}(\mathbf{x}) = \begin{bmatrix} \mathbf{B}_1(\mathbf{x}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \text{ with } \mathbf{B}_1(\mathbf{x}) \text{ strictly feasible} \quad (4.11)$$

can not be directly solved with the GEVP function in MATLAB because of the additional zero eigenvalues in $\mathbf{B}(\mathbf{x})$. A simple remedy consists of replacing the following constraints given by equation (4.12) by the ones given by equation (4.13):

$$\lambda \mathbf{B}(\mathbf{x}) - \mathbf{A}(\mathbf{x}) > \mathbf{0}, \mathbf{B}(\mathbf{x}) > \mathbf{0}, \mathbf{C}(\mathbf{x}) > \mathbf{0} \quad (4.12)$$

$$\mathbf{A}(\mathbf{x}) < \begin{bmatrix} \mathbf{Y} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{Y} < \lambda \mathbf{B}_1(\mathbf{x}), \mathbf{B}_1(\mathbf{x}) > \mathbf{0} \quad (4.13)$$

where \mathbf{Y} is an additional symmetric variable of proper dimension. The resulting problem is now equivalent to equation (4.5) and can be solved directly with the GEVP function in MATLAB.

4.2 Quadratic Lyapunov Functions

Lyapunov stability theory is based on two methods: the linearization method and the direct method, which are briefly introduced in the following.

The Lyapunov's linearization method is concerned with the local stability of a nonlinear system. It is a formalization of the intuitive argument that a nonlinear system should behave similarly to its linearized approximation within a small neighbourhood of an equilibrium point. Because all physical systems are inherently nonlinear, the Lyapunov's

linearization method serves as the fundamental justification for using linear control techniques for the local analysis of nonlinear processes. The Lyapunov's linearization theorem states the following (Slotine and Li, 1991)

- If linearized the system is strictly stable, i.e., if all the eigenvalues of the closed-loop matrix are strictly inside the unit circle, then the equilibrium point is asymptotically stable for the actual nonlinear system.
- If linearized the system is unstable, i.e., if at least one eigenvalue of the closed-loop matrix is outside of the unit circle, then the equilibrium point is unstable for the actual nonlinear system.
- If linearized the system is marginally stable, i.e., if all the eigenvalues of the closed-loop matrix are inside the unit circle, but at least one of them is on the unit circle, then a conclusion regarding stability for the actual nonlinear system can not be established from the linear approximation. Thus, the equilibrium point may be stable, asymptotically stable or unstable for the actual nonlinear system.

The Lyapunov's linearization theorem shows that the linear control design is a matter of consistency, i.e., the control system must be designed such as the system output and input remain within a small neighborhood of the nominal operating point, justifying the linear approximation. It also raises major questions regarding the limitations of linear design, i.e., how large are the linear ranges? what is the extent of the stability range? These questions motivate a more fundamental approach to the nonlinear control problem, the Lyapunov's direct method.

The Lyapunov's direct method is the mathematical extension of the energy conservation concepts associated with a mechanical system: the motion of a mechanical system is stable if its total mechanical energy decreases all the time. The basic procedure of this direct method is to construct a scalar energy-like function, referred to as the Lyapunov

function, for the dynamic system, and to examine the time-variation of this scalar function as time progresses.

However, there is no systematic way of finding Lyapunov functions for nonlinear systems that will result in the least conservative designs. This is a fundamental drawback of the direct method. Slotine and Li (1991) have discussed a number of techniques which can facilitate the search for appropriate Lyapunov functions. They showed that Lyapunov functions can be systematically found to describe stable linear systems. Given a linear time-invariant system of the form $\boldsymbol{\eta}(t+1) = \mathbf{A}\boldsymbol{\eta}(t)$, a quadratic Lyapunov function is defined to have the following form:

$$V(t) = \boldsymbol{\eta}(t)^T \mathbf{P}\boldsymbol{\eta}(t) \quad (4.14)$$

This has been proposed for the stability analysis, where \mathbf{P} is a symmetric positive-definite matrix, usually called Lyapunov matrix. The Lyapunov's direct method to assess the global stability of a system states the following (Slotine and Li, 1991):

- *Assume that there exists a scalar function V of the state $\boldsymbol{\eta}$, with continuous first-order derivatives such that*

- $V(t) = \boldsymbol{\eta}(t)^T \mathbf{P}\boldsymbol{\eta}(t)$ is positive-definite;
- $V(t+1) - V(t) < 0$
- $V(t) \rightarrow \infty$ as $\|\boldsymbol{\eta}(t)\| \rightarrow \infty$

Then the equilibrium at the origin is globally asymptotically stable.

Stability analysis based on the quadratic Lyapunov function given above is usually referred to as quadratic Lyapunov stability. This quadratic Lyapunov function and quadratic stability analysis are fundamental to the present work, and they are applied to nonlinear time-varying systems with proper modifications.

In this work, the following system is considered

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\eta}(t+1) \\ e(t) \end{bmatrix} &= \begin{bmatrix} \mathbf{A}(\boldsymbol{\delta}_t) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ v(t) \end{bmatrix} \\ \boldsymbol{\eta}(0) &= \boldsymbol{\eta}_0 \end{aligned} \quad (4.15)$$

where $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \dots, \delta_n) \in \mathbf{R}^n$ is a vector of uncertain and time-varying real parameters.

Throughout this work, the following assumptions are made:

1. Each parameter $\delta_{i,t}$ is real and ranges between known extreme values $\underline{\delta}_i$ and $\overline{\delta}_i$ as follows:

$$\delta_{i,t} \in [\underline{\delta}_i, \overline{\delta}_i] \quad (4.16)$$

2. The state matrix $\mathbf{A}(\boldsymbol{\delta}_t)$ depends affinely on the parameters as follows:

$$\mathbf{A}(\boldsymbol{\delta}_t) = \mathbf{A}_0 + \mathbf{A}_1 \delta_{1,t} + \mathbf{A}_2 \delta_{2,t} \dots + \mathbf{A}_n \delta_{n,t} \quad (4.17)$$

where $\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_n$ are known fixed matrices. This dependence is referred to as affine parametric dependence.

The first assumption means that the parameter vector $\boldsymbol{\delta}_t$ is valued in a hyper-rectangle called the parameter box. In the sequel, \mathbf{W} denotes the 2^n vertices or corners of this parameter box as follows:

$$\mathbf{W} := \{(\omega_1, \omega_2, \dots, \omega_n) : \omega_i \in \{\underline{\delta}_i, \overline{\delta}_i\}\} \quad (4.18)$$

The second assumption above is introduced for technical and simplicity reasons. Several extensions of this approach to more complex parameter dependences are also possible. Though somewhat restrictive, the state matrix form given by equation (4.17) still covers a wide variety of relevant problems.

4.2.1 Quadratic Lyapunov stability (QLS)

Given a control system, the most important question is whether it is stable. Every control system, whether linear or nonlinear, involves a stability problem which should be addressed. The approach in this work is built upon quadratic Lyapunov stability, and the details are reviewed now.

Definition 4.1 (Quadratic Lyapunov Stability, QLS, Gahinet and et al., 1994) *For systems defined by (4.15), a sufficient condition for asymptotic stability is the existence of a positive-definite quadratic Lyapunov function $V(t) = \boldsymbol{\eta}(t)^T \mathbf{P}\boldsymbol{\eta}(t)$, $V(t) > 0$, $\mathbf{P} > \mathbf{0}$, $\mathbf{P} = \mathbf{P}^T$ such that*

$$V(t+1) - V(t) < 0 \quad (4.19)$$

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

It should be noted that $V(t+1) - V(t) < 0 \Leftrightarrow \mathbf{x}^T (\mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P} \mathbf{A}(\boldsymbol{\delta}_t) - \mathbf{P}) \mathbf{x} < \mathbf{0}$, so the condition given by equation (4.19) is equivalent to equation (4.20) for all admissible values and trajectories of the parameter vector $\boldsymbol{\delta}_t$.

$$\mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P} \mathbf{A}(\boldsymbol{\delta}_t) - \mathbf{P} < \mathbf{0} \quad (4.20)$$

Assessing quadratic stability is not tractable in general since equation (4.20) places an infinite number of constraints on \mathbf{P} . However, equation (4.20) can be reduced to a finite set of LMIs constraints for the following two cases,

1. $\mathbf{A}(\boldsymbol{\delta}_t)$ ranges in a fixed polytope of matrices as follows:

$$\begin{aligned} \mathbf{A}(\boldsymbol{\delta}_t) &= q_1 \mathbf{A}_1 + \dots + q_K \mathbf{A}_K \\ q_i &> 0 \quad \sum_i q_i = 1 \end{aligned} \tag{4.21}$$

This is referred to as a polytopic model.

2. $\mathbf{A}(\boldsymbol{\delta}_t)$ is a fixed affine function of some uncertainty time-varying parameters $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \dots, \delta_n) \in \mathbf{R}^n$ as follows:

$$\mathbf{A}(\boldsymbol{\delta}_t) = \mathbf{A}_0 + \mathbf{A}_1 \delta_{1,t} + \mathbf{A}_2 \delta_{2,t} \dots + \mathbf{A}_n \delta_{n,t} \tag{4.17}$$

This is referred to as an affine parameter-dependent model.

The first case corresponds to time-varying systems modeled by an envelope of linear time-invariant systems, and the second case corresponds to systems whose state-space equations depend affinely on time-varying parameters, i.e., parameter-dependent systems. The details of these systems have been summarized in section 3.2.1. Budman and Knapp (2001) obtained a finite set of LMIs for the first case. In this work, the specific problem under study can be formulated in the form of the second case, and as a result, the conditions of QLS can be reduced to a finite set of LMIs. This result is summarized in the following theorem.

Theorem 4.1 *Let $\boldsymbol{\delta}_t = (\delta_{1,t}, \delta_{2,t}, \dots, \delta_{n,t}) \in \mathbf{R}^n$ be a vector of time-varying uncertain real parameters varying in the hyper-rectangle defined by (4.18) and let \mathbf{W} denote the set of*

vertices of this hyper-rectangle. Consider the time-varying system (4.15) where $\mathbf{A}(\delta_t)$ depends affinely on δ_t according to equation (4.17).

The system (4.15) satisfies QLS if there exists $\mathbf{P} > \mathbf{0}, \mathbf{P} = \mathbf{P}^T$ such that

$$\mathbf{A}(\omega)^T \mathbf{P} \mathbf{A}(\omega) - \mathbf{P} < \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (4.22)$$

In other words, it suffices that \mathbf{P} be positive-definite and satisfy the LMIs at each corner ω_i of the parameter box. This reformulation has the merit of reducing the problem with infinitely many constraints to a finite set of matrix inequalities. The resulting LMIs given by equation (4.22) are then solved numerically with existing LMIs software, e.g., MATLAB LMIs Toolbox. This test can be extended to quadratic Lyapunov H_∞ performance assessment as explained in the following section (Gao and Budman, 2003).

4.2.2 Quadratic Lyapunov H_∞ performance (QLP)

A way to measure performance is required before a controller which achieves nominal or robust performance can be designed. In robust control theory, the l_2 -norm, which is related to the energy of the signal, is usually used. For vector signals $\mathbf{e}(t)$, this norm is defined to be:

$$\|\mathbf{e}\|_{l_2} = \sum_{t=0}^{\infty} \mathbf{e}(t)^T \mathbf{e}(t) \quad (4.23)$$

For simplicity, it is usually written as $\|\mathbf{e}\|$. The operator norm induced by the l_2 -norm is the H_∞ -norm, and it is defined as follows:

$$\sup_{\|\mathbf{e}\|_{l_2} \neq 0} \frac{\|\mathbf{G}\mathbf{e}\|_{l_2}}{\|\mathbf{e}\|_{l_2}} = \sup_{\omega} \bar{\sigma}[\mathbf{G}(j\omega)] = \|\mathbf{G}\|_{\infty} \quad (4.24)$$

where \mathbf{G} is a proper stable transfer-function.

Definition 4.2 (Quadratic Lyapunov H_{∞} Performance, QLP, Gahinet and et al., 1994)

The system (4.15) with zero initial state satisfies QLS and

$$\|\mathbf{e}\|_{l_2} < \gamma \|\mathbf{v}\|_{l_2} \quad (4.25)$$

for all l_2 -bounded input v if there exists $\mathbf{P} > \mathbf{0}$, $\mathbf{P} = \mathbf{P}^T$ and a positive-definite quadratic Lyapunov function $V(t) = \boldsymbol{\eta}(t)^T \mathbf{P} \boldsymbol{\eta}(t)$, $V(t) > 0$, such that

$$V(t+1) - V(t) + \mathbf{e}^T(t)\mathbf{e}(t) - \gamma^2 \mathbf{v}^T(t)\mathbf{v}(t) < 0 \quad (4.26)$$

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

For zero initial states, equation (4.25) follows from the summation of equation (4.26) over an infinite period of time. Inequality (4.26) is true *iff* equation (4.27) holds for all admissible values and trajectories of the parameter vector $\boldsymbol{\delta}_t$ according to the following Theorem 4.2.

$$\begin{bmatrix} \mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P} \mathbf{A}(\boldsymbol{\delta}_t) - \mathbf{P} & \mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\boldsymbol{\delta}_t) & \mathbf{B}^T \mathbf{P} \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0} \quad (4.27)$$

Assessing quadratic Lyapunov H_{∞} performance is not tractable in general since (4.27) places an infinite number of constraints on \mathbf{P} . Under the affine dependence assumption

give by equation (4.17), Gao and Budman (2003) have proposed the following theorem to show that equation (4.27) holds iff \mathbf{P} satisfies a specific system of LMIs.

Theorem 4.2 Consider the stable time-varying system (4.15) where δ_t , $\mathbf{A}(\delta_t)$ and \mathbf{W} are defined the same as in Theorem 4.1. A sufficient condition for QLP of this system is the existence of $\mathbf{P} > \mathbf{0}, \mathbf{P} = \mathbf{P}^T$ such that

$$\begin{bmatrix} \mathbf{A}(\omega)^T \mathbf{P} \mathbf{A}(\omega) - \mathbf{P} & \mathbf{A}(\omega)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\omega) & \mathbf{B}^T \mathbf{P} \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (4.28)$$

Proof of Theorem 4.2: For a stable system with zero initial states, the summation of equation (4.26) over an infinite period of time gives the following:

$$\begin{aligned} \sum_{t=0}^{\infty} \{V(t+1) - V(t) + e^T(t)e(t) - \gamma^2 v^T(t)v(t)\} < 0 &\longrightarrow (4.29) \\ V(\infty) - V(0) + \sum_{t=0}^{\infty} \{e^T(t)e(t) - \gamma^2 v^T(t)v(t)\} < 0 &\xrightarrow{V(\infty)=0, V(0)=0} \\ \sum_{t=0}^{\infty} \{e^T(t)e(t) - \gamma^2 v^T(t)v(t)\} < 0 &\longrightarrow \|e\|_{l_2} < \gamma \|v\|_{l_2} \end{aligned}$$

This explains Definition 4.2. To prove Theorem 4.2, consider the time-varying system given by equation (4.15), where δ_t , $\mathbf{A}(\delta_t)$, \mathbf{W} are defined the same as in Theorem 4.2. Inequality (4.26) can be expanded using equation (4.15) and the definition of the quadratic Lyapunov function given by equation (4.14), as follows:

$$\begin{aligned} &V(t+1) - V(t) + e^T(t)e(t) - \gamma^2 v^T(t)v(t) \quad (4.30) \\ &= \boldsymbol{\eta}(t+1)^T \mathbf{P} \boldsymbol{\eta}(t+1) - \boldsymbol{\eta}(t)^T \mathbf{P} \boldsymbol{\eta}(t) + (\mathbf{C} \boldsymbol{\eta}(t) + \mathbf{D} v(t))^T (\mathbf{C} \boldsymbol{\eta}(t) + \mathbf{D} v(t)) - \gamma^2 v^T(t)v(t) \\ &= (\mathbf{A}(\delta_t) \boldsymbol{\eta}(t) + \mathbf{B} v(t))^T \mathbf{P} (\mathbf{A}(\delta_t) \boldsymbol{\eta}(t) + \mathbf{B} v(t)) - \boldsymbol{\eta}(t)^T \mathbf{P} \boldsymbol{\eta}(t) + \\ &(\mathbf{C} \boldsymbol{\eta}(t) + \mathbf{D} v(t))^T (\mathbf{C} \boldsymbol{\eta}(t) + \mathbf{D} v(t)) - \gamma^2 v^T(t)v(t) < 0 \end{aligned}$$

Which can be rewritten in a matrix form as follows:

$$\lambda^T \begin{bmatrix} \mathbf{A}(\delta_t)^T \mathbf{P} \mathbf{A}(\delta_t) - \mathbf{P} & \mathbf{A}(\delta_t)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\delta_t) & \mathbf{B}^T \mathbf{P} \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} \lambda < 0 \quad (4.31)$$

where $\lambda^T = [\boldsymbol{\eta}^T(t) \quad \boldsymbol{\nu}^T(t) \quad (\mathbf{C}\boldsymbol{\eta}(t) + \mathbf{D}\boldsymbol{\nu}(t))^T]$

The function in inequality (4.31) is quadratic with respect to all the uncertain parameters δ_t 's, which under the assumption given by equation (4.16), are all valued in a convex parameter box. Also, the coefficient of the quadratic terms can be easily shown to be positive. Thus, if the inequality (4.31) is proven at the vertices of the parameter box, it will also be satisfied for any uncertain parameter combination within the box. As a result, and also under the parameter-affine dependence assumption by equation (4.17), equations (4.30) and (4.31) hold *iff*:

$$\lambda^T \begin{bmatrix} \mathbf{A}(\omega)^T \mathbf{P} \mathbf{A}(\omega) - \mathbf{P} & \mathbf{A}(\omega)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\omega) & \mathbf{B}^T \mathbf{P} \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} \lambda < 0, \text{ for all } \omega \in \mathbf{W} \quad (4.32)$$

$\lambda = [\boldsymbol{\eta}^T(t) \quad \boldsymbol{\nu}^T(t) \quad (\mathbf{C}\boldsymbol{\eta}(t) + \mathbf{D}\boldsymbol{\nu}(t))^T]^T, \lambda \neq \mathbf{0}$

This is true *iff* the symmetric positive-definite matrix \mathbf{P} satisfies the system of LMIs:

$$\begin{bmatrix} \mathbf{A}(\omega)^T \mathbf{P} \mathbf{A}(\omega) - \mathbf{P} & \mathbf{A}(\omega)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\omega) & \mathbf{B}^T \mathbf{P} \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (4.28)$$

Thus, Theorem 4.2 is proved.

Inequality (4.28) can be solved as a feasibility problem (FEASP) for a pre-specified γ , or as a generalized eigenvalue problem (GEVP), to minimize the performance index γ . The

minimization of γ guarantees that $\|\mathbf{v}\|_{l_2}$ will have the least possible effect on $\|\mathbf{e}\|_{l_2}$. It is clear that equation (4.28) falls into the standard form of a GEVP (Boyd and et al., 1994) problem if it is rewritten in the following alternative form:

$$\begin{aligned} \min_{\mathbf{P}} \gamma^2 & \tag{4.33} \\ \text{subject to} & \begin{bmatrix} \mathbf{A}(\omega)^T \mathbf{P} \mathbf{A}(\omega) - \mathbf{P} & \mathbf{A}(\omega)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\omega) & \mathbf{B}^T \mathbf{P} \mathbf{B} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \gamma^2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \text{for all } \omega \in \mathbf{W} & \end{aligned}$$

4.3 Parameter-dependent Lyapunov Functions

Quadratic stability guarantees stability against arbitrarily fast parameter variations. As a result, the QLS and QLP conditions based on quadratic Lyapunov functions given in the previous section can be unnecessarily conservative for constant or slowly-varying parameters. To reduce conservatism in such cases, the notion of parameter-dependent Lyapunov functions, proposed by Gahinet, Apkarian and Chilali (1994) for continuous systems, is further developed in this work for discrete systems. The quadratic parameter-dependent Lyapunov function is given as follows:

$$V(t) = \boldsymbol{\eta}(t)^T \mathbf{P}(\boldsymbol{\delta}_t) \boldsymbol{\eta}(t) \tag{4.34}$$

where the Lyapunov weighting matrix $\mathbf{P}(\boldsymbol{\delta}_t)$ is no longer constant, but it is now a function of $\boldsymbol{\delta}_t$ in the following form:

$$\mathbf{P}(\boldsymbol{\delta}_t) = \mathbf{P}_0 + \mathbf{P}_1 \delta_{1,t} + \mathbf{P}_2 \delta_{2,t} + \cdots + \mathbf{P}_n \delta_{n,t} \tag{4.35}$$

where $\delta_{i,t}$'s are parameters relevant to the system under study. Such affine parameter-dependent Lyapunov functions are central to our approach. In the present work, it will be

shown that by imposing additional constraints on the parameter-dependent Lyapunov functions, the calculation of the parameter-dependent Lyapunov matrix of the form can be formulated into a LMIs problem for the unknown matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$. The resulting test is therefore numerically tractable while always less conservative than quadratic tests based on fixed Lyapunov functions because of the additional variables available for optimization. Note that the usual quadratic Lyapunov stability corresponds to the special case where $\mathbf{P}_1 = \mathbf{P}_2 = \dots = \mathbf{P}_n = \mathbf{0}$. The use of the function (4.34) suggests a natural extension of quadratic stability and performance, described by the definitions shown below.

Definition 4.3 (Affine Quadratic Lyapunov Stability, AQLS, Gahinet and et al., 1994) *For systems defined by (4.15), a sufficient condition for asymptotic stability is the existence of $n+1$ symmetric matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$ such that*

$$\mathbf{P}(\boldsymbol{\delta}_t) = \mathbf{P}_0 + \mathbf{P}_1 \delta_{1,t} + \mathbf{P}_2 \delta_{2,t} + \dots + \mathbf{P}_n \delta_{n,t} > \mathbf{0} \quad (4.36)$$

$$\mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P}(\boldsymbol{\delta}_{t+1}) \mathbf{A}(\boldsymbol{\delta}_t) - \mathbf{P}(\boldsymbol{\delta}_t) < \mathbf{0} \quad (4.37)$$

for all admissible values and trajectories of the uncertainties $\boldsymbol{\delta}_t$ and for all initial conditions $\boldsymbol{\eta}_0$.

Definition 4.4 (Affine Quadratic Lyapunov H_∞ Performance, AQLP, Gahinet and et al., 1994) *The system (4.15) with zero initial state satisfies AQLP if there exist $n+1$ symmetric matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$ such that*

$$\mathbf{P}(\boldsymbol{\delta}_t) = \mathbf{P}_0 + \mathbf{P}_1 \delta_{1,t} + \mathbf{P}_2 \delta_{2,t} + \dots + \mathbf{P}_n \delta_{n,t} > \mathbf{0} \quad (4.36)$$

$$\begin{bmatrix} \mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P}(\boldsymbol{\delta}_{t+1}) \mathbf{A}(\boldsymbol{\delta}_t) - \mathbf{P}(\boldsymbol{\delta}_{t+1}) & \mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P}(\boldsymbol{\delta}_{t+1}) \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P}(\boldsymbol{\delta}_{t+1}) \mathbf{A}(\boldsymbol{\delta}_t) & \mathbf{B}^T \mathbf{P}(\boldsymbol{\delta}_{t+1}) \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0} \quad (4.38)$$

for all admissible values and trajectories of the uncertainties $\boldsymbol{\delta}_t$ and for zero initial conditions $\boldsymbol{\eta}_0$.

From these definitions, affine quadratic stability and quadratic Lyapunov H_∞ performance amount to finding the $n+1$ symmetric matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$ that satisfy equations (4.37) and (4.38). In the next section, this task will be discussed first in the general case of time-varying uncertain parameter, and then for the simpler special case of constant uncertain parameters.

It should be noted that even when $\mathbf{A}(\boldsymbol{\delta}_t), \mathbf{P}(\boldsymbol{\delta}_t)$ are affine in $\boldsymbol{\delta}_t$, it is no longer sufficient to check equations (4.36) and (4.37) for AQLS or equations (4.36) and (4.38) for AQLP, at the corners of the parameter box. The conditions (4.37) and (4.38) are no longer quadratic with respect to the uncertain parameters $\delta_{i,t}$'s, because the term $\mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P}(\boldsymbol{\delta}_{t+1}) \mathbf{A}(\boldsymbol{\delta}_t)$ leads to 3rd-order terms of $\delta_{i,t}$'s. Consequently, checking the conditions at the vertices of the parameter box will not guarantee that the conditions are satisfied as well inside the box. However, convexity can be guaranteed by imposing a convexity requirement (Budman and Knapp 2000; Gahinet, Apkarian and Chilali, 1994), which relies on the concept of convexity along each direction $\delta_{i,t}$ of the parameter space. To recover convexity, an additional constraint must be introduced on $\mathbf{P}(\boldsymbol{\delta}_t)$. Obviously, this constraint restricts the choice of affine Lyapunov matrix $\mathbf{P}(\boldsymbol{\delta}_t)$, and therefore may lead to conservatism. However, the use of the rate of variation in parameters will be helpful to compensate for the increased conservatism. This convexity condition is detailed in the following section.

4.3.1 Time-varying uncertain parameters

In this section, the case of time-varying parameters δ_t with a bounded rate of variation is considered. To handle the time-varying case with less conservatism when the knowledge of parameter variation is available, the following set of rate of changes $\Delta\delta_t = (\Delta\delta_{1,t}, \Delta\delta_{2,t}, \dots, \Delta\delta_{n,t}) \in \mathbf{R}^n$ are considered as additional time-varying uncertain parameters in the design. As shown below, this more general case can be handled by extensions of Theorem 4.1 and Theorem 4.2 and the resulting LMIs conditions become less conservative than the previous quadratic Lyapunov tests. Throughout the section, the following two assumptions are made:

1. The rate of variation $\Delta\delta_t$ is well defined at all time-intervals;
2. $\Delta\delta_t$ satisfies

$$\Delta\delta_{i,t} = \delta_{i,t+1} - \delta_{i,t}, \Delta\delta_{i,t} \in [\underline{v}_i \quad \overline{v}_i] \quad (4.39)$$

where $\underline{v}_i, \overline{v}_i$ are *a priori* known lower and upper bounds on this rate of variation. Practically, these bounds are set during the process design stage due to the process limitations and operation specifications.

As a whole, the vector $\Delta\delta_t$ evolves in a n -dimensional hyper-rectangle whose vertices are given in the set:

$$\mathbf{S} := \{(\tau_1, \tau_2, \dots, \tau_n) : \tau_i \in \{\underline{v}_i, \overline{v}_i\}\} \quad (4.40)$$

For $\mathbf{P}(\delta_t)$ of the form given by equation (4.36):

$$\mathbf{P}(\delta_{t+1}) = \mathbf{P}(\delta_t) + \mathbf{P}(\Delta\delta_t) \quad (4.41)$$

The results of Theorem 4.1 and Theorem 4.2 can then be generalized to the parameter-dependent cases as follows.

3.5.3.4 Affine quadratic Lyapunov stability (AQLS)

Theorem 4.3 Consider the time-varying system (4.15) where δ_t , $\mathbf{A}(\delta_t)$ and \mathbf{W} are defined the same as in Theorem 4.1, $\Delta\delta_t$ and \mathbf{S} are defined according to equations (4.39) and (4.40). A sufficient condition for AQLS of this system is the existence of $n+1$ symmetric matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$ such that

$$\mathbf{P}(\omega) > \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (4.42)$$

$$\begin{aligned} \mathbf{A}(\omega)^T (\mathbf{P}(\omega) + \mathbf{P}(\tau)) \mathbf{A}(\omega) - \mathbf{P}(\omega) < \mathbf{0}, \\ \text{for all } (\omega, \tau) \in \mathbf{W} \times \mathbf{S} \end{aligned} \quad (4.43)$$

$$\begin{aligned} (3\omega + \tau)(\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i) + \sum_{\substack{j=0 \\ (j \neq i)}}^n \omega (\mathbf{A}_j^T \mathbf{P}_i \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_j + \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i) + \\ \sum_{\substack{j=0 \\ (j \neq i)}}^n \tau \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i \geq \mathbf{0}, \text{ for } i = 1, 2, \dots, n, \text{ for all } (\omega, \tau) \in \mathbf{W} \times \mathbf{S} \end{aligned} \quad (4.44)$$

Proof of Theorem 4.3: First note that the positivity constraint given by equation (4.36) is affine in δ_t . Consequently, equation (4.36) holds for all δ_t in the parameter box if it holds at all corners, which is exactly the condition given by equation (4.42). Hence, the only difficulty is to enforce equation (4.37) over the entire parameter box.

Substitute the expressions given by equations (4.17) and (4.35) into the following condition:

$$\mathbf{L}(\delta_t) = \mathbf{A}(\delta_t)^T \mathbf{P}(\delta_{t+1}) \mathbf{A}(\delta_t) - \mathbf{P}(\delta_t) < \mathbf{0} \quad (4.45)$$

For any nonzero vector λ , clearly $f(\delta_t) = \lambda^T \mathbf{L}(\delta_t) \lambda$ is a scalar function of the following form:

$$f(\delta_t) = \lambda^T \mathbf{L}(\delta_t) \lambda \quad (4.46)$$

$$f(\delta_t) = f(\delta_{i,t}, \delta_{i,t} \delta_{j,t}, \delta_{i,t}^2, \delta_{i,t}^2 \delta_{j,t}, \delta_{i,t}^3)$$

In general, the negative sign of $f(\delta_t)$ values at all corners of $\mathbf{W} \times \mathbf{S}$ does not guarantee its negativity over the entire parameter box. However, negativity is obtained when $f(\delta_t)$ is convex in the $\delta_{i,t}$'s. For a function with 3rd-order dependence with respect to $\delta_{i,t}$'s, this is true when $\frac{\partial^2 f(\delta_t)}{\partial \delta_{i,t}^2} \geq 0, i = 1, \dots, n$ for all δ_t (Gahinet, Apkarian and Chilali, 1994).

Expanding $\frac{\partial^2 f(\delta_t)}{\partial \delta_{i,t}^2} \geq 0, i = 1, \dots, n$ based on equations (4.45) and (4.46), the following condition is obtained:

$$\lambda^T \mathbf{M} \lambda \geq 0, \text{ for } i = 1, 2, \dots, n, \quad (4.47)$$

$$\mathbf{M} = (3\delta_{i,t} + \Delta\delta_{i,t})(\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i) +$$

$$\sum_{\substack{j=0 \\ (j \neq i)}}^n \delta_{j,t} (\mathbf{A}_j^T \mathbf{P}_i \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_j) +$$

$$\sum_{\substack{j=0 \\ (j \neq i)}}^n \Delta\delta_{j,t} \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i$$

This condition of the convexity requirement leads to the additional condition (4.44) in Theorem 4.3.

To conclude the proof, observe that equation (4.43) ensures the negativity of $f(\delta_t)$ at all corners of the parameter box. Consequently, for nonzero λ , $f(\delta_t) = \lambda^T \mathbf{L}(\delta_t) \lambda < 0$ holds

over the entire parameter box, from which, it can be concluded that $\mathbf{L}(\delta_t) < \mathbf{0}$ for all admissible δ_t .

To summarize, the additional constraint (4.44) reduces the problem of finding affine parameter-dependent Lyapunov matrices to a finite LMIs problem. Though somewhat restrictive, this still provides a significant additional number of degrees of freedom when compared to quadratic stability.

3.5.3.5 Affine quadratic Lyapunov H_∞ performance (AQLP)

Theorem 4.4 Consider the time-varying system (4.15) where δ_t , $\mathbf{A}(\delta_t)$, \mathbf{W} , $\Delta\delta_t$ and \mathbf{S} are defined the same as in Theorem 4.3. A sufficient condition for AQLP of this system is the existence of $n+1$ symmetric matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$ such that

$$\mathbf{P}(\omega) > \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (4.42)$$

$$\begin{bmatrix} \mathbf{A}(\omega)^T (\mathbf{P}(\omega) + \mathbf{P}(\tau)) \mathbf{A}(\omega) - \mathbf{P}(\omega) & \mathbf{A}(\omega)^T (\mathbf{P}(\omega) + \mathbf{P}(\tau)) \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T (\mathbf{P}(\omega) + \mathbf{P}(\tau)) \mathbf{A}(\omega) & \mathbf{B}^T (\mathbf{P}(\omega) + \mathbf{P}(\tau)) \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0} \quad (4.48)$$

for all $(\omega, \tau) \in \mathbf{W} \times \mathbf{S}$

$$(3\omega + \tau)(\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i) + \sum_{\substack{j=0 \\ (j \neq i)}}^n \tau \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i + \quad (4.49)$$

$$\sum_{\substack{j=0 \\ (j \neq i)}}^n \omega (\mathbf{A}_j^T \mathbf{P}_i \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_j + \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i) \geq \mathbf{0}$$

$$\mathbf{A}_i^T \mathbf{P}_i \mathbf{B} \geq \mathbf{0}$$

$$\text{for } i = 1, 2, \dots, n, \text{ for all } (\omega, \tau) \in \mathbf{W} \times \mathbf{S}$$

Proof of Theorem 4.4: Similarly, the convexity requirement to enforce equation (4.38) over the entire range of the parameters is equivalent to the convexity condition given by equation (4.45) and is given as follows:

$$\mathbf{G}(\delta_t) = \mathbf{A}(\delta_t)^T \mathbf{P}(\delta_{t+1}) \mathbf{B} < \mathbf{0} \quad (4.50)$$

Follow similar algebraic steps as applied in the previous theorem, the following convexity condition is obtained:

$$\begin{aligned} & (3\delta_{i,t} + \Delta\delta_{i,t})(\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i) + \sum_{\substack{j=0 \\ (j \neq i)}}^n \delta_{j,t} (\mathbf{A}_j^T \mathbf{P}_i \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_j) \quad (4.51) \\ & + \sum_{\substack{j=0 \\ (j \neq i)}}^n \Delta\delta_{j,t} \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i \geq \mathbf{0} \\ & \mathbf{A}_i^T \mathbf{P}_i \mathbf{B} \geq \mathbf{0} \\ & \text{for } i = 1, 2, \dots, n, \end{aligned}$$

Thus, the additional condition (4.49) in the Theorem 4.4 is the convexity requirement.

4.3.2 Constant uncertain parameters

In this section, the special case for which the uncertain parameters are assumed to be time-invariant and valued in the interval $\delta_{i,t} \in [\underline{\delta}_i, \overline{\delta}_i]$ is considered. The set \mathbf{S} of the parameter variation bounds given by equation (4.40) then reduces to the zero element, and hence, $\tau = 0, \mathbf{P}(\tau) = \mathbf{0}$ for all $\tau \in \mathbf{S}$ in Theorems 4.3 and 4.3. Consequently, in case of constant parameters, the conditions (4.37) and (4.38) reduce to the following inequalities:

$$\mathbf{A}(\delta_t)^T \mathbf{P}(\delta_t) \mathbf{A}(\delta_t) - \mathbf{P}(\delta_t) < \mathbf{0} \quad (4.52)$$

$$\begin{bmatrix} \mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P}(\boldsymbol{\delta}_t) \mathbf{A}(\boldsymbol{\delta}_t) - \mathbf{P}(\boldsymbol{\delta}_t) & \mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P}(\boldsymbol{\delta}_t) \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P}(\boldsymbol{\delta}_t) \mathbf{A}(\boldsymbol{\delta}_t) & \mathbf{B}^T \mathbf{P}(\boldsymbol{\delta}_t) \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0} \quad (4.53)$$

The following theorems are the corresponding simplifications of Theorems 4.3 and 4.4 respectively and apply to the case of constant uncertain parameters.

Theorem 4.5 Consider the time-varying system (4.15) where $\boldsymbol{\delta}_t$, $\mathbf{A}(\boldsymbol{\delta}_t)$, \mathbf{W} and $\Delta\boldsymbol{\delta}_t$ are defined the same as in Theorem 4.3, except that $\boldsymbol{\delta}_t$ is a vector of time-invariant but uncertain parameters. A sufficient condition for AQLS of this system is the existence of $n+1$ symmetric matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$ such that

$$\mathbf{P}(\omega) > \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (4.54)$$

$$\mathbf{A}(\omega)^T \mathbf{P}(\omega) \mathbf{A}(\omega) - \mathbf{P}(\omega) < \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (4.55)$$

$$(3\omega)(\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i) + \sum_{\substack{j=0 \\ (j \neq i)}}^n \omega(\mathbf{A}_j^T \mathbf{P}_i \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_j + \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i) \geq \mathbf{0} \quad (4.56)$$

for $i=1,2,\dots,n$, for all $\omega \in \mathbf{W}$

Theorem 4.6 Consider the time-varying system (4.15) where $\boldsymbol{\delta}_t$, $\mathbf{A}(\boldsymbol{\delta}_t)$, \mathbf{W} and $\Delta\boldsymbol{\delta}_t$ are defined the same as in Theorem 4.5. A sufficient condition for AQLP of this system is the existence of $n+1$ symmetric matrices $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$ such that

$$\mathbf{P}(\omega) > \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (4.54)$$

$$\begin{bmatrix} \mathbf{A}(\omega)^T \mathbf{P}(\omega) \mathbf{A}(\omega) - \mathbf{P}(\omega) & \mathbf{A}(\omega)^T \mathbf{P}(\omega) \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P}(\omega) \mathbf{A}(\omega) & \mathbf{B}^T \mathbf{P}(\omega) \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0}, \quad (4.57)$$

for all $\omega \in \mathbf{W}$

$$(3\omega)(\mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_i) + \sum_{\substack{j=0 \\ (j \neq i)}}^n \omega(\mathbf{A}_j^T \mathbf{P}_i \mathbf{A}_i + \mathbf{A}_i^T \mathbf{P}_i \mathbf{A}_j + \mathbf{A}_i^T \mathbf{P}_j \mathbf{A}_i) \geq \mathbf{0} \quad (4.58)$$

$$\mathbf{A}_i^T \mathbf{P}_i \mathbf{B} \geq \mathbf{0}$$

for $i = 1, 2, \dots, n$, for all $\omega \in \mathbf{W}$

Theorem 4.3, Theorem 4.4, Theorem 4.5, and Theorem 4.6 provide valuable representations for both the case of time-varying uncertain parameters and constant uncertain parameters. Specifically, the conditions given by Theorem 4.3 and Theorem 4.4, are equivalent to Theorem 4.5 and Theorem 4.6 for the case of constant uncertain parameters.

Finally, it should be noted that in the face of real time-varying parameters with bounded rate of variations, the sufficient conditions of Theorem 4.3 and Theorem 4.4 are always less conservative than the QLS and QLP tests of Theorem 4.1 and Theorem 4.2, because more variables are available for optimization. The reduction of conservatism will be clearly illustrated in Chapter 5 for the case studies.

4.4 SSV Analysis

The structured singular value for linear systems is also referred to in the literature as μ . This section will review the μ -based methods for analyzing the robust stability and performance properties of uncertain linear feedback systems, and then introduce some powerful extensions of this theory for nonlinear time-varying systems.

For simple unstructured uncertainty, robust stability leads naturally to a $\|\cdot\|_\infty$ test. The $\|\cdot\|_\infty$ norm, related to the largest singular value of an operator, thus provides a single norm which handles both the nominal performance and robust stability problems. Unfortunately, norm bounds are inadequate for dealing with more realistic models of process uncertainty with structure. Then, more complicated mathematical objects involving μ , are required. This leads to a robust stability test of the form $\|\mu(\cdot)\|_\infty \leq 1$ (Doyle and Packard, 1987). Obviously, it would be desirable to treat performance with both disturbance and uncertainties occurring simultaneously. This also leads to tests using μ . Thus μ emerges as an essential analysis tool in dealing with robust performance as well as with structured uncertainties.

4.4.1 Review of the SSV concept

The mathematical properties and computation of μ are first reviewed in the sequel for the case of complex perturbations. Here μ is viewed as a natural generalization of both spectral radius and spectral norm, and this viewpoint leads to useful characterizations of μ in terms of these more familiar quantities. One consequence is that estimates for μ can be obtained by scaling of ordinary singular values.

The structured singular value is useful to assess the robust stability and robust performance of systems represented by linear fractional interconnections, presented schematically in Figure 3.1 and Figure 4.1. This class of models has been introduced in section 3.3 and shown in Figure 3.1 for robust stability. It is shown in this section for robust performance and an additional uncertainty block Δ_{RP} is added for this purpose. In this generic model, the linear time-invariant (LTI) system $\mathbf{M} \in \mathbf{C}^{n \times n}$ represents all the known LTI components including the controller, the nominal models of the systems, sensors, and actuators. The input vector \mathbf{d} includes all external actions on the system, i.e., disturbance, noise and reference signal, and the vector \mathbf{e} consists of all output signals generated by the system. The uncertainty block $\mathbf{\Delta} = \text{diag}(\Delta_1, \dots, \Delta_n)$, which satisfies

$\bar{\sigma}(\Delta_i) \leq 1$, is a norm-bounded LTI uncertainty with some prescribed structure. $\bar{\sigma}$ denotes the maximum singular value of a matrix. $\Delta = \text{diag}(\Delta_1, \dots, \Delta_n)$ is characterized by the following aspects:

- the dimensions of each block Δ_i
- whether Δ_i is a complex or real-values matrix
- whether Δ_i is a full matrix or a scalar matrix of the form $\Delta_i = \delta_i \mathbf{I}$

Generally, the block $\Delta \in \mathbf{C}^{n \times n}$ is defined as follows:

$$\Delta = \{\text{diag}[\delta_1 \mathbf{I}_{r_1}, \dots, \delta_s \mathbf{I}_{r_s}, \Delta_1, \dots, \Delta_f] : \delta_i \in \mathbf{C}, \Delta_j \in \mathbf{C}^{m_j \times m_j}\} \quad (4.59)$$

Two nonnegative integers s and f , represent the number of repeated scalar blocks and the number of full blocks of uncertainties respectively. This structure is generally problem-specific and it depends on the nature of the uncertainty and the performance objectives of the problem. Real uncertainties typically arise from uncertain coefficients in the models of the physical systems. The focus here is on complex uncertainties because the theory is far more developed for complex uncertainties than for real ones, and also the algorithms for real uncertainties suffer often from discontinuity problems (Barmish and et al, 1990). Therefore, most applications use only complex uncertainties that include the real ones and therefore produce bounds for the original real uncertainty problem.

Let \mathbf{M} in Figure 3.1 and Figure 4.1 be partitioned as follows:

$$\begin{bmatrix} \mathbf{e} \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{d} \\ \mathbf{u}_2 \end{bmatrix} \quad (4.60)$$

such that the block structure Δ is compatible in size with \mathbf{M}_{22} . Then, the linear fractional transformation (LFT) based operator, $\mathbf{F}_l(\mathbf{M}, \Delta)$ is defined as follows:

$$\mathbf{F}_l(\mathbf{M}, \Delta) = \mathbf{M}_{11} + \mathbf{M}_{12}\Delta(\mathbf{I} - \mathbf{M}_{22}\Delta)^{-1}\mathbf{M}_{21} \quad (4.61)$$

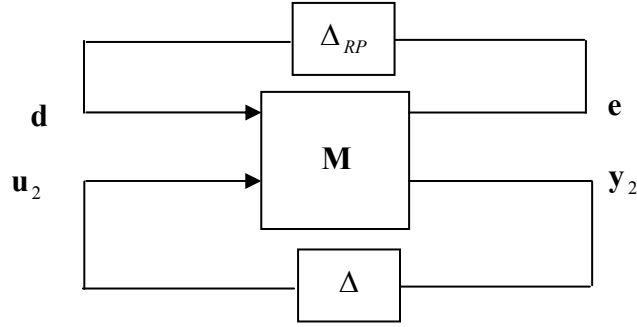


Figure 4.1 General $\mathbf{M} - \Delta$ LFT framework

From a system point of view, \mathbf{M}_{11} is the nominal map between \mathbf{d} and \mathbf{e} , and Δ affects the map in a known way, namely, through the additional matrices \mathbf{M}_{12} , \mathbf{M}_{21} , \mathbf{M}_{22} and the formula \mathbf{F}_l . The subscript “ l ” in \mathbf{F}_l pertains to the “lower” loop of \mathbf{M} which is closed by the uncertainty block Δ . An analogous formula can be used to describe $\mathbf{F}_u(\mathbf{M}, \Delta)$, which is the resulting matrix obtained by closing the “upper” loop of \mathbf{M} .

Formally, the SSV of $\mathbf{M} \in \mathbf{C}^{n \times n}$ with respect to the perturbation structure Δ , is defined as

$$\mu_{\Delta}(\mathbf{M}) := \frac{1}{\min_{\Delta \in \Delta} \{\bar{\sigma}(\Delta) : \det(\mathbf{I} - \mathbf{M}\Delta) = 0\}} \quad (4.62)$$

unless no $\Delta \in \Delta$ makes $(\mathbf{I} - \mathbf{M}\Delta)$ singular, then $\mu_{\Delta}(\mathbf{M}) = 0$. From this definition, $(\mathbf{I} - \mathbf{M}\Delta)$ remains invertible as long as $\Delta \in \Delta$ satisfies

$$\bar{\sigma}(\Delta) < 1 / \mu_{\Delta}(\mathbf{M}) \quad (4.63)$$

i.e., as long as the size of $\Delta \in \Delta$ does not exceed $K_\Delta := 1/\mu_\Delta(\mathbf{M})$. The critical size K_Δ is called the well-posedness margin. For unstructured perturbation $\Delta \in \mathbf{C}^{n \times n}$, $\bar{\sigma}(\Delta) = \mu_\Delta(\mathbf{M})$. Thus $\mu_\Delta(\mathbf{M})$ extends the notion of maximum singular value to the case of structured perturbations. μ is generally used as a frequency domain design tool, specifically, as a generalization of the $\bar{\sigma}$ design tools. Maximum singular values $\bar{\sigma}$ are useful for one full block of uncertainty, but are generally conservative when the uncertainty has structure, and the gap between μ and $\bar{\sigma}$ may be very large.

With the above definitions of μ , robust stability and performance of the system in Figure 3.1 and Figure 4.1 are given by the following theorems (Doyle and Packard, 1987).

Theorem RS: *Assume an uncertainty set Δ is defined. The feedback system in Figure 3.1 satisfies robust stability for stable Δ and $\|\Delta\|_\infty \leq 1$, iff*

$$\|\mu_\Delta(\mathbf{M}_{22})\|_\infty \leq 1 \quad (4.64)$$

where

$$\|\mu_\Delta(\mathbf{M}_{22})\|_\infty \stackrel{\text{def}}{=} \sup_\omega \mu(\mathbf{M}_{22}(j\omega)) \quad (4.65)$$

Theorem RP: *Assume an uncertainty set Δ is defined. The feedback system in Figure 4.1 satisfies robust performance for stable $\mathbf{F}_l(\mathbf{M}, \Delta)$ and $\|\mathbf{F}_l(\mathbf{M}, \Delta)\|_\infty \leq 1$, and for $\|\Delta\|_\infty \leq 1$, iff*

$$\|\mu_\Delta(\mathbf{M})\|_\infty \leq 1 \quad (4.66)$$

In summary, the robust stability and performance conditions with structured uncertainty reduce to computing μ for constant matrices $\mathbf{M}(j\omega)$, and then taking *sup* over all the ranges of frequencies ω . Unfortunately, definition (4.62) is not typically useful for

computing μ . $\mu_\Delta(\mathbf{M})$ can be easily calculated when Δ belongs to either one of the following two extreme sets.

- If $\Delta = \{\delta \mathbf{I} : \delta \in C\}$ ($s = 1, f = 0, r_1 = n$), then $\mu_\Delta(\mathbf{M}) = \rho(\mathbf{M})$. $\rho(\cdot)$ denotes the spectral radius of a matrix, i.e., the largest absolute value of the matrix's eigenvalues.
- If $\Delta = \mathbf{C}^{n \times n}$ ($s = 0, f = 1, m_1 = n$), then $\mu_\Delta(\mathbf{M}) = \bar{\sigma}(\mathbf{M})$.

For a general Δ as in equation (4.59), $\{\delta \mathbf{I} : \delta \in C\} \subset \Delta \subset \mathbf{C}^{n \times n}$. From the definition of μ and the above two extreme cases, it can be concluded that $\rho(\mathbf{M}) \leq \mu_\Delta(\mathbf{M}) \leq \bar{\sigma}(\mathbf{M})$. These bounds can be refined by considering transformations on \mathbf{M} that do not affect $\mu_\Delta(\mathbf{M})$, but do affect ρ and $\bar{\sigma}$. To do this, define the following two subsets of $\mathbf{C}^{n \times n}$:

$$\mathbf{Q}_\Delta = \{\mathbf{Q} \in \Delta : \mathbf{Q}^* \mathbf{Q} = \mathbf{I}_n\} \quad (4.67)$$

$$\mathbf{D} = \left\{ \begin{array}{l} \text{diag}[\mathbf{D}_1, \dots, \mathbf{D}_s, d_1 \mathbf{I}_{m_1}, \dots, d_{f-1} \mathbf{I}_{m_{f-1}}, \mathbf{0}_{m_f}] : \\ \mathbf{D}_i \in \mathbf{C}^{n_i \times n_i}, \mathbf{D}_i = \mathbf{D}_i^* > 0, d_j \in \mathbf{R}, d_j > 0 \end{array} \right\} \quad (4.68)$$

Therefore the bounds can be tightened as follows:

$$\max_{\mathbf{Q} \in \mathbf{Q}_\Delta} \rho(\mathbf{QM}) \leq \mu_\Delta(\mathbf{M}) \leq \inf_{\mathbf{D} \in \mathbf{D}_\Delta} \bar{\sigma}(\mathbf{DMD}^{-1}) \quad (4.69)$$

It is desirable to use both lower and upper bounds for μ using equation (4.69), since the existing bounds nicely complement each other. The lower bound is always an equality (Doyle, 1982). Unfortunately, the quantity $\rho(\mathbf{QM})$ can have multiple local maxima. Thus, a local search of this quantity cannot guarantee the finding of the true $\mu_\Delta(\mathbf{M})$, but can only yield a lower bound. The upper bound can be reformulated as a convex optimization problem because the function $\bar{\sigma}(\mathbf{DMD}^{-1})$ is convex with respect to \mathbf{D} , so

the global minimum can, in principle, be found. Unfortunately, the upper bound is not always equal to $\mu_{\Delta}(\mathbf{M})$. For block structures Δ satisfying $2s + f \leq 3$, the upper bound is always equal to $\mu_{\Delta}(\mathbf{M})$, and for block structures with $2s + f > 3$, there exist matrices for which $\mu_{\Delta}(\mathbf{M})$ is less than the upper bound.

It is important to realize that the frequency domain test, where \mathbf{D} is frequency-varying, only applies to linear and time-invariant perturbations. If the perturbations are time-varying, such as the ones considered in this work, Doyle and Packard (1988) proposed the upper bound approaches based on constant matrix \mathbf{D} optimization. These conditions will be reviewed in the next section.

Consider the class of matrices \mathbf{D} , which commute with the perturbation block Δ according to equation (4.68). If \mathbf{D} and Δ commute, then by definition:

$$\mathbf{D} \cdot \Delta \cdot \mathbf{D}^{-1} = \Delta \quad (4.70)$$

A list of appropriate $\mathbf{D} - \Delta$ commuting pairs is given in Table 4.1. If equation (4.70) holds, then it is possible to formulate less conservative robust stability and performance conditions than those proposed earlier in this section.

Table 4.1 Commuting $\mathbf{D} - \Delta$ pairs

Δ :complex	\mathbf{D}
Time-invariant, full-block	Frequency-varying, scalar-times-identity
Time-invariant, scalar-times-identity	Frequency-varying, full-block
Time-varying, full-block	Constant, scalar-times-identity
Time-varying, scalar-times-identity	Constant, full-block

Theorem RSD: *Assume an uncertainty set Δ is defined. The feedback system in Figure 3.1 satisfies robust stability for stable and time-varying Δ and $\|\Delta\|_{\infty} \leq 1$, iff*

$$\inf_{\mathbf{D}} \sup_{\omega} \bar{\sigma}(\mathbf{D}\mathbf{M}_{22}\mathbf{D}^{-1}) < 1 \quad (4.71)$$

where \mathbf{D} is appropriately constructed as in Table 4.1.

The \mathbf{D} scale which achieves or gets arbitrarily close to the infimum in equation (4.71) is referred to as the optimal \mathbf{D} scale. Absorbing the \mathbf{D} scale into the \mathbf{M} block yields the transformed \mathbf{M} matrix for the following robust performance condition.

Theorem RPD: Assume an uncertainty set Δ is defined. The feedback system in Figure 4.1 satisfies robust performance for stable $\mathbf{F}_u(\mathbf{M}, \Delta)$ and $\|\mathbf{F}_u(\mathbf{M}, \Delta)\|_{\infty} \leq 1$, and for time-varying $\|\Delta\|_{\infty} \leq 1$, iff

$$\inf_{\mathbf{D}} \sup_{\omega} \bar{\sigma} \left[\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{pmatrix} \mathbf{M} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^{-1} \end{pmatrix} \right] = \gamma < 1 \quad (4.72)$$

where \mathbf{D} is appropriately constructed as in Table 4.1.

This condition implies that

$$\|\mathbf{e}\|^2 + \|\mathbf{y}_2\|^2 \leq \gamma^2 (\|\mathbf{d}\|^2 + \|\mathbf{u}_2\|^2) \quad (4.73)$$

Furthermore, from equation (4.70) and $\|\Delta\|_{\infty} \leq 1$, it can be concluded that $\|\mathbf{y}_2\|^2 \geq \|\mathbf{u}_2\|^2$. Thus $\|\mathbf{e}\|^2 \leq \gamma^2 \|\mathbf{d}\|^2$ and $\|\mathbf{F}_l(\mathbf{M}, \Delta)\|_{\infty} \leq 1$. In a typical closed-loop system, \mathbf{d} represents the disturbance inputs and \mathbf{e} represents the output feedback errors. In order to use theorems RSD and RPD, a procedure has to be found for finding the optimal scaling matrices \mathbf{D} 's which are required on the left hand side of the inequalities (4.71) and (4.72). The following two sections establish an equivalent minimization problem which is easier to solve.

4.4.2 Generation of an $\mathbf{M} - \Delta$ LFT

As an illustration of the general procedure to obtain the $\mathbf{M} - \Delta$ LFT description, consider a discrete-time model completely described by a nominal linear process and some model uncertainty as follows:

$$\begin{aligned} \mathbf{x}(t+1) &= \left\{ \mathbf{A} + \sum_{i=1}^n \mathbf{A}_i \delta_i \right\} \mathbf{x}(t) + \left\{ \mathbf{B} + \sum_{i=1}^n \mathbf{B}_i \delta_i \right\} \mathbf{d}(t) \\ \mathbf{e}(t) &= \left\{ \mathbf{C} + \sum_{i=1}^n \mathbf{C}_i \delta_i \right\} \mathbf{x}(t) + \left\{ \mathbf{D} + \sum_{i=1}^n \mathbf{D}_i \delta_i \right\} \mathbf{d}(t) \end{aligned} \quad (4.74)$$

Here the scalar parameters δ_i 's represent the model uncertainty for the system. An LFT can be constructed for this perturbed system with an appropriate state-space matrix \mathbf{W}_Δ :

$$\mathbf{W}_\Delta = \begin{bmatrix} \mathbf{A} + \sum_{i=1}^n \mathbf{A}_i \delta_i & \mathbf{B} + \sum_{i=1}^n \mathbf{B}_i \delta_i \\ \mathbf{C} + \sum_{i=1}^n \mathbf{C}_i \delta_i & \mathbf{D} + \sum_{i=1}^n \mathbf{D}_i \delta_i \end{bmatrix} \quad (4.75)$$

This model is represented in Figure 4.2, and the perturbed LFT is given by:

$$\begin{aligned} \mathbf{e} &= \mathbf{G}_\Delta \mathbf{d} \\ \mathbf{G}_\Delta &= \mathbf{F}_u(\mathbf{W}_\Delta, z^{-1}\mathbf{I}) \end{aligned} \quad (4.76)$$

where z^{-1} is the typical one interval shift used in z -transform theory for discrete systems. It is desirable to transform it into the general $\mathbf{M} - \Delta$ framework Figure 3.1 and Figure 4.1 so that SSV analysis can be applied. To do this, it is first desired to isolate the uncertainty elements from the overall transfer function $\mathbf{G}_\Delta = \mathbf{F}_u(\mathbf{W}_\Delta, z^{-1}\mathbf{I})$. The matrix \mathbf{W}_Δ is rewritten as a feedback connection of a matrix \mathbf{N} and an uncertainty block Δ , and its LFT representation is $\mathbf{W}_\Delta = \mathbf{F}_l(\mathbf{N}, \Delta)$, shown in Figure 4.3. The matrix \mathbf{N} is an algebraic function of the elements of \mathbf{W}_Δ , but is independent of the uncertainty elements

δ_i 's. The block Δ is composed of n diagonal scalar-times-identity blocks. For example, the model given by equation (4.74) will be rewritten as follows:

$$\begin{aligned}
 \mathbf{x}(t+1) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{d}(t) + \mathbf{B}_{22}\mathbf{u}_2(t) \\
 \mathbf{e}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{d}(t) + \mathbf{D}_{12}\mathbf{u}_2(t) \\
 \mathbf{y}_2(t) &= \mathbf{C}_{22}\mathbf{x}(t) + \mathbf{D}_{21}\mathbf{d}(t) + \mathbf{D}_{22}\mathbf{u}_2(t) \\
 \mathbf{u}_2(t) &= \text{diag}[\delta_1\mathbf{I}_{n_s+1}, \delta_2\mathbf{I}_{n_s+1}, \dots, \delta_n\mathbf{I}_{n_s+1}]\mathbf{y}_2(t)
 \end{aligned} \tag{4.77}$$

where $\mathbf{B}_{22}, \mathbf{C}_{22}, \mathbf{D}_{12}, \mathbf{D}_{21}, \mathbf{D}_{22}$ are defined as follows:

$$\begin{aligned}
 \mathbf{B}_{22} &= [\mathbf{I}_{n_s \times n_s} \quad \mathbf{0}_{n_s \times 1} \quad \mathbf{I}_{n_s \times n_s} \quad \mathbf{0}_{n_s \times 1} \quad \dots]_{n_s \times n(n_s+1)} \\
 \mathbf{C}_{22} &= [\mathbf{A}_1^T \quad \mathbf{C}_1^T \quad \mathbf{A}_2^T \quad \mathbf{C}_2^T \quad \dots]_{n(n_s+1) \times n_s}^T \\
 \mathbf{D}_{12} &= [\mathbf{0}_{1 \times n_s} \quad 1 \quad \mathbf{0}_{1 \times n_s} \quad 1 \quad \dots]_{1 \times n(n_s+1)} \\
 \mathbf{D}_{21} &= [\mathbf{B}_1^T \quad \mathbf{D}_1^T \quad \mathbf{B}_2^T \quad \mathbf{D}_2^T \quad \dots]_{n(n_s+1) \times 1}^T \\
 \mathbf{D}_{22} &= [\mathbf{0}]_{n(n_s+1) \times n(n_s+1)}
 \end{aligned} \tag{4.78}$$

where n_s is the number of states and n is the number of individual uncertainties. The matrix \mathbf{N} and the uncertainty block Δ are given by:

$$\begin{aligned}
 \mathbf{N} &= \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{B}_{22} \\ \mathbf{C} & \mathbf{D} & \mathbf{D}_{12} \\ \mathbf{C}_{22} & \mathbf{D}_{21} & \mathbf{D}_{22} \end{bmatrix} \\
 \Delta &= \text{diag}\{\delta_1\mathbf{I}_{n_s+1}, \delta_2\mathbf{I}_{n_s+1}, \dots, \delta_n\mathbf{I}_{n_s+1}\}
 \end{aligned} \tag{4.79}$$

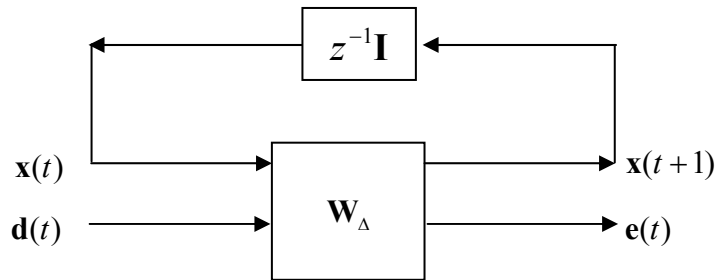


Figure 4.2 Equivalent $\mathbf{M} - \Delta$ framework (Equation (4.74))

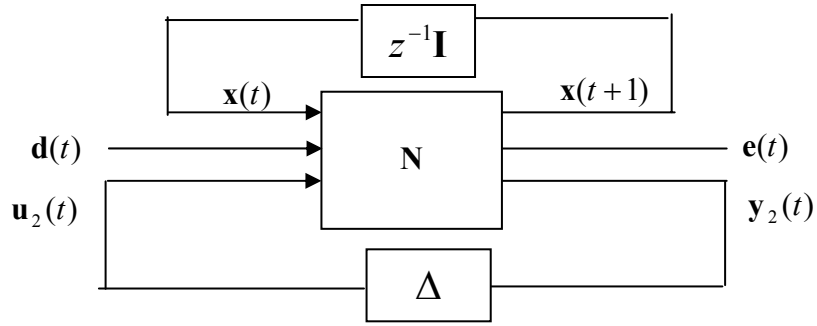


Figure 4.3 Equivalent $\mathbf{M} - \Delta$ framework (Equation (4.77))

Finally, a general LFT for the transfer-function from \mathbf{d} to \mathbf{e} can be given by:

$$\mathbf{G}_\Delta = \mathbf{F}_l\{\mathbf{F}_u(\mathbf{N}, z^{-1}\mathbf{I}), \Delta\} \quad (4.80)$$

Define $\mathbf{M} = \mathbf{F}_u(\mathbf{N}, z^{-1}\mathbf{I})$, then

$$\mathbf{G}_\Delta = \mathbf{F}_l(\mathbf{M}, \Delta) \quad (4.81)$$

Thus, the standard $\mathbf{M} - \Delta$ framework given in Figure 3.1 and Figure 4.1 can be constructed based on these definitions.

4.4.3 RS and RP conditions for time-varying uncertainty

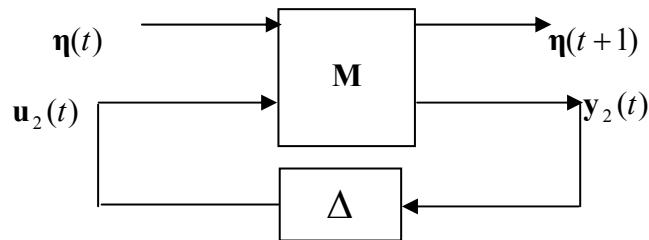


Figure 4.4 Equivalent $\mathbf{M} - \Delta$ framework (equation (4.82))

For time-varying uncertainty, Packard and Doyle (1988) have proposed sufficient conditions for robust stability and robust performance in the time domain. The motivation for this analysis is twofold: first, the usual Lyapunov results for stability and performance can be clearly represented in time domain; and second, the actual calculations involved for the scaled singular values are computationally more attractive in the time domain.

For robust stability, consider the closed-loop system represented in the standard $\mathbf{M} - \Delta$ structure Figure 4.4 and described by:

$$\begin{aligned}\boldsymbol{\eta}(t+1) &= \mathbf{F}_l(\mathbf{M}, \Delta(t))\boldsymbol{\eta}(t) \\ \Delta &= \text{diag}\{\delta_1\mathbf{I}, \delta_2\mathbf{I}, \dots, \delta_n\mathbf{I}\}\end{aligned}\tag{4.82}$$

$\boldsymbol{\eta}(t) \in \mathbf{C}^n$, $\mathbf{M} \in \mathbf{C}^{(n+m) \times (n+m)}$ and, for each time interval t , $\Delta(t)$ is an element of the uncertainty set Δ . $\Delta(t)$ satisfies the following conditions:

1. $\bar{\sigma}(\Delta(t)) \leq 1$
2. $\Delta(t)$ varies with discrete time t

The time-varying nature of $\Delta(t)$ invalidates the spectral radius arguments which do not guarantee that $\boldsymbol{\eta}(t)$ decreases for all t , as for the time-invariant case. However, the following sufficient condition does yield exponential stability:

$$\max_{\Delta(t)} \bar{\sigma}[\mathbf{F}_l(\mathbf{M}, \Delta(t))] < 1\tag{4.83}$$

If this condition is satisfied, the operator $\mathbf{F}_l(\mathbf{M}, \Delta(t))$ is referred to as a contraction. This conservative result can be strengthened by searching for a single quadratic Lyapunov function, $\boldsymbol{\eta}^* \mathbf{P} \boldsymbol{\eta}$, for the entire set of operators. A necessary and sufficient condition for the existence of such a function is given by

$$\begin{aligned} \max_{\Delta(t)} \bar{\sigma}[\mathbf{T}\mathbf{F}_l(\mathbf{M}, \Delta(t))\mathbf{T}^{-1}] &< 1, \\ \text{for some } \mathbf{T} \in \mathbf{C}^{n \times n} & \text{ (invertible)} \end{aligned} \quad (4.84)$$

This result is equivalent to the usual discrete-time Lyapunov result

$$\begin{aligned} \mathbf{F}^* \mathbf{P} \mathbf{F} - \mathbf{P} &< -\mathbf{Q}, \quad \text{where } \mathbf{Q} > \mathbf{0} \\ \text{and } \mathbf{F} &= \mathbf{F}_l(\mathbf{M}, \Delta(t)) \end{aligned} \quad (4.85)$$

In other words, $\mathbf{P} = \mathbf{T}^* \mathbf{T}$ is a suitable Lyapunov function and equivalently, given \mathbf{P} satisfying equation (4.85) then $\mathbf{T} = \mathbf{P}^{1/2}$ satisfies equation (4.84). Equation (4.84) can be rewritten as follows:

$$\begin{aligned} \max_{\Delta(t)} \bar{\sigma}[\mathbf{F}_l(\mathbf{M}_T, \Delta(t))] &< 1, \\ \text{for some } \mathbf{T} \in \mathbf{C}^{n \times n} & \text{ (invertible)} \\ \mathbf{M}_T &= \begin{pmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{T}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \end{aligned} \quad (4.86)$$

Comparing equations (4.71) and (4.86), it is clear that the Lyapunov approach involves a type of scaling similar to the optimal \mathbf{D} scale mentioned in the previous section. In this case, the scaling consists of a coordinate transformation \mathbf{T} on the state variable. It is possible to reformulate equation (4.86) by incorporating the appropriate \mathbf{D} scale. Then a sufficient robust stability condition for the closed-loop system given in Figure 4.5 is given as follows:

$$\inf_{\mathbf{T}, \mathbf{D}_2} \bar{\sigma} \left[\begin{pmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 \end{pmatrix} \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{T}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2^{-1} \end{pmatrix} \right] < 1 \quad (4.87)$$

where $\mathbf{T} \in \mathbf{C}^{n \times n}$ is invertible and \mathbf{D}_2 commutes with $\Delta(t)$. A list of $\mathbf{D} - \Delta$ commuting pairs is given in Table 4.1. This condition implies $\|\boldsymbol{\eta}(t+1)\|^2 < \|\boldsymbol{\eta}(t)\|^2$, which is

equivalent to the stability of the system. For a limited special class of uncertainties, equation (4.87) is also necessary for the existence of a single quadratic Lyapunov function. According to the SSV theory, this class is precisely those problems for which the SSV is equal to its upper bound.

Similarly, a sufficient robust performance condition for the closed-loop system given in Figure 4.6 is:

$$\inf_{\mathbf{T}, \mathbf{D}} \bar{\sigma} \left[\begin{pmatrix} \mathbf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{N}_{11} & \mathbf{N}_{12} & \mathbf{N}_{13} \\ \mathbf{N}_{21} & \mathbf{N}_{22} & \mathbf{N}_{23} \\ \mathbf{N}_{31} & \mathbf{N}_{32} & \mathbf{N}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{T}^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{D}^{-1} \end{pmatrix} \right] = \gamma < 1 \quad (4.88)$$

where \mathbf{N} is obtained by augmenting the matrix \mathbf{M} to include the effect of external inputs on the process. It is shown in Figure 4.6 for robust performance that an additional uncertainty block Δ_{RP} is added for this purpose. The diagram describing the robust performance condition, given by equation (4.88), is shown in Figure 4.6.

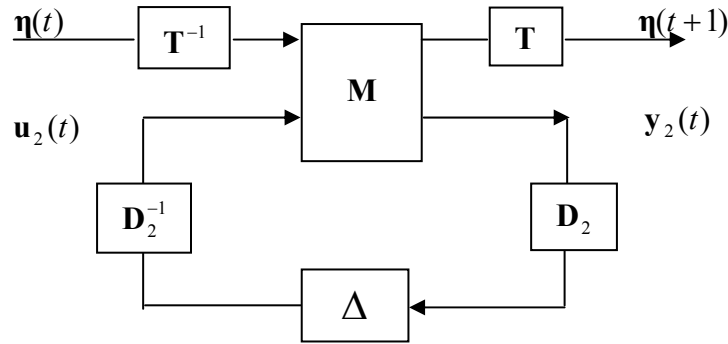


Figure 4.5 Equivalent scaled and transformed loops for robust stability

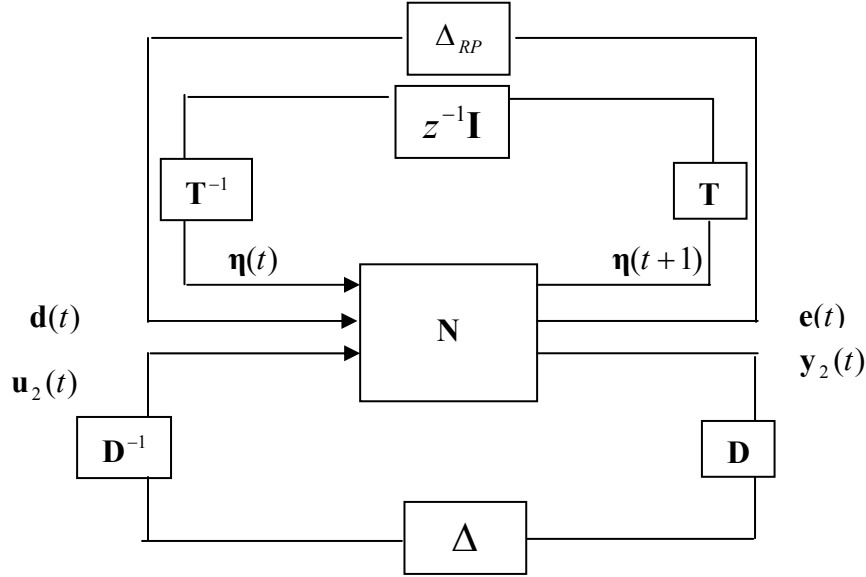


Figure 4.6 Equivalent scaled and transformed loops for robust performance

The proof of the robust stability and performance conditions given above can be found in Doyle, Packard and Morari (1989). The authors showed that the inequality (4.88) implies the desired H_∞ robust performance result: $\|\mathbf{e}\|^2 \leq \gamma^2 \|\mathbf{d}\|^2$, if the system has zero initial states.

4.5 Comparison of Quadratic Lyapunov Analysis and SSV Analysis

Based on the discussions in the previous sections, it can be seen that there is a close relationship between quadratic Lyapunov analysis and SSV analysis. As a summary, for nonlinear processes perturbed by time-varying uncertainties, for robust stability, they both guarantee the state variables decrease with time, and for robust performance, they both guarantee the H_∞ robust performance result: $\|\mathbf{e}\|^2 \leq \gamma^2 \|\mathbf{d}\|^2$, if the system has zero initial states. The conditions based on quadratic Lyapunov approach and the conditions using the upper bound of the SSV approach, are all sufficient conditions, and they can give useful conclusions when they succeed in establishing finite and feasible robust

stability and performance bounds. Both approaches are inherently conservative to some degree, because they are based on robust control design approach and they depend on the accuracy of the uncertainty bounds. However, there are many differences between the two approaches, which make one approach more functional and less conservative than the other.

First, it will be shown in a later section that quadratic Lyapunov analysis can easily deal with additional issues of input-saturation and modeling error. Also the rate of change of the uncertain time-varying parameters can be incorporated into the quadratic Lyapunov approach based design to reduce the conservatism. Specifically, parameter-dependent Lyapunov functions have been proposed to be used as an alternative to the fixed Lyapunov functions in the analysis, such that the information on the parameter time-variation can be integrated into the design. The SSV analysis can in principle also deal with additional issues of input-saturation and modeling error, but the resulting closed-loop system formulation when these issues are considered, is more complicated than for the quadratic Lyapunov approach.

Second, for time-varying uncertainties, SSV approach has assumed complex perturbations. Effect of real uncertainties can be covered by using complex ones, but more conservative results will be obtained. This makes quadratic Lyapunov approach less conservative in the case of real uncertain parameters, and this is the case of this work.

Third, the SSV analysis results reviewed in section 4.4 are based on the upper bound of the SSV, and it will give more conservative conclusions if SSV is far from its upper bound. One problem with μ is that the real value can not be accurately calculated, and it depends on the calculation of its lower and upper bounds. The calculation of its lower bound has local minimum, so it is the upper bound that is usually calculated in the case of time-varying uncertainty. For an uncertainty structure which has s repeated scalar blocks and f full blocks, the upper bound has been proven to be equal to the SSV when $s = 0, f = 1, 2, 3; s = 1, f = 1$. Otherwise, the upper bound will be far from SSV, and conservative conclusions will be obtained from the upper bound. Since the cases

considered in this thesis include a number of repeated scalars blocks, the SSV analysis is clearly conservative. Thus, the SSV approach based on its upper bound will give conservative conclusions, because it is based on sufficient conditions only.

Fourth, the application of the SSV analysis requires that the bounds of the uncertain parameters be absorbed into the system matrices \mathbf{M} and \mathbf{N} given by equation (4.87) and (4.88), so that the uncertainty is bounded between known bounds. It is often inconvenient to implement the uncertain parameters' bounds into the system and to obtain the closed-loop formulation shown in Figures 4.4 and 4.5. On the other hand, the quadratic Lyapunov approach uses the bounds of the uncertainty as the vertices of the parameter-box in a straightforward way. This point also favors the application of the quadratic Lyapunov approach over the SSV approach.

Based on the above discussions, for the generality of the approach proposed in this work, quadratic Lyapunov analysis is chosen instead of the SSV analysis. In Chapter 5, results based on both analysis methods will be given for the design of linear and gain-scheduled PI controllers. The results will be shown to favor, as expected, the use of the quadratic Lyapunov function approaches over the SSV approach, consistent with the above qualitative comparisons between these two approaches. However, the SSV approach still remains a useful tool for robust control design of linear time-invariant systems.

5 Robust Gain-Scheduled PI Controller

Gain-scheduling has proven to be a successful design methodology in many engineering applications. However, in the absence of a sound theoretical analysis, these designs come with no guarantees of robust stability, performance or even nominal stability of the overall gain-scheduled design (Shamma and Athans, 1990). The main purpose of this chapter is to present a new systematic approach to design robust gain-scheduled controllers for nonlinear processes, which guarantees closed-loop robust stability and performance. This approach is based on the analysis tools presented in Chapter 4. A large part of the work shown in this chapter has been previously reported by Gao and Budman (2004).

A gain-scheduled PI controller structure scheduling on the process input for nonlinear chemical processes is proposed in this chapter. The state-affine model under this gain-scheduled PI control results in a closed-loop system that can be shown to be an affine parameter-dependent model, with affine parameter-dependence on the process inputs. In Chapter 4, conditions on the robust stability and robust performance have been developed for this class of closed-loop system, i.e., affine parameter-dependent models. Based on these conditions, a robustness analysis is carried out to validate the design and obtain bounds, in terms of the controller tuning parameters of the closed-loop stability and performance, in the face of plant uncertainty. Thus, the robustness analysis is conducted to produce ranges of parameter values that result in closed-loop robust stability and performance.

Two additional issues, input-saturation and modeling error, are incorporated into the design, using a quadratic Lyapunov based analysis. First, the input-saturation situation occurring when the process inputs reach the controller limits, is explicitly addressed in this chapter. Second, since the state-affine model used in this work is an empirical model obtained from transformations of a Volterra series model (Sontag, 1978) identified from input/output data, modeling errors will result. The study of input-saturation and the modeling errors by the LMIs performance test, not studied in the previous work by

Budman and Knapp (2001), is discussed here. A simple way to incorporate these two problems into the LMIs test for robust stability and performance will be shown in this chapter.

This chapter is organized as follows. Section 5.1 proposes the novel gain-scheduled PI controller structure used in this work, and presents the closed-loop mathematical formulation of the process state-affine model in conjunction with this gain-scheduled PI controller. In section 5.2, design and optimization procedures are introduced for gain-scheduled PI controllers based on the robust stability and performance conditions. This section also shows the integration of the issues of input-saturation and modeling error into the design and optimization procedures. Two approaches are developed to reduce the conservatism of the design. One approach uses parameter-dependent Lyapunov functions, which has been first proposed in Chapter 4. The other approach is based on analytical calculation of the input-saturation factor bounds, which is proposed here in section 5.2. For comparison with the quadratic Lyapunov approach, in section 5.3, the design of the gain-scheduled PI controllers based on SSV (structured singular value) analysis is given. Section 5.4 illustrates the CSTR case study results and section 5.5 summarizes the conclusions of this chapter.

5.1 Gain-scheduled PI Controller

Gain-scheduling is a widely accepted technique for controlling nonlinear systems. In this section, a novel gain-scheduled PI controller, which is different from the traditional gain-scheduling approach will be presented. In contrast with Shamma and Athans' work (1990, 1991, 1992) where scheduling was conducted with respect to the output variable, in the current work, the scheduling variable is chosen to be the manipulated variable. This is a logical choice for the current work since the manipulated variable has been shown to be the source of the nonlinearity in the mathematical model representing the process in Chapter 3.

5.1.1 Closed-loop system

A gain-scheduled PI controller of the following form is proposed, where the tuning parameters are scheduled as continuous functions of the scheduling variable, i.e., the manipulated variable u as follows:

$$\begin{aligned}
 \xi(t+1) &= A_c \xi(t) + B_c e(t) \\
 \hat{u}(t) &= (C_c + \sum_{i=1}^m W_{ci} u^i(t)) \xi(t) + (D_c + \sum_{i=1}^m W_{di} u^i(t)) e(t) \\
 e(t) &= y_d(t) - y(t), y_d(t) = 0 \\
 A_c &= 1, B_c = 1, C_c = \frac{K_c}{\tau_I}, D_c = K_c + \frac{K_c}{\tau_I}
 \end{aligned} \tag{5.1}$$

where A_c, B_c, C_c, D_c are control parameters and W_{ci}, W_{di} are scheduling weights. $\xi(t)$ is the controller state, $e(t)$ is the feedback error, and $y_d(t)$ is the desired set-point of the process. $\hat{u}(t)$ is the PI controller output and $u(t)$ is the actuator output. The control action $\hat{u}(t)$ is calculated without saturation whereas $u(t)$ is computed with saturation limits. It should be noticed that this controller does not involve the problem of state estimation, which is usually not easy. The controller is used to stabilize the process and the tuning parameters to be tuned are $K_c, \tau_I, W_{ci}, W_{di}$. In this work, for simplicity, only the 1st-order scheduling weights, i.e., W_{c1}, W_{d1} , will be considered, and they will be referred to heretofore as W_c, W_d . This gain-scheduled PI controller is rewritten as follows:

$$\begin{aligned}
 \xi(t+1) &= A_c \xi(t) + B_c e(t) \\
 \hat{u}(t) &= (C_c + W_c u(t)) \xi(t) + (D_c + W_d u(t)) e(t) \\
 e(t) &= y_d(t) - y(t), y_d(t) = 0 \\
 A_c &= 1, B_c = 1, C_c = \frac{K_c}{\tau_I}, D_c = K_c + \frac{K_c}{\tau_I}
 \end{aligned} \tag{5.2}$$

When $W_c = W_d = 0$, the control law \hat{u} reduces to a conventional discrete PI controller with proportional gain and reset time K_c and τ_I respectively. Otherwise when $W_c \neq 0$ or $W_d \neq 0$ or both, the coefficients C_c and D_c of the PI controller are augmented in equation (5.2) by a linear dependency with respect to the manipulated variable $u(t)$ to allow for scheduling as a function of $u(t)$.

This controller is an output-feedback controller, and it does not require measurement of all the process states as state-feedback controllers do. In practice, measurement of the process states is usually very difficult and has to be estimated mathematically, while measurement of the process output is usually available.

The process state-affine model discussed in Chapter 3 is given as follows:

$$\begin{aligned} \mathbf{x}(t+1) &= \{\mathbf{F}_0 + \sum_{i=1}^{n-1} \mathbf{F}_i \delta_{i,t}\} \mathbf{x}(t) + \{\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t}\} u(t) \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t) \end{aligned} \quad (5.3)$$

For performance analysis, the rejection of output unmeasured disturbances is considered in this work, and the control objective is that the error $e(t) = 0 - y(t)$ remains in a desired bounded set for all bounded uncertainties and inputs. High-frequency disturbances can not be effectively rejected unless an infinite closed-loop bandwidth is used. This is clearly unattainable because of robust stability limitations. Therefore, the actual disturbance $v(t)$ entering the process is assumed to be modeled by the following set of equations:

$$\begin{aligned} y(t) &= \mathbf{H}_0 \mathbf{x}(t) + W_f d(t) \\ d(t+1) &= BWd(t) + (1 - BW)v(t) \end{aligned} \quad (5.4)$$

where $d(t)$ represents the filtered unmeasured disturbances in the system, and $0 \leq W_f \leq 1$ is a disturbance weight, and $0 \leq BW \leq 1$ is a bandwidth weight in performance computations.

Thus, to analyze the closed-loop performance, the system is described by the following parameter-dependent model:

$$\begin{aligned} \mathbf{x}(t+1) &= \{\mathbf{F}_0 + \sum_{i=1}^{n-1} \mathbf{F}_i \delta_{i,t}\} \mathbf{x}(t) + \{\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t}\} u(t) \\ y(t) &= \mathbf{H}_0 \mathbf{x}(t) + W_f d(t) \\ d(t+1) &= BW d(t) + (1-BW)v(t) \end{aligned} \quad (5.5)$$

The uncertainty description developed in Chapter 3, is given as follows:

$$\delta_{i,t} = u(t)^i \quad (5.6)$$

After substitution of equation (5.6) into equation (5.2), the following uncertain expression for the gain-scheduled PI controller is obtained:

$$\begin{aligned} \xi(t+1) &= A_c \xi(t) + B_c e(t) \\ \hat{u}(t) &= (C_c + W_c \delta_{1,t}) \xi(t) + (D_c + W_d \delta_{1,t}) e(t) \\ A_c &= 1, B_c = 1, C_c = \frac{K_c}{\tau_I}, D_c = K_c + \frac{K_c}{\tau_I} \end{aligned} \quad (5.7)$$

The process given by equation (5.5) and the controller given by equation (5.7) are combined together into one state-space representation as follows:

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \xi(t+1) \\ \frac{d(t+1)}{e(t)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}(\boldsymbol{\delta}_t) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \xi(t) \\ \frac{d(t)}{v(t)} \end{bmatrix} \quad (5.8)$$

$$\mathbf{B} = \begin{bmatrix} \mathbf{0}^T & \mathbf{0}^T & (1-BW)^T \end{bmatrix}^T, \mathbf{C} = \begin{bmatrix} -\mathbf{H}_0 & 0 & -W_f \end{bmatrix}, \mathbf{D} = [0]$$

where

$$\mathbf{A}(\boldsymbol{\delta}_t) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ -B_c \mathbf{H}_0 & A_c & -B_c W_f \\ \mathbf{0} & \mathbf{0} & BW \end{bmatrix} \quad (5.9)$$

$$\mathbf{A}_{11} = \mathbf{F}_0 + \sum_{i=1}^{n-1} \mathbf{F}_i \delta_{i,t} - (\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t})(D_c + W_d \delta_{1,t}) \mathbf{h}_0$$

$$\mathbf{A}_{12} = (\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t})(C_c + W_c \delta_{1,t})$$

$$\mathbf{A}_{13} = -(\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t})(D_c + W_d \delta_{1,t}) W_f$$

where $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \dots, \delta_n) \in \mathbf{R}^n$ is a vector of uncertain and time-varying real parameters, and according to equation (5.6), $\delta_{i,t} \times \delta_{j,t} = \delta_{i+j,t}$. Then, the state matrix $\mathbf{A}(\boldsymbol{\delta}_t)$ can be easily shown to depend affinely on the parameters as follows:

$$\mathbf{A}(\boldsymbol{\delta}_t) = \mathbf{A}_0 + \mathbf{A}_1 \delta_{1,t} + \mathbf{A}_2 \delta_{2,t} \dots + \mathbf{A}_n \delta_{n,t} \quad (5.10)$$

where $\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_n$ are *a priori* known fixed matrices.

Then, equation (5.8) can be rewritten in the more compact form:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\eta}(t+1) \\ e(t) \end{bmatrix} &= \begin{bmatrix} \mathbf{A}(\boldsymbol{\delta}_t) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ v(t) \end{bmatrix} \\ \boldsymbol{\eta}(0) &= \boldsymbol{\eta}_0 \end{aligned} \quad (5.11)$$

where the closed-loop state vector is defined as $\boldsymbol{\eta}(t) = [\mathbf{x}(t)^T, \xi(t)]^T$. Because the closed-loop systems of the state-affine model and the gain-scheduled PI controllers, given by equations (5.10) and (5.11), have affine-dependence with respect to the uncertain parameters $\delta_{i,t}$'s, the robust stability and performance conditions developed in Chapter 4 can be used to design gain-scheduled PI controllers.

5.1.2 Input-saturation

In defining $\delta_{i,t}$ according to (5.6), it was assumed that the inputs $u(t)$ remain between upper and lower limits determined, for example, by actuator constraints. Input-saturation would occur when the controller outputs $\hat{u}(t)$ exceeded the limits, e.g., $[-1 \ 1]$ in terms of normalized input values. However, in the closed-loop equations (5.8) and (5.11), the fact of controller saturation was not explicitly accounted for. To address this issue, the gain-scheduled PI controller can be reformulated using a variable gain \tilde{K}_c as follows. Define:

$$\text{When the actuator is saturated, i.e., when } |\hat{u}| \geq 1: \quad (5.12)$$

$$\psi = \frac{1}{|\hat{u}|} = \frac{1}{K_c} \frac{1}{\left| \frac{1}{\tau_I} \xi + \left(1 + \frac{1}{\tau_I}\right) e \right|}$$

$$\text{When the actuator is not saturated, i.e., when } |\hat{u}| < 1:$$

$$\psi = 1$$

Then the gain of the controller is given by:

$$\text{When the actuator is saturated:} \quad (5.13)$$

$$\text{if } 0 \leq \psi \leq 1 \quad \tilde{K}_c = K_c \psi$$

When the actuator is not saturated:

$$\text{else } \psi = 1 \quad \tilde{K}_c = K_c = \text{constant}$$

This variable gain formulation ensures that $u(t)$ never exceeds the saturation limits, and the ψ is referred heretofore as a saturation factor with $\psi \in [\underline{\psi} \quad \bar{\psi}]$, where $\underline{\psi}$ is the lower bound, and $\bar{\psi}$ is the upper bound of this factor. When there is no input-saturation, i.e., $|\hat{u}| < 1$, then from equation (5.12) $\psi > 1$, and the limits of $|\hat{u}| \rightarrow 1, \psi \in [1,1]$ will be considered. When input-saturation occurs, i.e., $|\hat{u}| \geq 1$, then from equation (5.12), the limits of $\bar{\psi} = 1$ corresponding to $|\hat{u}| = 1$, and $\underline{\psi} = 0$ corresponding to $|\hat{u}| \rightarrow \infty$ will be considered, i.e., $\psi \in [0,1]$.

This formulation raises the problem that for $\underline{\psi} = 0$, the interaction between $\xi(t)$ and $\mathbf{x}(t+1)$ is cancelled according to equation (5.7), and consequently, it is not possible to attain convergence of $\xi(t)$ to the origin. For instance, assuming $\mathbf{x}(t)$ converges to the origin, following equation (5.7), $\xi(t+1) = \xi(t)$. Consequently, the controller state may converge to a constant value different than zero, which is not the asymptotic stability required by Lyapunov theory. To ensure convergence of $\xi(t)$ to the origin, a rudimentary form of anti-windup is implemented whenever the input saturation occurs. The resulting controller is then rewritten as follows:

$$\begin{aligned} \xi(t+1) &= \psi(A_c \xi(t) + B_c e(t)) \\ u(t) &= \psi(C_c + W_c u(t))\xi(t) + \psi(D_c + W_d u(t))e(t) \\ \text{for } 0 &\leq \psi \leq 1 \end{aligned} \tag{5.14}$$

In addition, when the lower limit of saturation factor is assumed to be zero for the case that $|\hat{u}| \rightarrow \infty$, i.e., when $\underline{\psi} = 0$, the closed-loop performance condition as defined by Theorem 4.2 was found in the examples to be very conservative, and in some cases could

not be met. Fortunately, the output in a real process is always bounded due to sensor saturation or the physical limitation of the process, e.g., concentration is always between $[0 \ 1]$. Thus, in reality, the controller output $\hat{u}(t)$ in a process will not achieve infinity, but bounded by practical constraints. As a result, the lower limit of ψ would be $\underline{\psi} > 0$. This fact can be used to relax the lower limit of the saturation factor in order to meet the robust performance criterion and will be further discussed later in section 5.2.3.

The closed-loop system models for the purpose of the LMIs based approach, taking into account the input-saturation factor ψ , are obtained as follows.

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \xi(t+1) \\ d(t+1) \\ e(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}(\delta_t) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \xi(t) \\ d(t) \\ v(t) \end{bmatrix} \quad (5.15)$$

$$\mathbf{B} = [\mathbf{0}^T \quad \mathbf{0}^T \quad (1 - BW)^T]^T \quad \mathbf{C} = [-\mathbf{H}_0 \quad 0 \quad -W_f] \quad \mathbf{D} = [0]$$

where

$$\mathbf{A}(\delta_t) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ -B_c \psi \mathbf{H}_0 & A_c \psi & -B_c \psi W_f \\ \mathbf{0} & \mathbf{0} & BW \end{bmatrix} \quad (5.16)$$

$$\mathbf{A}_{11} = \mathbf{F}_0 + \sum_{i=1}^{n-1} \mathbf{F}_i \delta_{i,t} - (\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t}) (D_c + W_d \delta_{1,t}) \psi \mathbf{H}_0$$

$$\mathbf{A}_{12} = (\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t}) (C_c + W_c \delta_{1,t}) \psi$$

$$\mathbf{A}_{13} = -(\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t}) (D_c + W_d \delta_{1,t}) \psi W_f$$

The inclusion of the input-saturation factor as an additional uncertain parameter will add more vertices to the LMIs test corresponding to limits of $\psi \in [\underline{\psi} \ \bar{\psi}]$. The closed-loop system given by equations (5.15) and (5.16) does not have the standard parameter affine-dependence structure shown in Chapter 4, with respect to both $\delta_{i,t}$'s and ψ . However, the robustness conditions developed in Chapter 4 can still be applied to the system. The rationale is that equation (5.15) is quadratic with respect to the uncertain parameters $\delta_{i,t}$'s when $\psi = 1$, i.e., for the case when there is no input saturation, and equation (5.15) is linear with respect to ψ and quadratic with respect to the uncertain parameters $\delta_{i,t}$'s, when $|\delta_{i,t}| = 1$ corresponding to the saturation situation. Thus, the robust stability and performance conditions are still quadratic or linear with respect to the uncertain parameters $\delta_{i,t}$'s and ψ , and it is possible to check the vertices of the uncertain parameter box instead of checking each internal point of it.

For SSV analysis method, the $\mathbf{M} - \Delta$ LFT framework of the closed-loop system must be obtained, and the uncertainty will include not only the uncertain parameters $\delta_{i,t}$'s, but also the input-saturation factor ψ . Since SSV is not the main design approach in this work, the corresponding SSV formulation has not been investigated for the saturation case.

5.1.3 Modeling error

In addition to the nonlinear time-varying powers of $u(t)$ accounted for as model uncertainty between the state-affine model and the linear nominal model, the approximation of the real process by an empirical state-affine model also results in some modeling error.

Modeling error will arise due to both truncations of the infinite Volterra series model to a finite one and its subsequent transformation step into the state-affine model. There are

different ways to account for the modeling error in the final state-affine model. In principle, modeling error exists in each one of the polynomial matrices $\mathbf{F}_i, \mathbf{G}_i, \mathbf{H}_i$ in the state-affine model due to the truncation and approximation issues. However, it is difficult to identify how the modeling error is distributed among these matrices. In the current study, for simplicity, a lumped error δ_i in the output is considered so that the \mathbf{H} matrix can be rewritten as follows:

$$\mathbf{H} = \mathbf{H}_0 \xrightarrow{\delta_i} \mathbf{H} = \mathbf{H}_0 + W_i \delta_i \quad (5.17)$$

The modeling error uncertainty is normalized so that $|\delta_i| = 1$ and the weight W_i gives the magnitude of the largest expected modeling error. W_i is calculated from comparisons of the actual system output with the prediction output of the state-affine model, by solving the following minimum problem:

$$W_i \delta_i = \min_{W_i} (\max_{t_k} |y((t_k)_{actual}) - (\mathbf{H}_0 + W_i \delta_i)x(t_k)|) \quad (5.18)$$

where $y(t_k)_{actual}$ is the output of the actual process and $(\mathbf{H}_0 + W_i \delta_i)x(t_k)$ is the prediction output of the state-affine model including the modeling error. The input sequence used to identify the model may be also used to identify the modeling error $W_i \delta_i$ by solving the above problem. Starting with an initial estimate of $W_i \delta_i$, equation (5.18) is solved using iterative optimization. The matrix \mathbf{H}_0 in the closed-loop equations (5.15) and (5.16), needs to be modified to include this modeling error based on equation (5.17). When modeling error is not considered, $W_i = 0$. When there is modeling error, W_i is set to be the magnitude of the modeling error calculated from equation (5.18), with $\delta_i = \pm 1$. In summary, for the quadratic Lyapunov approach, the inclusion of modeling error as an

additional uncertain parameter will add more vertices to the LMIs test corresponding to limits of $\delta_t = [0], [-1], [1]$.

The resulting closed-loop system does not have the standard parameter affine-dependence structure shown in Chapter 4, with respect to both $\delta_{i,t}$'s, ψ and δ_t . However, the robustness conditions developed in Chapter 4 can still be applied to the system. The reason is that the robust stability and performance conditions are still quadratic or linear with respect to the uncertain parameters $\delta_{i,t}$'s, ψ and δ_t , and it is possible to check the vertices of the uncertain parameter box instead of checking each internal point of it.

For the SSV design method, the $\mathbf{M} - \Delta$ LFT framework of the closed-loop system must be obtained, and the uncertainty will include not only the uncertain parameters $\delta_{i,t}$'s, but also the modeling error δ_t . This will be shown later in section 5.3.1.

5.2 Design and Optimization using Quadratic Lyapunov Functions

Note that in Chapter 4, for systems that can be put in the form of equation (5.11), LMIs-based conditions have been developed for the analysis of robust stability and performance. Based on these conditions, robust gain-scheduled PI controllers are designed and optimized.

5.2.1 Design of robust gain-scheduled PI controller

By trial and error, regions of controller parameters values in the parameter space given by $\theta = \{K_c, \tau_I, W_c, W_d\}$ are generated by checking if the conditions in Theorems 4.1 and 4.2 are satisfied. A set of gain-scheduled PI controllers are designed based on the feasibility of equation (4.26), which guarantees that a desired performance criterion $\gamma_{objective}$ is satisfied.

For a pre-specified performance index $\gamma_{objective}$ for a process, the procedure to design a robust gain-scheduled PI controller is as follows:

1. Set a range and a discrete grid of values in that range for the controller design parameters set θ , i.e., K_c, τ_I, W_c, W_d .
2. Choose values for θ according to the grid values within the parameter range. Set W_c, W_d in the set θ to zero if linear PI controllers are designed.
3. Substitute values of the set θ and $\gamma_{objective}$ into equations of Theorem 4.2.
4. Solve the above equation as a FEASP problem in MATLAB.
5. If a feasible solution exists for the above equation, accept values chosen in step 2, otherwise, discard the current values.
6. Go to step 2.

The same procedure as above will be used to design a PI controller satisfying robust stability, but equations of Theorem 4.1 are used in step 3, instead of Theorem 4.2 equations. For reducing conservatism, parameter-dependent Lyapunov functions could be used instead of the fixed-parameter Lyapunov functions. In this case, the equations corresponding to Theorems 4.3 and 4.4 are used instead of Theorems 4.1 and 4.2 respectively. Theorems 4.5 and 4.6 will be applied in the case of constant uncertain parameters.

5.2.2 Optimization of robust gain-scheduled PI controllers

The performance index γ can be optimized by solving equations of Theorem 4.2 as a GEVP (generalized eigenvalue problem) problem in MATLAB. Since the performance of the controller is directly related to the parameter γ , the objective of this optimization problem is to minimize this parameter γ . The equation of Theorem 4.2 is as follows:

$$\begin{bmatrix} \mathbf{A}(\omega)^T \mathbf{P} \mathbf{A}(\omega) - \mathbf{P} & \mathbf{A}(\omega)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\omega) & \mathbf{B}^T \mathbf{P} \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0}, \text{ for all } \omega \in \mathbf{W} \quad (5.19)$$

It is easy to show that this problem falls into the standard form of a GEVP problem if it is rewritten in the following alternative form:

$$\begin{aligned} & \min_{\mathbf{P}} \gamma^2 & (5.20) \\ & \text{subject to} & \begin{bmatrix} \mathbf{A}(\omega)^T \mathbf{P} \mathbf{A}(\omega) - \mathbf{P} & \mathbf{A}(\omega)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\omega) & \mathbf{B}^T \mathbf{P} \mathbf{B} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \gamma^2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ & \text{for all } \omega \in \mathbf{W} \end{aligned}$$

If a controller which optimizes the closed-loop performance is sought, the procedure to design a robust gain-scheduled PI controller with near optimal performance is as follows:

1. Set a range and a discrete grid of values in that range for the controller design parameters set $\boldsymbol{\theta}$, i.e., K_c, τ_I, W_c, W_d .
2. Choose values for $\boldsymbol{\theta}$ according to the grid values within the parameter range. Set W_c, W_d in the set $\boldsymbol{\theta}$ to zero if linear PI controllers are designed.
3. Substitute values of the set $\boldsymbol{\theta}$ into equation (4.33).
4. Minimize γ subject to equation (4.33) (GEVP problem in MATLAB).

5. If a feasible solution exists for equation (4.33), accept values chosen in step 2 and the optimized performance index γ , otherwise, discard the current values.
6. Go to step 2 until $\gamma_{optimal} = \min(\gamma)$ is obtained over the whole parameter range.

For reducing conservatism, parameter-dependent Lyapunov functions could be used instead of the fixed-parameter Lyapunov functions. In this case, Theorem 4.4 is used instead of Theorem 4.2. Theorem 4.6 can also be applied in the case of constant uncertain parameters.

The problem of searching for the optimal performance index, $\gamma_{optimal}$, is not quadratic in terms of the controller parameters K_c, τ_I, W_c, W_d and the optimization matrix variable \mathbf{P} simultaneously. Thus, the resulting problem is a nonlinear matrix inequality for all of these parameters. For example, equation (4.33) includes higher-order terms like $K_c^2 \mathbf{P} / \tau_I, W_c \mathbf{P} W_d$. Thus, the optimization in terms of all of these parameters may be near optimal instead of a global optimal solution. Branch and bound methods have been proposed to solve LMIs that are not convex with respect to certain variables (Fukuda and Kojima, 2001; Braatz, VanAntwerp & Sahinidis, 1997). This is beyond the scope of the current study.

5.2.3 Relaxation of the input-saturation factor ψ

In section 5.2.1, a robust control approach has been proposed to design gain-scheduled PI controllers, which guarantee closed-loop stability and performance. The inherent conservatism of the robust control analysis results in smaller ranges of parameters that satisfy the design criteria and consequently in degraded performance based on these parameter ranges. When the rate of change of the uncertain parameters is available, a design based on parameter-dependent Lyapunov functions may be used to reduce conservatism. This approach has been proposed in detail in Chapter 4. The main focus of

this section is to propose a second approach to reduce conservatism based on the calculation of less conservative saturation factor bounds.

When the lower limit of the saturation factor is assumed to be zero for the case that the calculated control action tends to infinity, i.e., when $\underline{\psi} = 0$, the closed-loop performance condition as defined by the equation in Theorem 4.2 was found to be very conservative, and in some cases could not be met. Fortunately, the output in a real process is always bounded due to sensor saturation or the physical limitation of the process, e.g., concentration is always between $[0 \ 1]$ and this fact can be used to reduce conservatism and to meet the robust performance criterion. For example, using the physical limits of the output, a finite upper limit for the control action $|\hat{u}|$ exists and consequently a lower bound of ψ different than zero according to equation (5.12) can be calculated analytically as follows.

Method 5.1 (saturation factor lower bound $\underline{\psi}$) Consider the controller (5.7) with the error signal bounded $e(t) \in [e \ \bar{e}]$. The analytical saturation factor lower bound is calculated as follows:

Step 1: For any $e(t) \in [e \ \bar{e}]$, if $\{|\hat{u}(k-1)| < 1 \text{ and } |\hat{u}(k)| \geq 1\}$, then $\underline{\psi}(t) = 1/|\hat{u}(k)|$;

Step 2: For all $e(t) \in [e \ \bar{e}]$, $\underline{\psi} = \min(\underline{\psi}(t))$.

Step 1 involves iterative calculation of the controller output $\hat{u}(k)$ using equation (5.1) for each error $e(t)$ in the range of $[e \ \bar{e}]$ until a $\underline{\psi}(t)$ is obtained for each $e(t)$ based on equation (5.12). Then a set of $\underline{\psi}(t)$ values is obtained for the range of the error signal $e(t) \in [e \ \bar{e}]$. Step 2 consists in deriving the minimum value of all the $\underline{\psi}(t)$ values obtained in Step 1, i.e., $\underline{\psi} = \min(\underline{\psi}(t))$, which is then adopted as the lower bound of the saturation factor to be used for the LMIs analysis.

5.3 Design based on SSV Analysis

Nonlinear and/or time-varying uncertainty can be also addressed using extensions of SSV analysis, and for simplicity, these extensions based on the upper bound of μ will be referred to as SSV analysis in this section. The application of SSV analysis to the design of gain-scheduled PI controllers is presented in the sequel, for comparison with the LMIs based methodology.

5.3.1 Generation of an $\mathbf{M} - \Delta$ LFT: simple case

It has been shown in a previous section that a nonlinear process can be completely described by a state-affine model given by equation (5.5), which is composed of a nominal linear process and some model uncertainty. The proposed gain-scheduled PI controller is given by equation (5.7). To apply the SSV approach to the robust stability and robust performance analysis of the closed-loop system, it is desired to first transform equations (5.5) and (5.7) into the standard $\mathbf{M} - \Delta$ structure.

For this purpose the state-affine model is partitioned as follows:

$$\begin{aligned}
 \mathbf{x}(t+1) &= \mathbf{F}_0 \mathbf{x}(t) + \mathbf{G}_1 u(t) + \mathbf{B}_{22} \mathbf{u}_2(t) \\
 y(t) &= \mathbf{H}_0 \mathbf{x}(t) + \mathbf{D}_{12} u_2(t) + W_f d(t) \\
 \mathbf{y}_2(t) &= \mathbf{C}_{22} \mathbf{x}(t) + \mathbf{D}_{21} u(t) + \mathbf{D}_{22} \mathbf{u}_2(t) \\
 \mathbf{u}_2(t) &= \text{diag}[\delta_1 \mathbf{I}_{n_s}, \delta_2 \mathbf{I}_{n_s}, \dots, \delta_k \mathbf{I}_{n_s}] \mathbf{y}_2(t) \\
 d(t+1) &= BWd(t) + (1 - BW)v(t)
 \end{aligned} \tag{5.21}$$

where $\delta_i = \delta_{i,t}$ for simple notation, and k is the number of uncertainties in the state-affine model. The schematic description of this formulation is shown in Figure 4.3. The matrices $\mathbf{B}_{22}, \mathbf{C}_{22}, \mathbf{D}_{12}, \mathbf{D}_{21}, \mathbf{D}_{22}$ are

$$\begin{aligned}
\mathbf{B}_{22} &= \begin{bmatrix} \mathbf{I}_{n_s \times n_s} & \mathbf{I}_{n_s \times n_s} & \cdots \end{bmatrix}_{n_s \times kn_s} \\
\mathbf{C}_{22} &= \begin{bmatrix} \mathbf{F}_1^T & \mathbf{F}_2^T & \cdots \end{bmatrix}_{kn_s \times n_s}^T \\
\mathbf{D}_{12} &= \begin{bmatrix} \mathbf{0}_{1 \times n_s} & \mathbf{0}_{1 \times n_s} & \cdots \end{bmatrix}_{1 \times kn_s} \\
\mathbf{D}_{21} &= \begin{bmatrix} \mathbf{G}_2^T & \mathbf{G}_3^T & \cdots \end{bmatrix}_{kn_s \times 1}^T \\
\mathbf{D}_{22} &= [\mathbf{0}]_{kn_s \times kn_s}
\end{aligned} \tag{5.22}$$

The gain-scheduled PI controller, equation (5.7) is partitioned as follows:

$$\begin{aligned}
\xi(t+1) &= A_c \xi(t) + B_c e(t) + B_{c22} u_{c2}(t) \\
u(t) &= C_c \xi(t) + D_c e(t) + D_{c12} u_{c2}(t) \\
y_{c2}(t) &= C_{c22} \xi(t) + D_{c21} e(t) + D_{c22} u_{c2}(t) \\
u_{c2}(t) &= [\delta_1] y_{c2}(t) \\
B_{c22} &= [0]_{1 \times 1} \quad D_{c22} = [0]_{1 \times 1} \quad D_{c12} = [1]_{1 \times 1} \\
C_{c22} &= [W_c]_{1 \times 1} \quad D_{c21} = [W_d]_{1 \times 1}
\end{aligned} \tag{5.23}$$

The closed-loop system is obtained by combining equations (5.21), (5.22) and (5.23) and the closed-loop state vector is defined as $\boldsymbol{\eta}(t) = [\mathbf{x}(t)^T, \xi(t)^T]^T$. The input and output of the uncertainty block are $\mathbf{y}_\Delta(t) = [\mathbf{y}_2^T(t) \quad y_{c2}]^T$ and $\mathbf{u}_\Delta(t) = [\mathbf{u}_2^T(t) \quad u_{c2}]^T$. For the robust stability framework shown in Figure 4.5, the effect of external inputs on the process is not considered, so the \mathbf{M} matrix and the uncertainty block structure are given as follows:

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{y}_\Delta \end{bmatrix} &= \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{u}_\Delta \end{bmatrix} \\
\mathbf{u}_\Delta(t) &= \text{diag}[\delta_1 \mathbf{I}_{n_s}, \delta_2 \mathbf{I}_{n_s}, \dots, \delta_k \mathbf{I}_{n_s}, \delta_1] \mathbf{y}_\Delta(t)
\end{aligned} \tag{5.24}$$

where the matrices are as follows:

$$\begin{aligned}
\mathbf{M}_{11} &= \begin{bmatrix} \mathbf{F}_0 - \mathbf{G}_1 D_c \mathbf{H}_0 & \mathbf{G}_1 C_c \\ -B_c \mathbf{H}_0 & A_c \end{bmatrix} \\
\mathbf{M}_{12} &= \begin{bmatrix} \mathbf{B}_{22} - \mathbf{G}_1 D_c \mathbf{D}_{12} & \mathbf{G}_1 D_{c12} \\ -B_c \mathbf{D}_{12} & B_{c22} \end{bmatrix} \\
\mathbf{M}_{21} &= \begin{bmatrix} \mathbf{C}_{22} - \mathbf{D}_{21} D_c \mathbf{H}_0 & \mathbf{D}_{21} C_c \\ -D_{c21} \mathbf{H}_0 & C_{c22} \end{bmatrix} \\
\mathbf{M}_{22} &= \begin{bmatrix} \mathbf{D}_{22} - \mathbf{D}_{21} D_c \mathbf{D}_{12} & \mathbf{D}_{21} D_{c12} \\ -D_{c21} \mathbf{D}_{12} & D_{c22} \end{bmatrix}
\end{aligned} \tag{5.25}$$

For robust performance framework shown in Figure 4.6, the filtered output unmeasured disturbance d is considered, and the closed-loop state is augmented as $\boldsymbol{\eta}(t) = [\mathbf{x}(t)^T \quad \xi(t) \quad d(t)]^T$. The matrix \mathbf{N} and the uncertainty block structure are given as follows:

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{y}_\Delta \\ e(t) \end{bmatrix} &= \begin{bmatrix} \mathbf{N}_{11} & \mathbf{N}_{12} & \mathbf{N}_{13} \\ \mathbf{N}_{21} & \mathbf{N}_{22} & \mathbf{N}_{23} \\ \mathbf{N}_{31} & \mathbf{N}_{32} & \mathbf{N}_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{u}_\Delta \\ v(t) \end{bmatrix} \\
\mathbf{u}_\Delta(t) &= \text{diag}[\delta_1 \mathbf{I}_{n_s}, \delta_2 \mathbf{I}_{n_s}, \dots, \delta_k \mathbf{I}_{n_s}, \delta_1] \mathbf{y}_\Delta(t)
\end{aligned} \tag{5.26}$$

where the matrices are:

$$\begin{aligned}
\mathbf{N}_{11} &= \begin{bmatrix} \mathbf{M}_{11} & -\mathbf{G}_1 D_c W_f \\ \mathbf{0}_{1 \times n_s} & 0 \end{bmatrix} & \mathbf{N}_{12} &= \begin{bmatrix} \mathbf{M}_{12} \\ \mathbf{0}_{1 \times kn_s} \end{bmatrix} & \mathbf{N}_{13} &= \begin{bmatrix} \mathbf{0}_{n_s \times 1} \\ 0 \\ 1 - BW \end{bmatrix} \\
\mathbf{N}_{21} &= \begin{bmatrix} \mathbf{M}_{21} & -\mathbf{D}_{21} D_c W_f \\ -D_{c21} W_f \end{bmatrix} & \mathbf{N}_{22} &= \mathbf{M}_{22} & \mathbf{N}_{23} &= \begin{bmatrix} \mathbf{0}_{kn_s \times 1} \\ 0 \end{bmatrix} \\
\mathbf{N}_{31} &= [-\mathbf{H}_0 \quad 0 \quad -W_f] & \mathbf{N}_{32} &= [\mathbf{0}_{1 \times (kn_s + 1)}] & \mathbf{N}_{33} &= [0]
\end{aligned} \tag{5.27}$$

5.3.2 Generation of an $\mathbf{M} - \Delta$ LFT: with modeling error

If the effect of modeling error is considered, the following state-affine model is obtained by combining equations (5.5) and (5.17).

$$\begin{aligned}
\mathbf{x}(t+1) &= \{\mathbf{F}_0 + \sum_{i=1}^{n-1} \mathbf{F}_i \delta_{i,t}\} \mathbf{x}(t) + \{\mathbf{G}_1 + \sum_{i=1}^{n-1} \mathbf{G}_{i+1} \delta_{i,t}\} \mathbf{u}(t) \\
y(t) &= (\mathbf{H}_0 + W_t \delta_t) \mathbf{x}(t) + W_f d(t) \\
d(t+1) &= BWd(t) + (1-BW)v(t)
\end{aligned} \tag{5.28}$$

The state-affine model is partitioned into the structure shown in Figure 4.3 as follows:

$$\begin{aligned}
\mathbf{x}(t+1) &= \mathbf{F}_0 \mathbf{x}(t) + \mathbf{G}_1 \mathbf{u}(t) + \mathbf{B}_{22} \mathbf{u}_2(t) \\
y(t) &= \mathbf{H}_0 \mathbf{x}(t) + \mathbf{D}_{12} \mathbf{u}_2(t) + W_f d(t) \\
\mathbf{y}_2(t) &= \mathbf{C}_{22} \mathbf{x}(t) + \mathbf{D}_{21} \mathbf{u}(t) + \mathbf{D}_{22} \mathbf{u}_2(t) \\
\mathbf{u}_2(t) &= \text{diag}[\delta_1 \mathbf{I}_{n_s}, \delta_2 \mathbf{I}_{n_s}, \dots, \delta_k \mathbf{I}_{n_s}, \delta_t] \mathbf{y}_2(t) \\
d(t+1) &= BWd(t) + (1-BW)v(t)
\end{aligned} \tag{5.29}$$

where $\delta_i = \delta_{i,t}$ for simple notation, and k is the number of uncertainties in the state-affine model. The matrices $\mathbf{B}_{22}, \mathbf{C}_{22}, \mathbf{D}_{12}, \mathbf{D}_{21}, \mathbf{D}_{22}$ are

$$\begin{aligned}
\mathbf{B}_{22} &= \begin{bmatrix} \mathbf{I}_{n_s \times n_s} & \mathbf{I}_{n_s \times n_s} & \cdots & \mathbf{I}_{n_s \times n_s} & \mathbf{0}_{n_s \times 1} \end{bmatrix}_{n_s \times (kn_s + 1)} \\
\mathbf{C}_{22} &= \left(\begin{bmatrix} \mathbf{F}_1^T & \mathbf{F}_2^T & \cdots & \mathbf{F}_k^T & (W_t \mathbf{I}_{1 \times n_s})^T \end{bmatrix} \right)_{(kn_s + 1) \times n_s} \\
\mathbf{D}_{12} &= \begin{bmatrix} \mathbf{0}_{1 \times n_s} & \mathbf{0}_{1 \times n_s} & \cdots & \mathbf{0}_{1 \times n_s} & 1 \end{bmatrix}_{1 \times (kn_s + 1)} \\
\mathbf{D}_{21} &= \left(\begin{bmatrix} \mathbf{G}_2^T & \mathbf{G}_3^T & \cdots & \mathbf{G}_k^T & W_t \end{bmatrix} \right)_{(kn_s + 1) \times 1} \\
\mathbf{D}_{22} &= [\mathbf{0}]_{(kn_s + 1) \times (kn_s + 1)}
\end{aligned} \tag{5.30}$$

The gain-scheduled PI controller given by equation (5.7) is partitioned into equation (5.23) obtained in the previous section.

The closed-loop system is obtained by combining equations (5.29), (5.30) and (5.23) and the closed-loop state is defined as $\boldsymbol{\eta}(t) = [\mathbf{x}(t)^T, \boldsymbol{\xi}(t)^T]^T$. The input and output of the uncertainty block are $\mathbf{y}_\Delta(t) = [\mathbf{y}_2^T(t) \ y_{c2}]^T$ and $\mathbf{u}_\Delta(t) = [\mathbf{u}_2^T(t) \ u_{c2}]^T$. For robust stability, the \mathbf{M} matrix and the uncertainty structure in Figure 4.5 are given as follows:

$$\begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{y}_\Delta \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{u}_\Delta \end{bmatrix} \quad (5.31)$$

$$\mathbf{u}_\Delta(t) = \text{diag}[\delta_1 \mathbf{I}_{n_s}, \delta_2 \mathbf{I}_{n_s}, \dots, \delta_k \mathbf{I}_{n_s}, \delta_t, \delta_1] \mathbf{y}_\Delta(t)$$

where the matrices are:

$$\begin{aligned} \mathbf{M}_{11} &= \begin{bmatrix} \mathbf{F}_0 - \mathbf{G}_1 D_c \mathbf{H}_0 & \mathbf{G}_1 C_c \\ -B_c \mathbf{H}_0 & A_c \end{bmatrix} \\ \mathbf{M}_{12} &= \begin{bmatrix} \mathbf{B}_{22} - \mathbf{G}_1 D_c \mathbf{D}_{12} & \mathbf{G}_1 D_{c12} \\ -B_c \mathbf{D}_{12} & B_{c22} \end{bmatrix} \\ \mathbf{M}_{21} &= \begin{bmatrix} \mathbf{C}_{22} - \mathbf{D}_{21} D_c \mathbf{H}_0 & \mathbf{D}_{21} C_c \\ -D_{c21} \mathbf{H}_0 & C_{c22} \end{bmatrix} \\ \mathbf{M}_{22} &= \begin{bmatrix} \mathbf{D}_{22} - \mathbf{D}_{21} D_c \mathbf{D}_{12} & \mathbf{D}_{21} D_{c12} \\ -D_{c21} \mathbf{D}_{12} & D_{c22} \end{bmatrix} \end{aligned} \quad (5.32)$$

For robust performance test, the output unmeasured disturbance d is considered, and the closed-loop state is augmented as $\boldsymbol{\eta}(t) = [\mathbf{x}(t)^T \ \boldsymbol{\xi}(t) \ d(t)]^T$. The matrix \mathbf{N} and the uncertainty structure in Figure 4.6 are given as follows:

$$\begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{y}_\Delta \\ e(t) \end{bmatrix} = \begin{bmatrix} \mathbf{N}_{11} & \mathbf{N}_{12} & \mathbf{N}_{13} \\ \mathbf{N}_{21} & \mathbf{N}_{22} & \mathbf{N}_{23} \\ \mathbf{N}_{31} & \mathbf{N}_{32} & \mathbf{N}_{33} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{u}_\Delta \\ v(t) \end{bmatrix} \quad (5.33)$$

$$\mathbf{u}_\Delta(t) = \text{diag}[\delta_1 \mathbf{I}_{n_s}, \delta_2 \mathbf{I}_{n_s}, \dots, \delta_k \mathbf{I}_{n_s}, \delta_t, \delta_1] \mathbf{y}_\Delta(t)$$

where the matrices are:

$$\begin{aligned}
\mathbf{N}_{11} &= \begin{bmatrix} \mathbf{M}_{11} & -\mathbf{G}_1 D_c W_f \\ \mathbf{0}_{1 \times n_s} & 0 & BW \end{bmatrix} & \mathbf{N}_{12} &= \begin{bmatrix} \mathbf{M}_{12} \\ \mathbf{0}_{1 \times (kn_s+1)} & 0 \end{bmatrix} & \mathbf{N}_{13} &= \begin{bmatrix} \mathbf{0}_{n_s \times 1} \\ 0 \\ 1 - BW \end{bmatrix} \\
\mathbf{N}_{21} &= \begin{bmatrix} \mathbf{M}_{21} & -\mathbf{D}_{21} D_c W_f \\ -D_{c21} W_f \end{bmatrix} & \mathbf{N}_{22} &= \mathbf{M}_{22} & \mathbf{N}_{23} &= \begin{bmatrix} \mathbf{0}_{(kn_s+1) \times 1} \\ 0 \end{bmatrix} \\
\mathbf{N}_{31} &= \begin{bmatrix} -\mathbf{H}_0 & 0 & -W_f \end{bmatrix} & \mathbf{N}_{32} &= \begin{bmatrix} \mathbf{0}_{1 \times (kn_s+1)} & 0 \end{bmatrix} & \mathbf{N}_{33} &= [0]
\end{aligned} \tag{5.34}$$

5.3.3 Design of robust gain-scheduled PI controllers: SSV analysis

According to equation (5.6), the perturbations are equal to the powers of the manipulated variable u and consequently, the uncertainty is time-varying. For this type of uncertainty, Packard and Doyle (1988) have proposed sufficient conditions for robust stability and robust performance.

A sufficient robust stability condition for the system given by equation (5.24) is:

$$\inf_{\mathbf{T}, \mathbf{D}} \bar{\sigma} \left[\begin{pmatrix} \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{T}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^{-1} \end{pmatrix} \right] < 1 \tag{5.35}$$

where $\mathbf{T} \in \mathbf{C}^{(n_s+1) \times (n_s+1)}$ is invertible and \mathbf{D} commutes with $\Delta(t)$. According to equation (5.24), the uncertainty block is time-varying scalar-times-identity, and the commuting \mathbf{D} is a constant full-block matrix, i.e., $\mathbf{D} \in \mathbf{C}^{(kn_s+1) \times (kn_s+1)}$. If the modeling error δ_i is going to be considered during the design, then the closed-loop system will be given by equation (5.31), and the uncertainty structure is also given in equation (5.31). According to this equation, the commuting \mathbf{D} is a constant full-block matrix but with a dimension of $\mathbf{D} \in \mathbf{C}^{(kn_s+2) \times (kn_s+2)}$.

Similarly, a sufficient robust performance condition for the system given by equation (5.26) is:

$$\inf_{\mathbf{T}, \mathbf{D}} \bar{\sigma} \left[\begin{pmatrix} \mathbf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{N}_{11} & \mathbf{N}_{12} & \mathbf{N}_{13} \\ \mathbf{N}_{21} & \mathbf{N}_{22} & \mathbf{N}_{23} \\ \mathbf{N}_{31} & \mathbf{N}_{32} & \mathbf{N}_{33} \end{pmatrix} \begin{pmatrix} \mathbf{T}^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \right] = \gamma < 1 \quad (5.36)$$

where $\mathbf{T} \in \mathbf{C}^{(n_s+1) \times (n_s+1)}$ is invertible and $\mathbf{D} \in \mathbf{C}^{(kn_s+1) \times (kn_s+1)}$ commutes with $\Delta(t)$. If the modeling error δ_t is being considered during the design, then the closed-loop system will be given by equation (5.33), and the uncertainty structure is given by equation (5.33). According to this equation, the commuting \mathbf{D} is a constant full-block matrix but with a dimension of $\mathbf{D} \in \mathbf{C}^{(kn_s+2) \times (kn_s+2)}$.

The conditions given by equations (4.87) and (4.88) are investigated for the closed-loop system given by equations (5.24) and (5.26), or equations (5.31) and (5.33), to guide the selection of the gain-scheduled PI controller parameters K_c , τ_I , W_c and W_d . According to these conditions for robust stability, only the dynamic states and the uncertainty feedback related variables are considered, whereas for robust performance, the external disturbances and the error are also considered.

By trial and error, regions of controller parameters, $\boldsymbol{\theta} = \{K_c, \tau_I, W_c, W_d\}$, will be generated by checking if the conditions (4.87) and (4.88) are satisfied. A set of gain-scheduled PI controllers can be designed, which satisfy a desired performance criterion $\gamma_{objective}$.

For a pre-specified performance index $\gamma_{objective}$, the procedure to design a robust gain-scheduled PI controller is as follows:

1. Set a range of values and a discrete numerical grid of values within that range for the controller design parameters set $\boldsymbol{\theta}$, i.e., K_c, τ_I, W_c, W_d .

2. Choose values for the set θ according to the parameter grid inside the parameter range. Set W_c, W_d in θ to zero if linear PI controllers are designed.
3. Substitute values of θ and $\gamma_{objective}$ into equation (4.88).
4. Solve the above equation.
5. If a feasible solution exists for the above equation, accept values chosen in step 2, otherwise, discard the current values.
6. Go to step 2.

The same procedure as above is used to design a gain-scheduled PI controller satisfying robust stability, by using equation (4.87) in step 3, instead of equation (4.88).

5.4 CSTR Case Study

For the CSTR problem, the open-loop system was initially studied by performing step changes in the input, i.e., cooling water temperature, and measuring their effect on the output, i.e., the reactor concentration. Then, 1st-order transfer functions were identified from these step tests. A summary of the open-loop properties of the CSTR process selected for the current study is given in Figure 5.1 and it shows that, assuming that the process can be approximated as a 1st-order one, the CSTR system has varying process gain and time-constant over the input range of $x_c = [-10 \ 40]$. This is assumed to be the operating range for the current work, and the CSTR process is nonlinear over this range.

The state-affine model obtained in Chapter 3 is used in the design of the gain-scheduled PI controllers. A modeling error weight of $W_l = 0.025$ was identified from simulations

according to equation (5.18) and it will be included in all the following design results. First, no input-saturation is included, and the cases of fixed-parameter Lyapunov functions in section 5.4.1.1 and parameter-dependent Lyapunov functions in section 5.4.2.1 are compared, based on the quadratic Lyapunov designs. Second, the effect of input-saturation will be investigated through different designs in section 5.4.2.2. Last, the results obtained using the SSV approach are given in section 5.4.3. In all the design results shown in this chapter, for the linear PI controllers, the stability region is the area above the stability boundary including the boundary, and for the gain-scheduled PI controllers, the stability region is the area inside the boundary including the boundary. These rules apply to the performance regions as well.

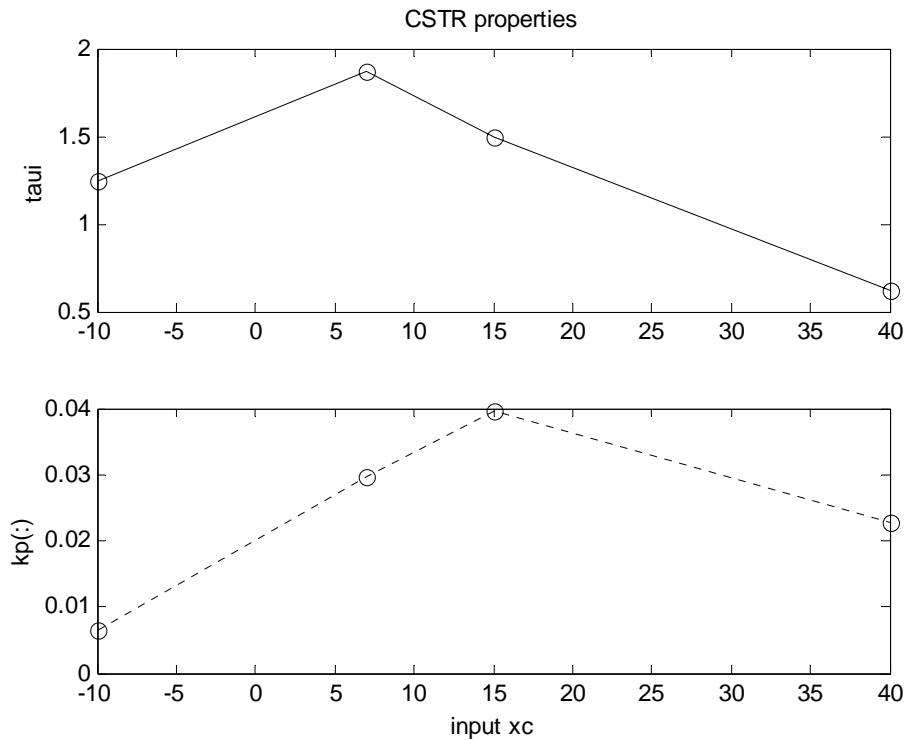


Figure 5.1 Open-loop properties of CSTR ($x_c = [-10 \ 40]$)

5.4.1 Design and optimization using quadratic Lyapunov functions

This section will summarize the design and optimization results based on fixed-parameter Lyapunov functions. As explained in the beginning of this section, the state-affine model obtained in Chapter 3 is used in the design of the gain-scheduled PI controllers. A modeling error weight of $W_t = 0.025$ is included in the design and input saturation is initially not considered. Part of the results shown here has been reported by Gao and Budman (2003).

5.4.1.1 Design of gain-scheduled PI controllers

First, linear PI controllers, i.e., with the scheduling parameters W_c and W_d set to zero, are designed using the procedure proposed in section 5.2.1. The design results are plotted in Figure 5.2 as regions in a system of coordinates corresponding to the proportional gain and reset time K_c, τ_I , respectively. Linear PI controllers with the parameter values inside these regions will guarantee robust stability and robust performance, i.e., $\gamma \leq 1$, for the closed-loop system, whereas parameter values outside these regions do not satisfy the robust stability and robust performance tests, i.e., $\gamma > 1$.

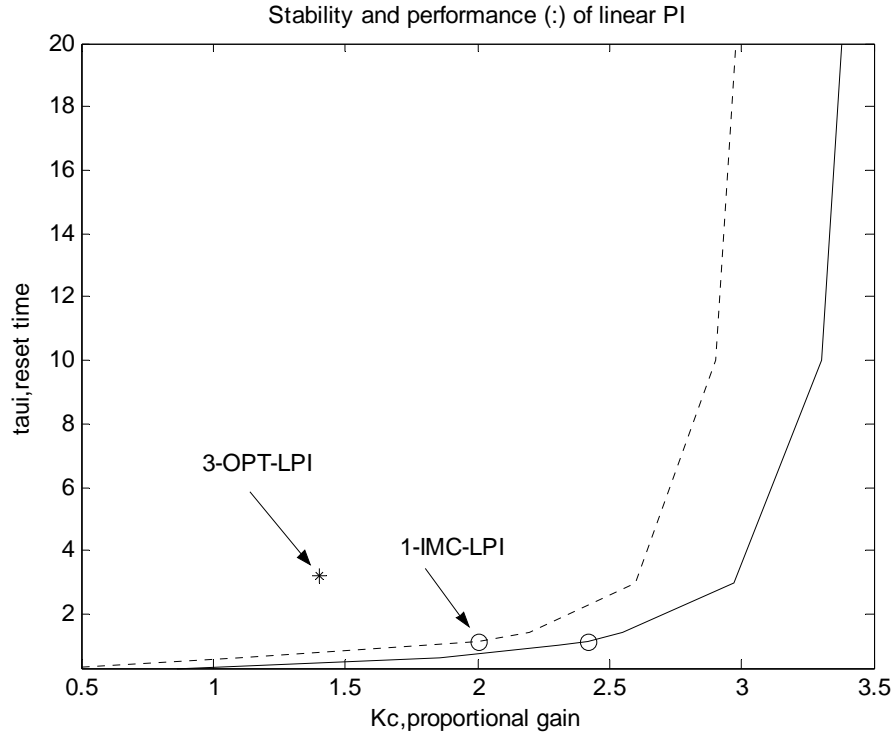


Figure 5.2 Stability and performance regions (to the left of the lines) of linear PI controller parameters.

Second, gain-scheduled PI controllers with W_c and W_d different than zero are designed using the procedure proposed in section 5.2.1. For the purpose of comparison with the linear PI controllers and also as an initial guess for further optimization of controller parameters, a set of linear PI controller parameters was first selected in Figure 5.2 as follows: $K_c = 2, \tau_I = 1.1545$ on the robust performance boundary, shown as a circle on the boundary in Figure 5.2. This set of parameters was selected as follows. Assuming that the process can be modeled by a 1st-order transfer-function around the nominal operating point, i.e., $\bar{x}_c = 0$, where \bar{x}_c is the normalized deviated variable of x_c , the time-constant τ is found to be 1.1545 seconds based on a step test around this point. According to the Internal Model Control (IMC) rules for PID controller settings, available in the literature (Morari and Zafiriou, 1989, Rivera, Morari and Skogestad), τ_I is set equal to $\tau = 1.1545$. Thus, the controller with $K_c = 2, \tau_I = 1.1545$ corresponds to the IMC tuning parameters

around the nominal operating point on the robust performance boundary, and it is referred heretofore as 1-IMC-LPI. Based on the linear IMC PI controller parameters, gain-scheduled PI controller weights W_c and W_d will be calculated according to the performance test. Figure 5.3 shows the regions in terms of W_c and W_d required to satisfy robust performance conditions. The gain-scheduled PI controllers defined by parameter values within the regions in Figure 5.3 will guarantee robust performance with $\gamma \leq 1$. The circle in Figure 5.3 corresponds to the 1-IMC-LPI controller selected on the limit of the robust performance, i.e., with $W_c = 0, W_d = 0$.

Similarly, the controller with $K_c = 2.42, \tau_I = 1.1545$ corresponds to the IMC tuning parameters around the nominal operating point on the robust stability boundary, shown as a circle on the boundary in Figure 5.2. Based on the linear IMC PI controller parameters, gain-scheduled PI controller weights W_c and W_d will be calculated according to the stability tests. Figure 5.4 shows the regions in terms of W_c and W_d required to satisfy robust stability conditions. The gain-scheduled PI controllers defined by parameter values within the regions in Figure 5.4 will guarantee robust stability. The circle in Figure 5.4 corresponds to the linear IMC PI controllers selected on the limit of the robust stability, i.e., with $W_c = 0, W_d = 0$.

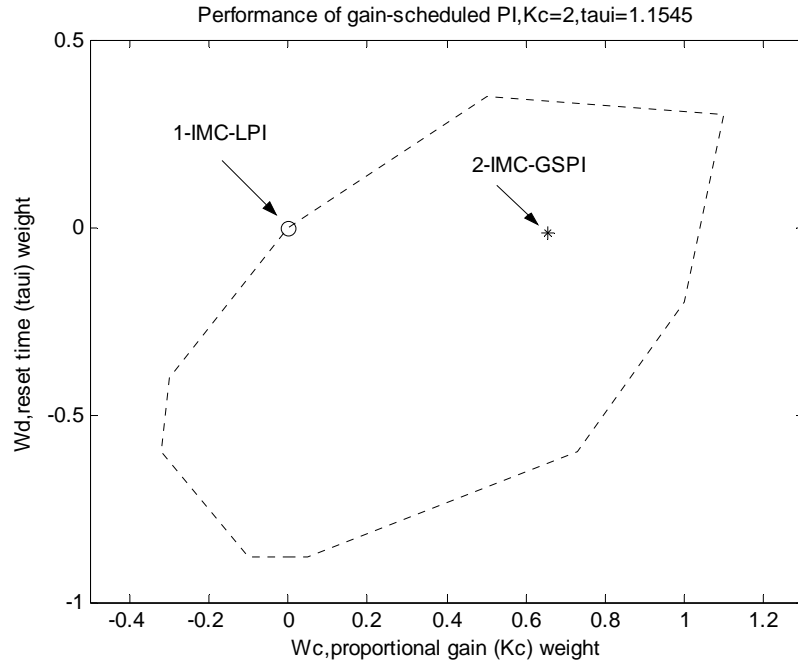


Figure 5.3 Performance region (inside the lines) of gain-scheduled PI controller parameters.

5.4.1.2 Optimization of gain-scheduled PI controllers

In order to improve upon the performance, the designed gain-scheduled PI controllers are to be optimized based on the optimization procedure proposed in section 5.2.2. In section 5.2.2, it has been discussed that the problem of searching for the optimal performance index, $\gamma_{optimal}$, is not quadratic in terms of the controller parameters, resulting in a nonlinear matrix inequality for these parameters. For simplicity, it was decided to limit the search to a near optimal design in the neighborhood of the selected linear PI controller using the FMIN optimization function in MATLAB, by selecting the linear PI parameters as initial guess for optimization. The FMIN algorithm is based on golden section search and parabolic interpolation, and it tries to find a minimum of a function.

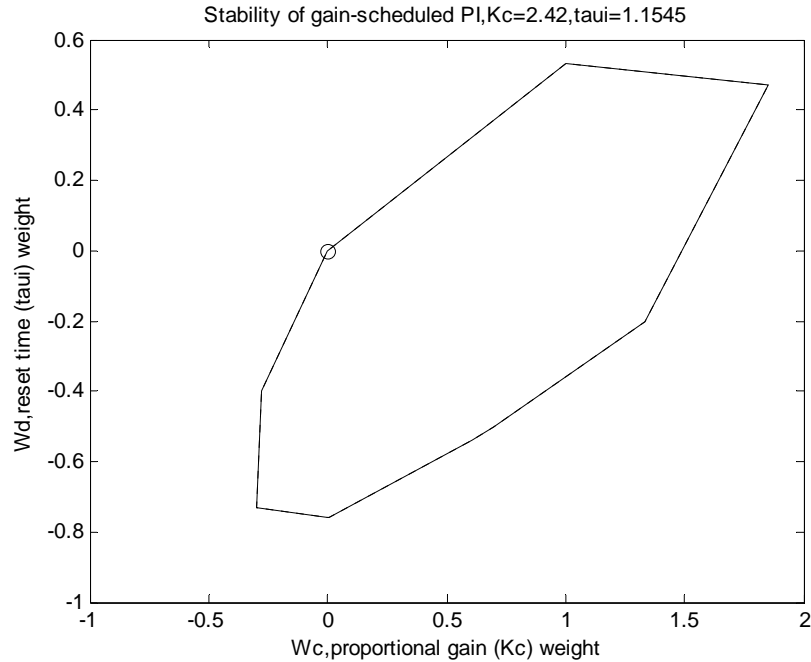


Figure 5.4 Stability region (inside the lines) of gain-scheduled PI controller parameters.

The optimization can be carried out in different ways depending on the design objectives as follows:

- To obtain a set of optimized linear PI controller parameters. In this case, only the controller parameters K_c, τ_I are optimized.
- To obtain a gain-scheduled PI controller, which improves over a designed linear PI controller. In this case, only the controller scheduling weights W_c, W_d are optimized.
- To obtain a set of optimized gain-scheduled PI controller parameters. In this case, all the four controller parameters K_c, τ_I and W_c, W_d are optimized.

Specifically, the optimization carried out in this work for the case study example CSTR started with initial guesses corresponding to the values of the linear PI controller tuned

using the IMC rules, i.e., the 1-IMC-LPI controller. The five design cases carried out in this work are explained as follows:

1. The linear PI controller, $K_c = 2, \tau_I = 1.1545$, referred to as 1-IMC-LPI in the sequel has a performance index $\gamma_{optimal}^0$ of 0.9634.

2. A gain-scheduled PI controller is designed based on the optimization procedure to improve closed-loop performance over the 1st controller, 1-IMC-LPI. A pair of gain scheduling weights W_c, W_d is to be sought inside the robust performance parameter region defined in Figure 5.3. The resulting controller will be referred in the following discussion as 2-IMC-GSPI. The controller 2-IMC-GSPI, shown as a star in Figure 5.3, produces $\gamma_{optimal}^* = 0.5890$ and this is an improvement of 38.9% over $\gamma_{optimal}^0 = 0.9634$ of the 1-IMC-LPI design.

3. A set of linear PI controller parameters K_c, τ_I , was obtained through optimization, and the resulting controller is referred to as 3-OPT-LPI. The optimized linear PI controller 3-OPT-LPI, shown as a star in Figure 5.2, improved the robust performance by 63.1% with $\gamma_{optimal} = 0.3552$ over the 1-IMC-LPI design.

4. Using the K_c, τ_I obtained for 3-OPT-LPI, the values of the weights W_c, W_d were further optimized. This controller is referred to as 4-OPT-GSPI-1. This design further improved the performance to $\gamma_{optimal} = 0.3291$, which is an additional improvement of 7.35% over 3-OPT-LPI.

5. Subsequently, an additional optimization was conducted where all the parameters, i.e., K_c, τ_I and the weights W_c, W_d , were allowed to change simultaneously in order to minimize γ . The resulting controller based on the optimization of all the four parameters is referred to as 5-OPT-GSPI-2. When all the four parameters are optimized, the best result, i.e., the smallest γ , is obtained. This case is better than all the other four cases as

expected. For this case an additional 45.6% improvement in performance over 2-IMC-GSPI is obtained with $\gamma_{optimal}=0.3204$.

The above results are summarized in Table 5.1.

Table 5.1 Optimization design results

No.	Controller name	Controller parameters θ				$\gamma_{optimal}$	$\gamma_{simulation}$
		K_c	τ_I	W_c	W_d		
1	IMC-LPI	2	1.1545	0	0	0.9634	0.3787
2	IMC-GSPI	2	1.1545	0.6547	-0.015	0.5890	0.3495
3	OPT-LPI	1.4023	3.2087	0	0	0.3552	0.2022
4	OPT-GSPI-1	1.4023	3.2087	0.1033	0.0721	0.3291	0.2009
5	OPT-GSPI-2	1.2168	1.9309	0.1802	0.009	0.3204	0.2025

5.4.1.3 Simulation of gain-scheduled PI controllers

Clearly, all the values of the performance index γ reported in Table 5.1 represent the worst possible performance according to the robust performance test. Therefore, some conservatism is expected. To assess the conservatism of the analysis, a detailed simulation study is conducted for the CSTR process using the different controllers synthesized in this work. $\gamma_{simulation}$ is the performance index obtained from the simulation, calculated using $\|\mathbf{e}\|_{l_2} = \gamma_{simulation} \|\mathbf{v}\|_{l_2}$. Different disturbance signals were used in the simulations, including for example step signals, sinusoidal signals, white noise and combinations of them. A multi-spike disturbance signal was selected to be used in the following simulations, because it resulted in the worst performance among different cases for the different signals and the results are clear to quantify for comparison purpose. Then for the worst case found from simulation, $\gamma_{simulation}$ was calculated for the different controllers and the results are reported in Table 5.1. It is clear from this table that

$\gamma_{simulation}$ is always bounded by $\gamma_{optimal}$, confirming that the analysis tests produce the worst-performance bound as expected. However, the differences between $\gamma_{optimal}$ and $\gamma_{simulation}$ for all controllers show that the designs are conservative to some degree.

Simulation results for the 1-IMC-LPI controller and the near optimal gain-scheduled PI controller 5-OPT-GSPI-2 are shown in Figure 5.5. These simulations correspond to a two-consecutive-spike-like disturbance signal shown in Figure 5.5.

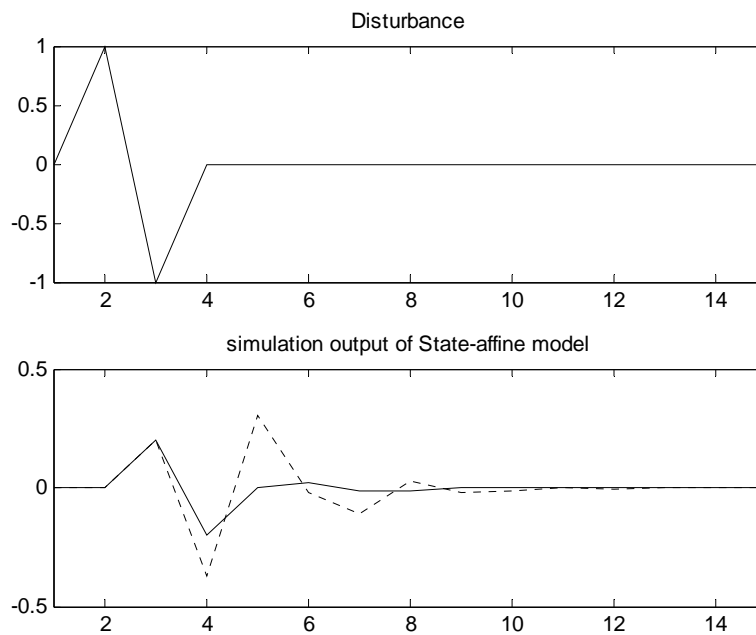


Figure 5.5 Closed-loop simulations of state-affine model (lower two curves).

1-IMC-LPI (dotted line), $\gamma_{simulation}=0.3787$.

5-OPT-GSPI-2 (solid line), $\gamma_{simulation}=0.2025$.

The results in Table 5.1 show that optimization of the tuning parameters has reduced the conservatism of the designs. For the 1-IMC-LPI controller, the difference between the analysis $\gamma_{optimal} = 0.9634$ and the simulation $\gamma_{simulation} = 0.3787$ is bigger than the difference between $\gamma_{optimal} = 0.3204$ and $\gamma_{simulation} = 0.2025$ for the optimized

controller 5-OPT-GSPI-2. A simulated performance worse than the one shown in Figure 5.5, that will bring the simulated result closer to the analysis result, may be possible but there is no systematic way to find the specific disturbance function that will lead to the largest γ value.

Conservatism associated with the design approach comes from a number of facts. First, a possible source of this conservatism is that the simulation can only be done for a limited period of time, while the calculation of the performance index $\gamma_{simulation}$ requires an infinite simulation interval. Second, conservatism is obviously inherent to the robust control approach where several scenarios included in the analysis will not actually occur during actual closed-loop operation. Last but not the least, for time-varying uncertainty parameters, conservatism might be introduced if the time-variation of the parameters is not explicitly considered in the design and optimization. The conservatism of the analysis associated with this time-variation can be somewhat reduced with parameter-dependent Lyapunov functions instead of the fixed Lyapunov functions as shown in the following section.

5.4.2 Reducing conservatism of the quadratic design and optimization

In this section, the two approaches proposed in this work will be applied to the previous designs shown in section 5.4.1.1, and the results will show that the two methods are both efficient in terms of reducing conservatism of the quadratic designs. In section 5.4.2.1, the parameter-dependent Lyapunov function is used and the design results are compared with those results obtained with fixed Lyapunov function (section 5.4.1.1). In section 5.4.2.2, the method of obtaining a less conservative lower bound of the input-saturation factor ψ is applied, and its effectiveness in reducing the conservatism of the designs will also be given in that section. As in the previous sections, the same state-affine model obtained in Chapter 3 is used in this section, and the modeling error weight of

$W_i = 0.025$ is included in the designs. Part of the results shown here has been reported by Gao and Budman (2004).

5.4.2.1 Design based on parameter-dependent Lyapunov functions

In this section, stability and performance conditions were calculated with parameter-dependent Lyapunov functions and compared to the results using a fixed-parameter Lyapunov function. The purpose is to reduce the conservatism of the designs obtained in the previous sections. No input-saturation was considered in this section.

In Chapter 3, it has been discussed that the simplicity in the quantification of the uncertainty is the key advantage of using the state-affine model, i.e., the function given by equation (5.37). For a normalized process input, it is valid to assume that $u(t) \in [-1 \ 1]$. According to equations (5.37) and (5.38), the bound of $\delta_{i,t} \in [\underline{\delta}_i \ \overline{\delta}_i]$ can thus be obtained, for example, $\delta_{1,t} \in [-1 \ 1], \delta_{2,t} \in [0 \ 1]$, where $\delta_{1,t} = u(t), \delta_{2,t} = u(t)^2$. According to equation (5.39), the bound of $\Delta\delta_{i,t} \in [\underline{v}_i \ \overline{v}_i]$ can also be obtained, for example, assuming the largest possible change $\Delta\delta_{1,t} \in [-2 \ 2]$, because $u(t) \in [-1 \ 1]$.

$$\delta_{i,t} = u(t)^i \quad (5.37)$$

$$u(t) \in [\underline{u} \ \overline{u}] \rightarrow \delta_{i,t} \in [\underline{\delta}_i, \overline{\delta}_i] \quad (5.38)$$

$$\Delta\delta_{i,t} = \delta_{i,t+1} - \delta_{i,t}, \Delta\delta_{i,t} \in [\underline{v}_i \ \overline{v}_i] \quad (5.39)$$

Based on the above information, the parameter box \mathbf{W} for the uncertain parameters and \mathbf{S} for the rate of change of the uncertainty parameters can both be determined based on the following definitions:

$$\mathbf{W} := \{(\omega_1, \omega_2, \dots, \omega_n) : \omega_i \in \{\underline{\delta}_i, \overline{\delta}_i\}\} \quad (5.40)$$

$$\mathbf{S} := \{(\tau_1, \tau_2, \dots, \tau_n) : \tau_i \in \{\underline{\nu}_i, \overline{\nu}_i\}\} \quad (5.41)$$

In Chapter 4, Theorems 4.3 and 4.4 have been developed to assess robust stability and robust performance. The theorems depend on the application of parameter-dependent Lyapunov functions when the rate of change of the uncertain parameters is available. In the sequel, these two theorems will be applied to design gain-scheduled PI controllers. The design results will be compared with the results obtained in the section 5.4.1.1.

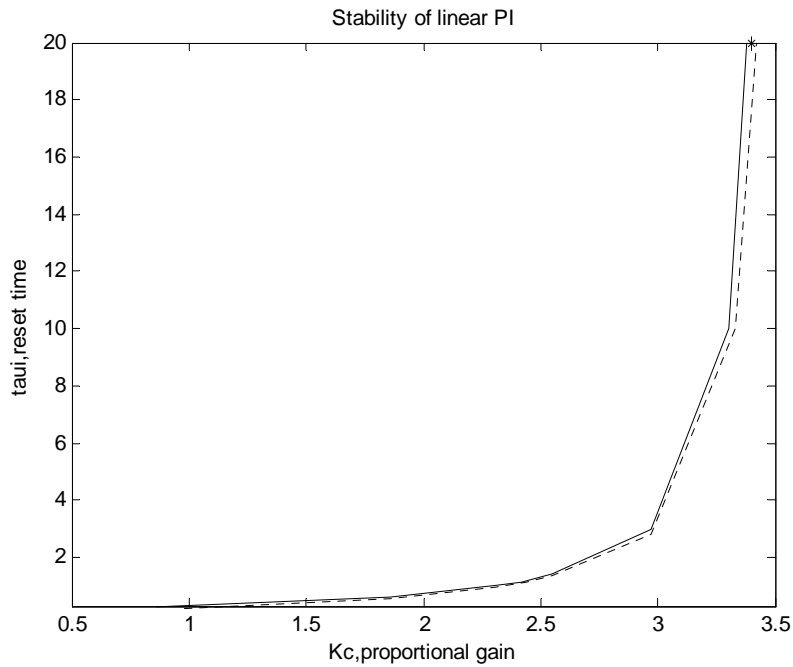


Figure 5.6 Stability region (to the left of the lines) of linear PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).

First, linear PI controllers with W_c and W_d set to zero are designed. The stability boundaries in terms of the values of the tuning parameters K_c and τ_I are shown in

Figure 5.6. The solid line, calculated by Theorem 4.1, is based on the fixed-parameter Lyapunov function and the dotted line, calculated by Theorem 4.3, is based on the parameter-dependent Lyapunov function. Linear PI controllers with the parameter values inside these regions will guarantee robust stability. The stability boundaries are very close to each other, but the one based on the parameter-dependent Lyapunov function defines, as expected, a slightly larger stability region. For a clearer illustration of this slight difference, a portion of these two lines is shown again in Figure 5.6a.

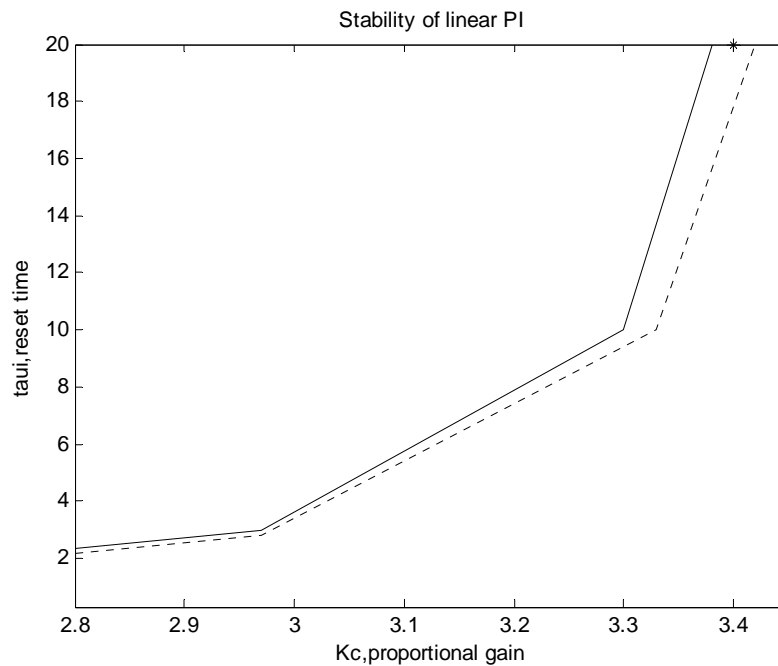


Figure 5.6a Stability region (to the left of the lines) of linear PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).

Similar results are obtained for the performance boundaries and shown in Figure 5.7. For the fixed parameter Lyapunov function, the results are shown by the solid line and for the parameter-dependent Lyapunov function, shown by the dotted line. These lines were calculated using Theorems 4.2 and 4.4. Once again, the region of robust performance obtained using the parameter-dependent Lyapunov function is slightly larger than the one

obtained with the fixed parameter Lyapunov function. For a clearer illustration of this slight difference, a portion of these two lines is shown again in Figure 5.7a.

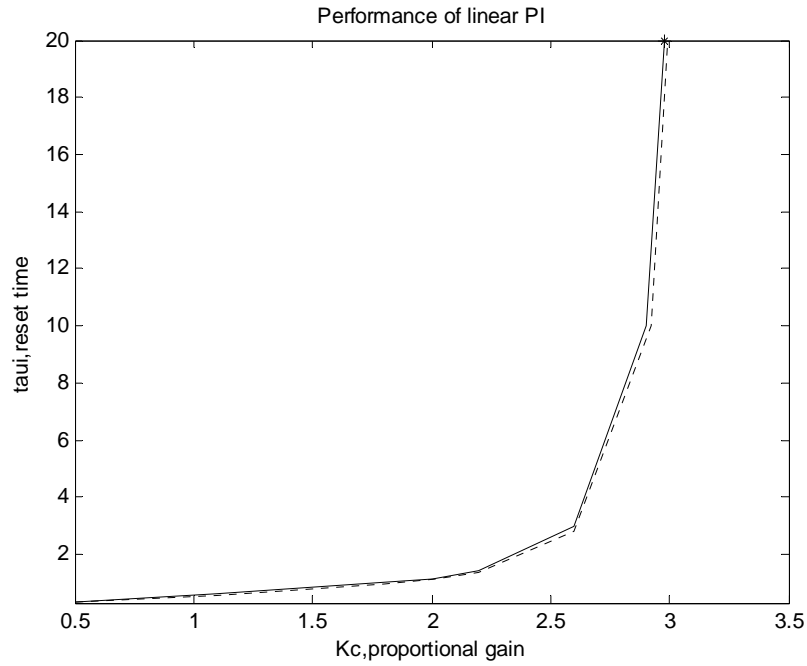


Figure 5.7 Performance region (to the left of the lines) of linear PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).

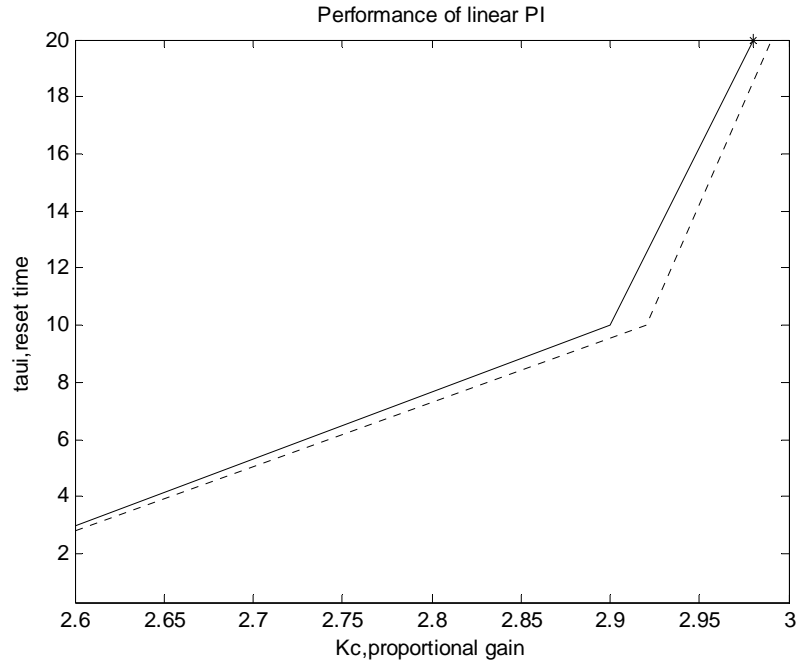


Figure 5.7a Performance region (to the left of the lines) of linear PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).

A more significant reduction in the conservatism of the design is observed when gain-scheduled PI controllers are designed. In this case, W_c and W_d are different from zero. The stability results are shown in Figure 5.8 in terms of the gain-scheduling weights W_c and W_d .

The results in Figure 5.8 were computed for a specific set of $K_c = 3.4$ and $\tau_I = 20$, which was selected on the stability boundary, shown as a star in Figure 5.6. The solid line corresponds to the fixed parameter Lyapunov function analysis, Theorem 4.1, whereas the dotted line is obtained using the variable parameter Lyapunov function design, Theorem 4.3. It is clear that the range of stability given by the region enclosed by the lines is much larger when the parameter-dependent Lyapunov function is used. Similar calculations were performed for robust performance and the results are shown in Figure 5.9. The results in Figure 5.9 were computed for a specific set of $K_c = 2.98$ and $\tau_I = 20$,

which was selected from Figure 5.7 on the performance boundary, shown as a star in Figure 5.7. The results indicate a clear improvement in the design, i.e., a larger performance region of parameter values defined by the dotted line, when the parameter-dependent Lyapunov function is used for analysis.

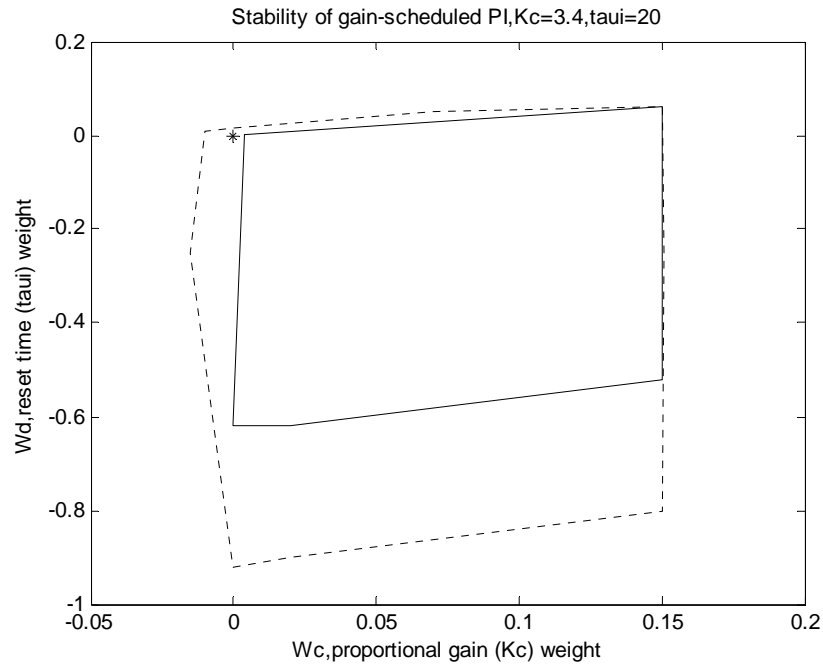


Figure 5.8 Stability region (inside the lines) of gain-scheduled PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).

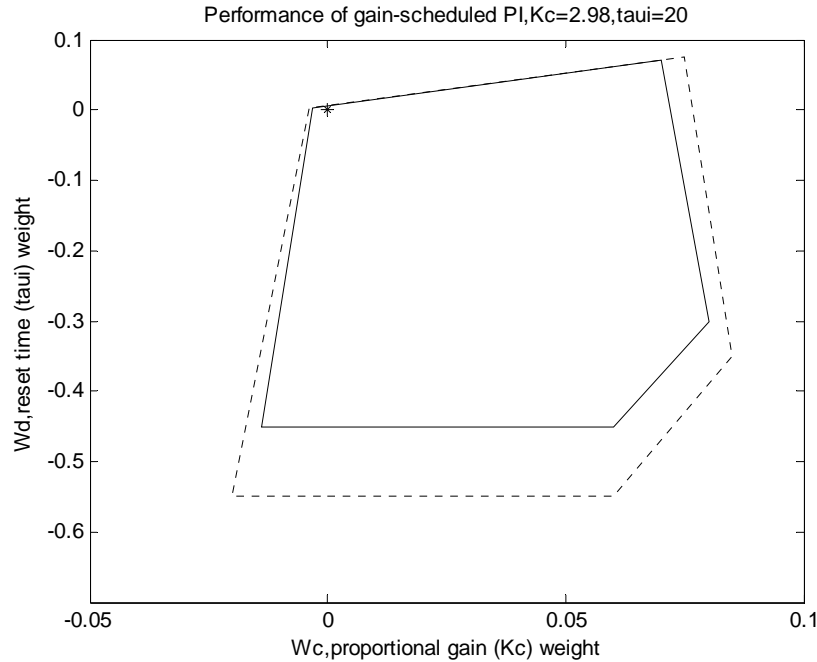


Figure 5.9 Performance region (inside the lines) of gain-scheduled PI controller parameters (comparing: fixed Lyapunov function (solid) and parameter-dependent Lyapunov function (dotted)).

5.4.2.2 Design based on the relaxation of the input-saturation factor ψ

In the previous sections, input-saturation was not accounted for. In this section, the effect of input-saturation and its bounds will be investigated. First, the two cases in terms of linear PI controller designs are compared. One case does not consider input-saturation, while the other case does. Second, Method 5.1 will be applied to calculate a less conservative value for $\underline{\psi}$ when saturation occurs.

First, the two cases of design results of linear PI controllers are compared. The first case without input-saturation has been considered in section 5.4.1.1, and the design results are given by the two lines in Figure 5.2. The second case considers the effect of input-saturation. Initially, the saturation factor ψ was assumed to be in the range of $[0,1]$. However, for this range of ψ , it was not possible to meet the robustness criteria for any

possible values of the gain-scheduled PI controller parameter set θ . To illustrate the effect of the lower bound of ψ , it was decided to test for an arbitrary lower limit $\underline{\psi}$. Later in this section, a more accurate lower limit of ψ was found analytically based on the Method 5.1 described earlier in this chapter. Following the same procedures described in section 5.4.1.1 to obtain the two lines in Figure 5.2, regions of linear PI controller tuning parameters in the parameter space are obtained when input-saturation is considered. The closed-loop system matrix is given by equation (5.15) which includes the input-saturation factor. The results are shown in Figure 5.10 and Figure 5.11.

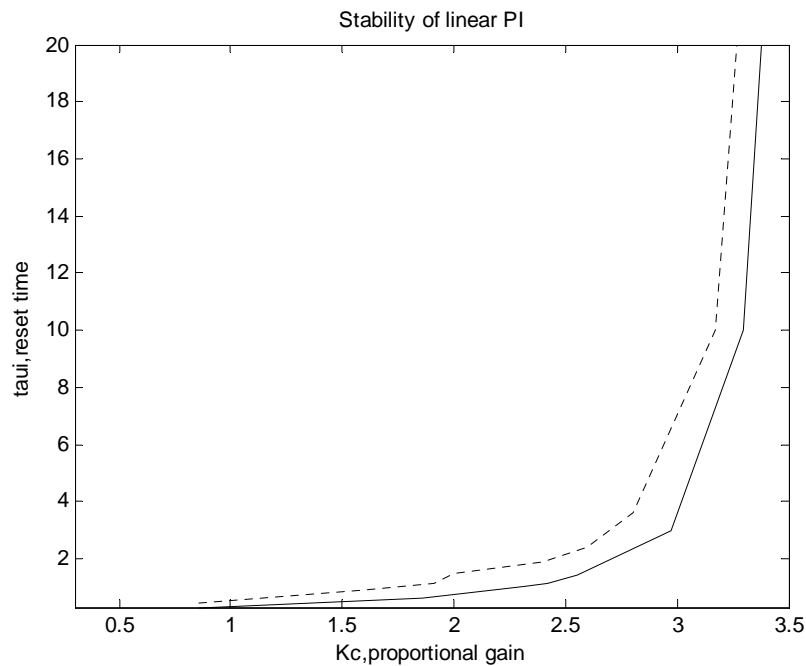


Figure 5.10 Stability regions (to the left of the lines) of linear PI controller parameters (comparing: without input-saturation (solid) and with $\psi \in [0.4 \ 1]$ (dotted)).

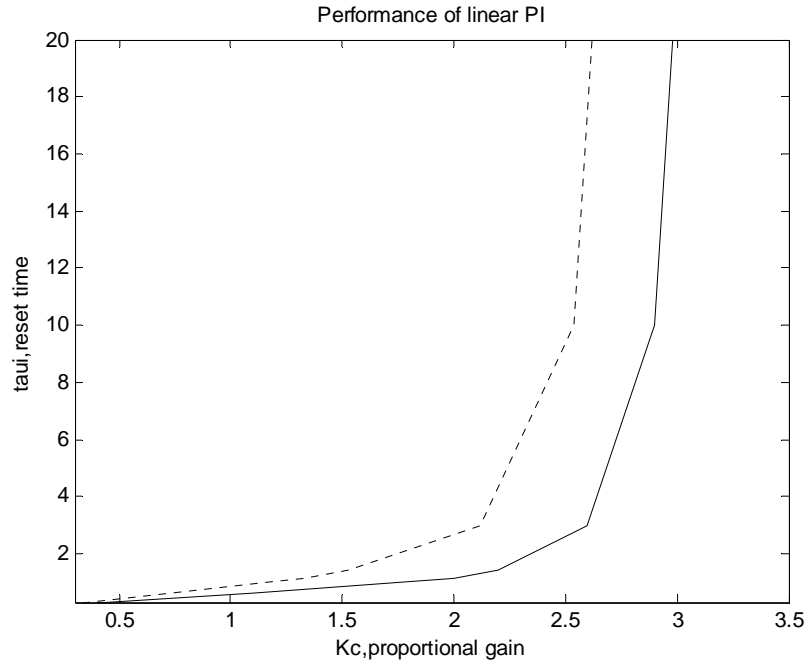


Figure 5.11 Performance regions (to the left of the lines) of linear PI controller parameters (comparing: without input-saturation (solid) and with $\psi \in [0.4 \ 1]$ (dotted)).

The results in Figure 5.10 and Figure 5.11 both clearly show the following: 1- the inclusion of input-saturation makes the stability and performance regions smaller, thus the designs are more conservative; and 2- by selecting the lower bound of ψ to be larger than zero, the robustness criteria can be met. Since the input-saturation problem cannot be ignored in practice, one possible solution to this problem is to obtain less conservative lower bounds of the input-saturation factor using the analytical method proposed in this work, i.e., Method 5.1, and hopefully less conservative designs will be obtained.

As a result, Method 5.1 is applied to calculate a less conservative value for $\underline{\psi}$. It is desired to obtain a saturation factor lower bound $\underline{\psi}$ which applies to all the tuning parameter combinations in the stability region given by the solid line in Figure 5.10. It is easy to obtain from equation (5.12) that for K_c values on the stability limit, $\underline{\psi}$ increases as τ_I increases. This means that the $\psi \in [\underline{\psi} \ \bar{\psi}]$ calculated along the stability limit will

include as well all the possible values of ψ inside the stability region shown in Figure 5.10, since in that region, τ_I is higher than the values on the boundary. Since the stability region is larger than the performance region, the bound on ψ obtained from the stability limit will surely apply also to the parameter sets in the robust performance region. The results obtained using Method 5.1 along the stability limit, the solid line in Figure 5.10, are summarized in Table 5.2.

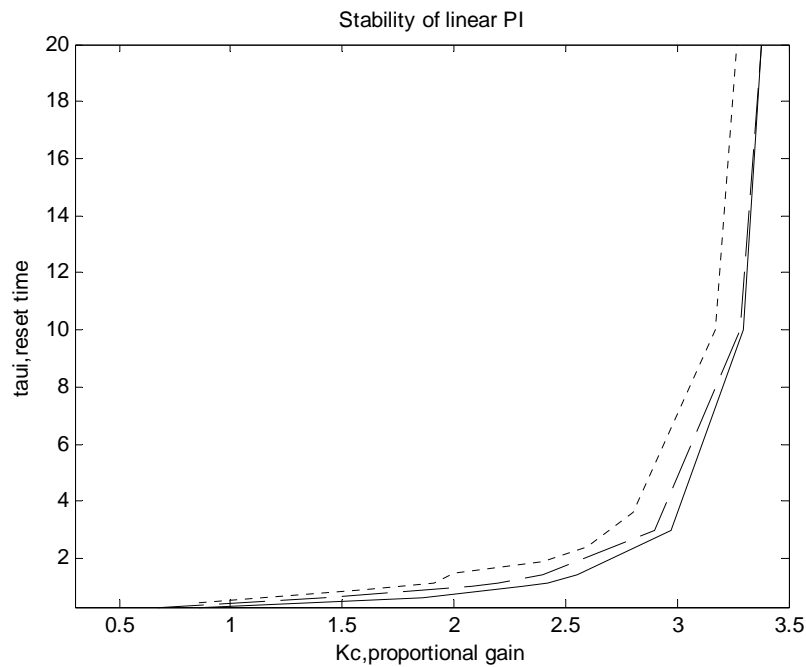


Figure 5.12 Stability regions (to the left of the lines) of linear PI controller parameters (comparing: without input-saturation (solid), with $\psi \in [0.4 \ 1]$ (dotted) and with $\psi \in [0.6203 \ 1]$ (dashed)).

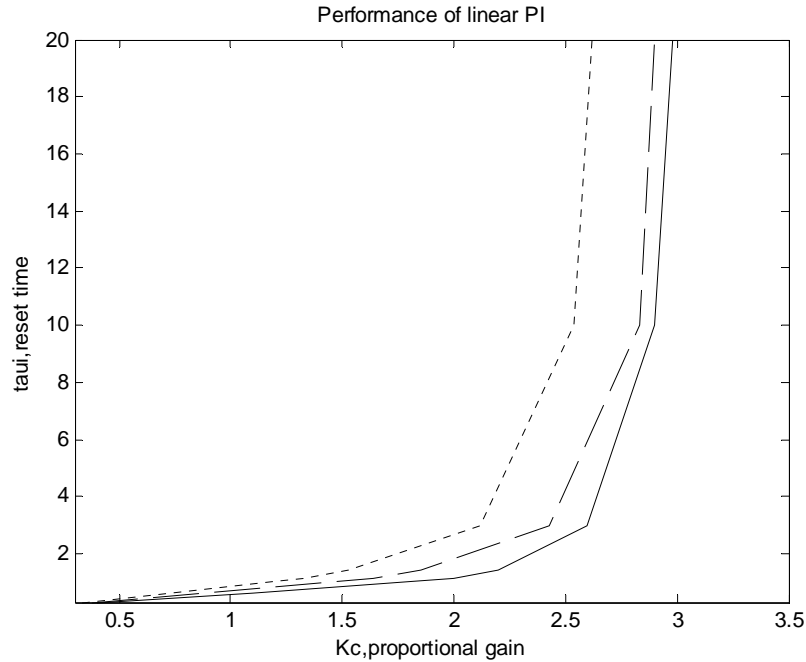


Figure 5.13 Performance regions (to the left of the lines) of linear PI controller parameters (comparing: without input-saturation (solid), with $\psi \in [0.4 \ 1]$ (dotted) and with $\psi \in [0.6203 \ 1]$ (dashed)).

Table 5.2 Input-saturation factor lower bound for controllers on stability limit (the solid line in Figure 5.10)

K_c	0.86	1.86	2.31	2.42	2.55	2.97	3.3	3.38
τ_I	0.3	0.6	1	1.1545	1.4	3	10	20
$\underline{\psi}$	0.7583	0.6203	0.7215	0.7562	0.8074	0.8418	0.8418	0.8418

Results obtained using Method 5.1 gives the less conservative bounds of the input-saturation factor, i.e., $\psi \in [0.6203 \ 1]$.

To show the effectiveness of reducing conservatism with this approach, the regions of linear PI controllers are obtained again with this new bounds of ψ , and compared with those shown by Figure 5.10 and Figure 5.11. The new boundaries are shown by dashed

lines in Figure 5.12 and Figure 5.13. The dashed lines in Figure 5.12 and Figure 5.13 are closer to the solid lines than those dotted lines are. This shows that bigger regions of robust stability and robust performance in terms of linear PI controller parameters can be obtained with the input-saturation factor bounds obtained using Method 5.1.

To illustrate the impact of the input-saturation limits on the design, the performance index $\gamma_{optimal}$ for different situations are computed and tabulated for comparison in Table 5.3.

Table 5.3 Relaxation of input-saturation factor bound and conservatism reduction

Case	$[K_c, \tau_I, W_c, W_d]$	ψ	$\delta_{1,t}$	$\delta_{2,t}$	$\gamma_{optimal}$
1	[1.3,2.5,0,0]	$\psi \in [0 \ 1]$	[-1,1]	[0,1]	∞
2	[1.3,2.5,0,0]	$\psi \in [0.6203 \ 1]$	[-1,1]	[0,1]	0.4702
3	[1.3,2.5,-0.05,0.01]	$\psi \in [0.8376 \ 1]$	[-1,1]	[0,1]	0.4280
4	[1.3,2.5,-0.05,0.01]	$\psi \in [1 \ 1]$	[-1,1]	[0,1]	0.3979
5	[1.3,2.5,-0.05,0.01]	$\psi \in [1 \ 1]$	[-0.4,0.4]	[0,0.16]	0.2846
6	[1.3,2.5,-0.05,0.01]	$\gamma_{simulation} = 0.2010$			

The case 1 in Table 5.3 corresponds to a fixed PI controller where the lower bound of ψ was assumed to be zero corresponding to a maximum control action of infinity according to equation (5.12). For this case the design results in unfeasible robust performance, i.e., an infinite value of the index $\gamma_{optimal}$. Case 2 corresponds to the same PI controller in case 1, but the lower bound of the saturation factor obtained from Method 5.1, i.e., $\psi \in [0.6203 \ 1]$, was used, and the robust performance was obtained with a finite value of $\gamma_{optimal} = 0.4702$.

Subsequently, a gain-scheduled PI controller was selected to show the conservatism reduction based on relaxation of the saturation factor bounds. This controller has the same $K_c = 1.3$, $\tau_I = 2.5$ as used in cases 1 and 2, and scheduling weights $W_c = -0.05$, $W_d = 0.01$. For this case, the lower bound of ψ was recalculated using Method 5.1 to be 0.8376, covered by the bound of $\psi \in [0.6203 \quad 1]$. The design results are given as case 3 in Table 5.3. This gain-scheduled controller of case 3 results in a better performance index than the fixed PI controller given in case 2.

Case 4 corresponds to the same gain-scheduled controller used in case 3 but without the input-saturation condition defined by equation (5.12). This was done to show the significant decrease in the value of $\gamma_{optimal}$ indicating that the saturation condition is a major contributor to the conservatism of the design. For the purpose of comparison, case 5 consists in the recalculation of case 4 with a smaller range of variation in $u(t)$, i.e., a smaller uncertainty range of $\delta_{i,t} \in [\underline{\delta}_i \quad \overline{\delta}_i]$. This results in an even smaller value of $\gamma_{optimal}$ of 0.2846.

Finally, to realistically assess the conservatism of the analysis, the system was numerically simulated for a large range of possible disturbances and the worst case was used to compute the $\gamma_{simulation}$ obtained in these simulations. This largest value was obtained from a two-consecutive-opposite-sign-pulse disturbance as shown in Figure 5.5. The simulation result $\gamma_{simulation} = 0.201$ of case 6 in Table 5.3, is smaller than the analysis results in cases 1-5. This indicates that some conservatism is inherent to any robust control analysis where many scenarios assumed in the robust analysis will not actually happen during closed-loop operation. If a smaller range of uncertainty was assumed in the analysis, as done in case 5, the analysis performance index γ will be closer to the simulated γ .

5.4.3 Design based on the SSV analysis

The state-affine model obtained in Chapter 3 is used in the design of the gain-scheduled PI controller in this section. A modeling error weight of $W_t = 0.025$ is included in the design and input-saturation is not considered. The SSV design is based on the two upper bound conditions (4.87) and (4.88), which are summarized in section 5.3.3. Since the modeling error is included, the closed-loop LFT frameworks are those obtained in section 5.3.2 and given by equations (5.31) and (5.33). The design results of SSV approach will be compared with those of quadratic Lyapunov approach obtained in section 5.4.1.1.

First, for linear PI controllers, following the same procedures described in section 5.4.1.1, regions of robust stability and performance in terms of the controller parameters are obtained and compared with the results from section 5.4.1.1 based on quadratic Lyapunov approach. The two design approaches gave almost the same design results, shown by the two lines in Figure 5.2.

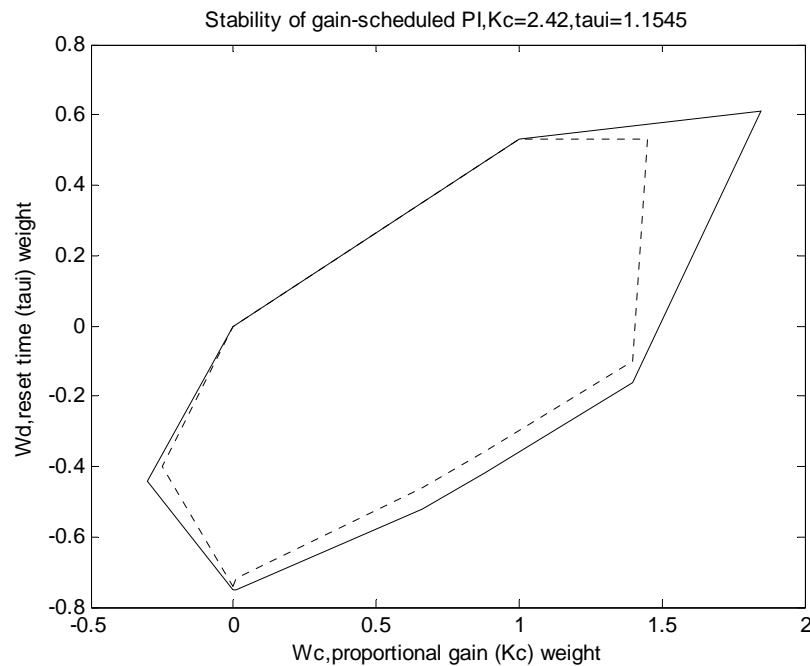


Figure 5.14 Stability region (inside the lines) of gain-scheduled PI controller parameters (comparing: quadratic Lyapunov approach (solid) and SSV approach (dotted)).

Second, gain-scheduled PI controllers are designed, for the same two linear PI controllers chosen as in section 5.4.1.1, i.e., $K_c = 2.42, \tau_I = 1.1545$ for robust stability and $K_c = 2, \tau_I = 1.1545$ for robust performance. The stability and performance limits in terms of the scheduling weights are shown in Figure 5.14 and Figure 5.15 as dotted lines and compared to the results obtained with the quadratic Lyapunov tests.

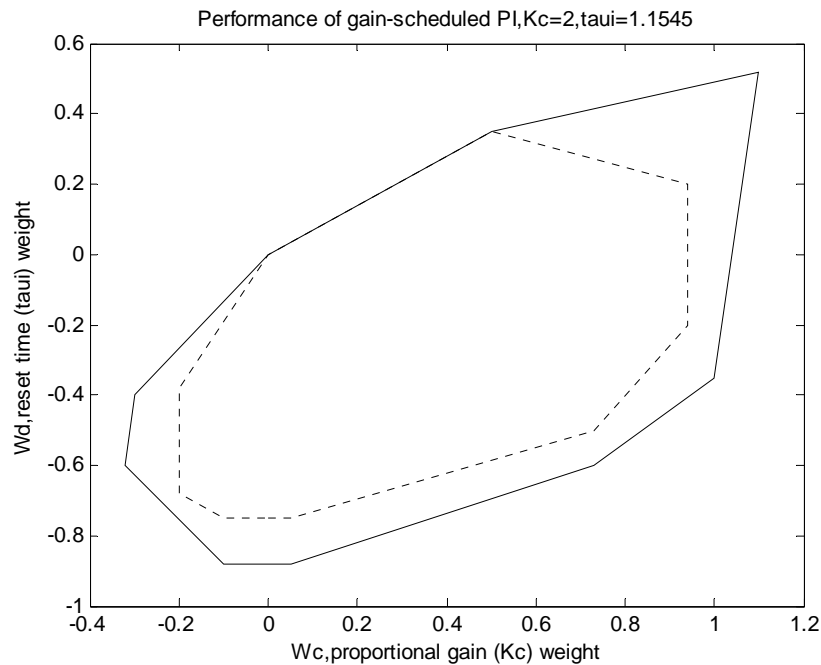


Figure 5.15 Performance region (inside the lines) of gain-scheduled PI controller parameters (comparing: quadratic Lyapunov approach (solid) and SSV approach (dotted)).

Figure 5.14 and Figure 5.15 show that the stability and performance regions obtained with the SSV approach are smaller than those obtained with the quadratic Lyapunov approach. This shows that the SSV approach is more conservative than the LMIs based quadratic tests. The reason for this conservatism is as follows.

SSV analysis is based on the upper bound of μ , and when the uncertainty structure has repeated scalar blocks, the upper bound of μ will not equal μ . In this case, the conclusions drawn from the upper bound of μ will be conservative, especially more conservative as the number of repeated scalars blocks increases. According to the closed-loop system formulation given by equations (5.31) and (5.33), the uncertainty structures have repeated scalars blocks for both the tests of robust stability and performance. The number of repeated scalars blocks is $k + 1$ for linear PI controllers design and $k + 2$ for gain-scheduled PI controllers design. Because the number of repeated scalars blocks for the design of gain-scheduled PI controllers is more than the number for linear PI controllers, the design results for gain-scheduled controllers are more conservative than the results for the design of linear PI controllers from SSV approach. As a result, the designs using SSV approach are more conservative than the results obtained from the quadratic Lyapunov approach. As the SSV approach is more conservative as explained above, it is not chosen as the main method in this work. However, it still remains a useful tool for robust control design of linear time-invariant systems.

5.5 Conclusions

A systematic approach has been proposed to design gain-scheduled PI controllers for nonlinear processes. It is based on the empirical state-affine models of the process that can be directly identified from process data. The proposed gain-scheduled PI controller contains small number of parameters, which facilitates the controller design. The designed robust gain-scheduled PI controllers guarantee robust stability and robust performance of the closed-loop system.

The linear PI controllers and gain-scheduled PI controllers can also be optimized to achieve near optimal performance based on a GEVP based optimization algorithm. The optimized controllers all showed improvement in terms of robust performance. Simulations showed that the gain-scheduled controller provided better performance than a linear PI controller tuned according to IMC rules. A performance index γ , although

conservative, has been found to be a reliable indicator of the relative performance of the different controllers considered in this work.

It was also shown that the designed robust controllers tend to be conservative, and thus, conservatism reduction has turned out to be an important emphasis of the current research. Two approaches have been proposed in this work to improve over the design of gain-scheduled PI controllers. The first approach is based on parameter-dependent Lyapunov functions and the second one is for the relaxation of the input-saturation factor.

For characterization of time-varying uncertain parameters, the rate of parameter variation is as important as the parameter range. When the bounds of the parameter and the rate of change are both available, it is desired to integrate them into the stability and performance analysis for less conservative designs. Parameter-dependent Lyapunov function takes into account parameter variation, and thus, it represents the general case of time-varying uncertain parameters, including the special case of constant uncertain parameters. Stability and performance tests have been developed based on it and the improvements were shown over the original design procedure. Design results based on these parameter-dependent Lyapunov functions showed that conservatism can be reduced, especially by a large amount, for the design of gain-scheduled PI controllers.

Relaxation of saturation factor lower bound is another approach proposed in this work to reduce the conservatism. The relaxation of this bound is made possible by the fact that controlled variables will have physical bounds due to process limits or sensor saturation. Simulation results on a CSTR process and comparison to the analysis results showed that this approach is very efficient in reducing the conservatism of the design.

Linear and gain-scheduled PI controllers have also been designed based on the SSV approach. The design results showed that the SSV approach is more conservative than the quadratic Lyapunov approach, especially for the design of gain-scheduled PI controllers. As it has been explained in Chapter 4, the SSV approach is generally more conservative than the quadratic Lyapunov approach for time-varying uncertainties. As a result, the

quadratic Lyapunov approach has been chosen as the main design approach in this work. Therefore, in the following chapter, the SSV design approach will not be used and only results obtained with the less conservative quadratic Lyapunov designs will be shown.

6 Robust Gain-scheduled MPC

In this chapter, a more general approach, Model Predictive Control (MPC), will be considered, where the process model is used to predict future outputs over a long time period. MPC is a widely accepted control algorithm in the chemical industry used for multivariate systems with constraints. The main purpose of this chapter is to present a new systematic approach to design robust gain-scheduled MPC controllers for nonlinear processes, which guarantee closed-loop robust stability and performance. This approach is based on the analysis tools presented in Chapter 4.

A gain-scheduled MPC controller scheduling on the process input for nonlinear chemical processes is proposed in this chapter. The state-affine model under this gain-scheduled MPC control results in a closed-loop system that can be shown to be an affine parameter-dependent model, with affine parameter-dependence on the process inputs. In Chapter 4, conditions on the robust stability and robust performance have been developed for this class of closed-loop system, i.e., affine parameter-dependent models. Based on these conditions, a robustness analysis is carried out to validate the design and obtain a series of input weights over the operation range according to the discretization, in the face of plant uncertainty.

The primary disadvantage of the design techniques in the literature for MPC is their inability to deal explicitly with plant uncertainty. In this chapter, a new approach for robust MPC synthesis is presented that allows explicit incorporation of the description of plant uncertainty in the problem formulation. The state-affine model is used to model the process output in the MPC optimization objective function, using the uncertainty description associated with the nonlinearity of the state-affine model. In this way, it is possible to account for the effect of model/plant mismatch and unmeasured disturbances. By using this approach, it is also possible to formulate the closed-loop system of the state-affine model together with a state-space form of the gain-scheduled MPC controller into a form suitable for robustness analysis.

The output predictions are done with step response models as for the linear case. However, to account for the process nonlinearity, instead of using one step response model, a family of step or equivalently impulse response models will be defined for different sub-ranges of values of the manipulated variable u . Then, for each of these models a linear MPC calculation can be conducted based on the current value of u . This approach results in a simple gain-scheduled MPC strategy that it somewhat resembles the traditional gain-scheduled approach based on local linearization. The key difference is that in this work, global closed-loop stability and performance will be tested instead of testing only the local closed loop stability and performance as proposed by practitioners for the traditional gain-scheduling approach.

In this approach, calculations are conducted offline to produce a sequence of optimal design tuning parameters for the MPC algorithm based on the values of the manipulated variable. Then, the resulting gain-scheduled MPC controller can be implemented on-line with the calculated tuning parameters scheduled based on the manipulated variable. The designed robust gain-scheduled MPC controller guarantees closed-loop system robust stability and performance.

In this chapter, section 6.1 reviews the traditional linear MPC (LMPC) based on step response models. Initially, the single-input-single-output (SISO) case, and the multi-input-multi-output (MIMO) version of MPC are presented. Then, the unconstrained MPC control law is formulated into a state-space form based on a straightforward matrix manipulation. Based on this state-space formulation, the closed-loop equations, composed of the state-affine model and the MPC controller, are formulated as an affine parameter-dependent system. The controller parameters can be tuned to achieve a desirable performance, and comments on their effects are given in section 6.2. The robust gain-scheduled design approach, similar to the one used in Chapter 5, is proposed to design MPC controllers based on the robust stability and performance conditions proposed in Chapter 4. The procedures for the design and optimization of robust gain-scheduled MPC are detailed in section 6.2. In section 6.3, the above proposed approach is

applied to the SISO CSTR case study and a simple 2x2 system, leading to a series of results and conclusions.

6.1 Unconstrained MPC Control Law

6.1.1 Model prediction based on step response models

Step response models are based on the following idea: for a linear time-invariant SISO system, assuming the system is at a rest, i.e., $\Delta u(t+i) = 0, i > 0$, the output change for the unit input change at $t-1$ is given by $y(t) = \{S_0^u, S_1^u, S_2^u, \dots, S_n^u, S_n^u, S_n^u, \dots\}$ or, by equation (6.1). Here it is assumed that the system settles after exactly n sampling steps. S_0^u is zero and this is because it is assumed that there is no immediate effect of the manipulated variable on the output. The step response of a process can also be shown by the following figure:

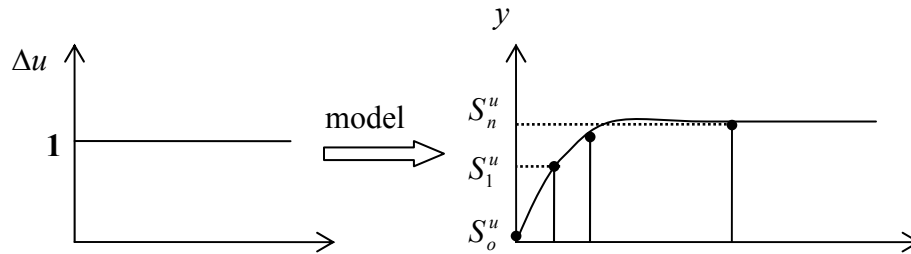


Figure 6.1 Step response of a process

$$y(t) = \sum_{i=1}^n S_i^u \Delta u(t-i) \quad (6.1)$$

$$\Delta u(t) = u(t) - u(t-1)$$

For continuous systems, the impulse response can be expressed as the first derivative of the step response. Equivalently, for a digital system with a zero-order hold the impulse response can be found by taking the first backward difference of the step response. The

unit impulse response coefficients of the process, $h_0, h_1, h_2, \dots, h_n$, are then given as follows:

$$h_0 = 0, h_i = S_i^u - S_{i-1}^u, i = 1, 2, \dots, n \quad (6.2)$$

and the discrete impulse response model using the impulse response coefficients is:

$$y(t) = \sum_{i=1}^n h_i u(t-i) \quad (6.3)$$

Equation (6.3) can be rearranged into the form of equation (6.1) by substituting the expression for h_i in (6.2) and then grouping terms for each S_i^u .

The step response model (6.1) can be made general to include an arbitrary number of output predictions into the future. Also, for a linear time-invariant system, using the superposition principle, the effect of the initial condition is added to the effect of the manipulated variable move on the response to obtain the overall response.

The prediction vectors are defined as follows

$$\begin{aligned} \bar{\mathbf{Y}}(t-1) &= [\bar{y}(t-1) \quad \bar{y}(t) \quad \dots \quad \bar{y}(t+n-3) \quad \bar{y}(t+n-2)]^T = \\ & [y(t-1) \quad y(t) \quad \dots \quad y(t+n-3) \quad y(t+n-2)]^T \\ & \text{(computed for } \Delta u(t+i) = 0, i \geq -1) \end{aligned} \quad (6.4)$$

$$\begin{aligned} \bar{\mathbf{Y}}(t) &= [\bar{y}(t) \quad \bar{y}(t+1) \quad \dots \quad \bar{y}(t+n-2) \quad \bar{y}(t+n-1)]^T = \\ & [y(t) \quad y(t+1) \quad \dots \quad y(t+n-2) \quad y(t+n-1)]^T \\ & \text{(computed for } \Delta u(t+i) = 0, i \geq 0) \end{aligned}$$

Then it is possible to find the effect of the initial conditions and the manipulated variables as follows:

1. Effect of the initial conditions:

For a linear time-invariant system, if $\Delta u(t+i) = 0, i \geq -1$, then

$$\bar{\mathbf{Y}}(t) = [\bar{y}(t) \quad \bar{y}(t+1) \quad \cdots \quad \bar{y}(t+n-2) \quad \bar{y}(t+n-1)]^T =$$

$$[y(t) \quad y(t+1) \quad \cdots \quad y(t+n-2) \quad y(t+n-2)]^T \text{ (computed for } \Delta u(t+i) = 0, i \geq -1)$$

or

$$\bar{\mathbf{Y}}(t) = \mathbf{M}_I \bar{\mathbf{Y}}(t-1) \quad (6.5)$$

where

$$\mathbf{M}_I = \begin{bmatrix} 0 & 1 & 0 & \cdot & 0 \\ 0 & 0 & 1 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 \\ 0 & 0 & \cdot & 0 & 1 \end{bmatrix}_{n \times n} \quad (6.6)$$

2. Effect of the manipulated variable move:

For the system at rest, $\Delta u(t-1) \neq 0$, the effect of the manipulated variable on the output is as follows:

$$\bar{\mathbf{Y}}(t) = \mathbf{s}^u \Delta u(t-1), \text{ where } \mathbf{s}^u = \begin{bmatrix} S_1^u \\ \vdots \\ S_n^u \end{bmatrix} \quad (6.7)$$

3. The combined effect of initial condition and manipulated variable move:

After adding the effect of the initial conditions and of the manipulated variable move, assuming there is no measured disturbance, the prediction vector is as follows:

$$\bar{\mathbf{Y}}(t) = \mathbf{M}_J \bar{\mathbf{Y}}(t-1) + \mathbf{s}^u \Delta u(t-1) \quad (6.8)$$

For the general case of an arbitrary sequence of m input changes, i.e., for $\Delta \mathbf{U}(t) = [\Delta u(t) \quad \Delta u(t+1) \quad \dots \quad \Delta u(t+m-1)]^T$, the prediction vector can be calculated using the following matrix equation

$$\bar{\mathbf{Y}}(t+1|t) = \mathbf{M}_J \bar{\mathbf{Y}}(t) + \mathbf{S}^u \Delta \mathbf{U}(t) \quad (6.9)$$

where the step response matrix \mathbf{S}^u is given as follows:

$$\mathbf{S}^u = \begin{bmatrix} S_1^u & 0 & 0 & \dots & 0 \\ S_2^u & S_1^u & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ S_n^u & S_{n-1}^u & \dots & \dots & S_{n-m+1}^u \end{bmatrix}_{n \times m} \quad (6.10)$$

Equation (6.9) gives the n -step-ahead prediction. In order to simplify computations, the prediction is generally performed over a prediction horizon $p, (p \leq n)$, which is then obtained simply by taking the first p rows of equation (6.9).

The above prediction given by equation (6.9) is an open-loop prediction, in the sense that it does not provide any corrections due to model errors or unmeasured disturbances that may have occurred at any previous time step. To address this shortcoming, a vector $\mathbf{W}(t+1|t)$ is defined to represent the unmeasured disturbance and model/plant mismatch. It is assumed that the disturbances are step-like, i.e., the current difference between the measurement and the prediction is applicable for any prediction into the future. $\mathbf{W}(t+1|t)$ is then given as follows:

$$\mathbf{W}(t+1|t) = \begin{bmatrix} w(t+1|t) \\ w(t+2|t) \\ \dots \\ w(t+p|t) \end{bmatrix} = \begin{bmatrix} w(t|t) \\ w(t|t) \\ \dots \\ w(t|t) \end{bmatrix} = \mathbf{N}_2[y(t) - \bar{y}(t)] \quad (6.11)$$

$$\mathbf{N}_2 = [1 \quad \dots \quad 1]_{1 \times p}^T$$

$y(t)$ is the new value of the measured output. The purpose of using this measurement at each time step is to compensate for unmeasured disturbances and model inaccuracy, both of which cause the system output to be different from the one predicted by the model.

The p – step-ahead prediction vector $\bar{\mathbf{Y}}(t+1|t)$, including the effect of modeling error and unmeasured disturbances, is then given as follows:

$$\bar{\mathbf{Y}}(t+1|t) = \mathbf{M}_p \bar{\mathbf{Y}}(t) + \mathbf{W}(t+1|t) + \mathbf{S}_p^u \Delta \mathbf{U}(t) \quad (6.12)$$

where \mathbf{S}_p^u is the sub-matrix made of the first p rows of \mathbf{S}^u , i.e., $\mathbf{S}_p^u = [\mathbf{I}_{p \times p} \quad 0]_{p \times n} \mathbf{S}^u$, and similarly $\mathbf{M}_p = [\mathbf{I}_{p \times p} \quad 0]_{p \times n} \mathbf{M}_I$. In equation (6.12), the first two terms are completely defined by past control actions and present measurements, and the last term describes the effect of future manipulated variable moves. $\bar{\mathbf{Y}}(t)$ is obtained from (6.8) and now it is referred to as the model update vector.

6.1.2 Unconstrained SISO MPC control law

The MPC control law can be easily explained by referring to Figure 6.2.

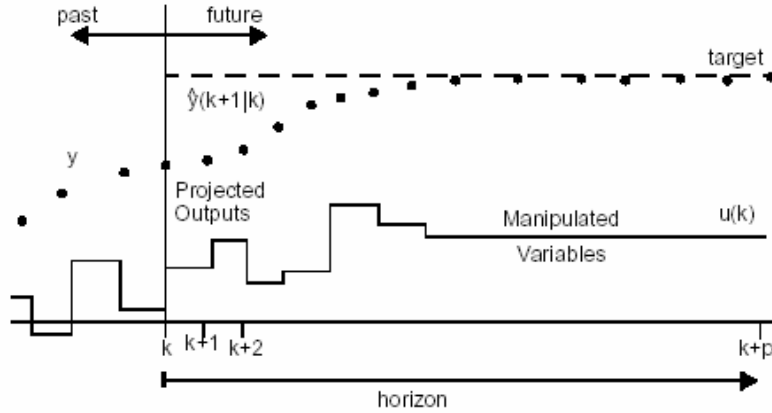


Figure 6.2 Model predictive control problem

The current time interval k in the above figure is denoted with t in the sequel. For any assumed set of present and future control moves $\Delta u(t), \Delta u(t+1), \dots, \Delta u(t+m-1)$, the future behavior of the process outputs $\bar{y}(t+1|t), \bar{y}(t+2|t), \dots, \bar{y}(t+p|t)$ can be predicted over a horizon p ($m \leq p$) using equation (6.12). Though m control moves are calculated, only the first one ($\Delta u(t)$) is actually implemented at time t . At the next sampling interval, new values of the measured output are obtained, the control horizon is shifted forward by one step, and the same computations are repeated. Hence, the resulting control law is referred to as “moving horizon” or “receding horizon.”

The control objective is to force the predictions $\bar{Y}(t+1|t)$ approach the set-point trajectory as closely as possible. The set-point trajectory, that is, the desired values of the set point p time steps into the future, is defined as $\mathbf{R}(t) = [r(t+1) \ r(t+2) \ \dots \ r(t+p)]^T$. Then, the unconstrained model predictive control problem consists of computing future control moves $\Delta \mathbf{U}(t) = [\Delta u(t) \ \Delta u(t+1) \ \dots \ \Delta u(t+m-1)]^T$ so that the future sum of squares errors between the output and target is minimized, i.e., MPC solves the following optimization problem:

$$\begin{aligned} & \min_{\Delta u(t)} \frac{1}{2} \left\{ \left\| \mathbf{\Gamma} [\bar{\mathbf{Y}}(t+1|t) - \mathbf{R}(t+1)] \right\|^2 + \left\| \mathbf{\Lambda} \Delta \mathbf{U} \right\|^2 \right\} \\ & s.t. \quad \bar{\mathbf{Y}}(t+1|t) = \mathbf{M}_p \bar{\mathbf{Y}}(t) + \mathbf{W}(t+1|t) + \mathbf{S}_p^u \Delta \mathbf{U}(t) \end{aligned} \quad (6.13)$$

where

$$\begin{aligned} \mathbf{\Lambda} &= \text{diag}\{\Lambda_1, \Lambda_2, \dots, \Lambda_m\} \\ \mathbf{\Gamma} &= \text{diag}\{\Gamma_1, \Gamma_2, \dots, \Gamma_p\} \end{aligned} \quad (6.14)$$

$\mathbf{\Lambda}, \mathbf{\Gamma}$ are positive-definite weighting matrices for u and y respectively at future time intervals. The $\mathbf{\Lambda}$'s weight's purpose is to penalize large moves in the manipulated variables, while the $\mathbf{\Gamma}$'s weight is used to penalize the errors according to the relative importance of the outputs in the problem under consideration. The parameters of $\mathbf{\Lambda}, \mathbf{\Gamma}$ are tuned to improve the performance of the model predictive controller.

Equation (6.13) can be solved via linear least squares algorithm. A solution that minimizes the sum of squares of the residuals of these equations, is given by $\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$, i.e.,

$$\Delta \mathbf{U}(t) = (\mathbf{S}_p^{uT} \mathbf{\Gamma}^T \mathbf{\Gamma} \mathbf{S}_p^u + \mathbf{\Lambda}^T \mathbf{\Lambda})^{-1} \mathbf{S}_p^{uT} \mathbf{\Gamma}^T \mathbf{\Gamma} \boldsymbol{\varepsilon}(t+1|t) \quad (6.15)$$

where $\boldsymbol{\varepsilon}(t+1|t) = [\varepsilon(t+1|t) \quad \varepsilon(t+2|t) \quad \dots \quad \varepsilon(t+p|t)]^T$ is the feedback corrected vector of future output deviations from the reference trajectory, assuming all present and future input moves $\Delta u(t), \Delta u(t+1), \dots, \Delta u(t+m-1)$ are zero. The solution of equation (6.15) gives a sequence of control moves, i.e., $\Delta \mathbf{U}(t) = [\Delta u(t) \quad \Delta u(t+1) \quad \dots \quad \Delta u(t+m-1)]^T$, but only the first control move $\Delta u(t)$ is implemented using equation:

$$u(t) = u(t-1) + \Delta u(t) \quad (6.16)$$

Then the calculations are redone at the next time step when new information about outputs and disturbances is available. Rewriting the above equations, the present control move $\Delta u(t)$ is obtained as follows

$$\begin{aligned} \Delta u(t) &= \mathbf{K}_{MPC} \boldsymbol{\varepsilon}(t+1|t) \\ \mathbf{K}_{MPC} &= [1 \quad 0 \quad \cdots \quad 0]_{1 \times m} (\mathbf{S}_p^{uT} \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \mathbf{S}_p^u + \boldsymbol{\Lambda}^T \boldsymbol{\Lambda})^{-1} \mathbf{S}_p^{uT} \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \end{aligned} \quad (6.17)$$

The matrix \mathbf{K}_{MPC} can be computed offline according to equation (6.17).

The algorithm is summarized as follows: given $\Delta u(t-1), y(t)$

1. Assume the system is at steady-state $y(0)$, initialize the model prediction vector at time $t = 0$ as

$$\bar{\mathbf{Y}}(0) = [y(0) \quad y(0) \quad \cdots \quad y(0) \quad y(0)]^T \quad (6.18)$$

2. Update the model according to equation (6.8)

$$\bar{\mathbf{Y}}(t) = \mathbf{M}_f \bar{\mathbf{Y}}(t-1) + \mathbf{s}^u \Delta u(t-1) \quad (6.8)$$

The first element of $\bar{\mathbf{Y}}(t)$, i.e., $\bar{y}(t)$, is the model prediction of the output at time t .

3. Compute the reference trajectory error vector

$$\boldsymbol{\varepsilon}(t+1|t) = \mathbf{R}(t+1) - \mathbf{M}_p \bar{\mathbf{Y}}(t) - \mathbf{W}(t+1|t) \quad (6.19)$$

where

$$\begin{aligned}\mathbf{W}(t+1|t) &= \mathbf{N}_2[y(t) - \bar{y}(t)] \\ \mathbf{N}_2 &= \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}_{1 \times p}^T\end{aligned}\quad (6.11)$$

4. Compute the current manipulated variable move

$$\begin{aligned}\Delta u(t) &= \mathbf{K}_{MPC} \boldsymbol{\varepsilon}(t+1|t) \\ \mathbf{K}_{MPC} &= \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}_{1 \times m} (\mathbf{S}_p^u T \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \mathbf{S}_p^u + \boldsymbol{\Lambda}^T \boldsymbol{\Lambda})^{-1} \mathbf{S}_p^u T \boldsymbol{\Gamma}^T \boldsymbol{\Gamma}\end{aligned}\quad (6.17)$$

$\Delta u(t)$ is implemented on the plant.

5. Go to step 2.

6.1.3 Unconstrained MIMO MPC control law

The results developed in the previous sections can be generalized to MIMO systems. For a system with n_u inputs and n_y outputs, the step response coefficient S_i^u in equation (6.1) would be

$$\mathbf{S}_i^u = \begin{bmatrix} S_{1,1,i}^u & S_{1,2,i}^u & \dots & S_{1,n_u,i}^u \\ S_{2,1,i}^u & \cdot & \cdot & \mathbf{0} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ S_{n_y,1,i}^u & S_{n_y,2,i}^u & \dots & S_{n_y,n_u,i}^u \end{bmatrix}_{n_y \times n_u} \quad (6.20)$$

where $S_{l,k,i}^u$ is the i^{th} step response coefficient describing the effect of k^{th} input on l^{th} output. The step response vector \mathbf{s}^u defined in equation (6.7) is then given by the following:

$$\mathbf{s}^u = \begin{bmatrix} \mathbf{S}_1^u \\ \vdots \\ \mathbf{S}_n^u \end{bmatrix} \quad (6.21)$$

and the step response matrices \mathbf{S}^u and \mathbf{S}_p^u defined in equations (6.10), are then given by the following:

$$\mathbf{S}^u = \begin{bmatrix} \mathbf{S}_1^u & 0 & 0 & \cdot & 0 \\ \mathbf{S}_2^u & \mathbf{S}_1^u & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{S}_n^u & \mathbf{S}_{n-1}^u & \cdot & \cdot & \mathbf{S}_{n-m+1}^u \end{bmatrix}_{mn_y \times mn_u} \quad (6.22)$$

$$\mathbf{S}_p^u = \begin{bmatrix} \mathbf{S}_1^u & 0 & 0 & \cdot & 0 \\ \mathbf{S}_2^u & \mathbf{S}_1^u & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{S}_p^u & \mathbf{S}_{p-1}^u & \cdot & \cdot & \mathbf{S}_{p-m+1}^u \end{bmatrix}_{pn_y \times mn_u}$$

For the multivariable case, the relevant variables are redefined as follows:

$$\bar{\mathbf{Y}}(t+1|t) = [\mathbf{y}^T(t) \quad \mathbf{y}^T(t+1) \quad \cdots \quad \mathbf{y}^T(t+n-2) \quad \mathbf{y}^T(t+n-1)]^T \quad (6.23)$$

$$\mathbf{y}(t) = [y_1(t) \quad y_2(t) \quad \cdots \quad y_{n_y-1}(t) \quad y_{n_y}(t)]^T$$

$$\mathbf{R}(t) = [\mathbf{r}^T(t+1) \quad \mathbf{r}^T(t+2) \quad \cdots \quad \mathbf{r}^T(t+p)]^T \quad (6.24)$$

$$\mathbf{r}(t+1) = [r_1(t+1) \quad r_2(t+1) \quad \cdots \quad r_{n_y}(t+1)]^T$$

$$\Delta \mathbf{U}(t) = [\Delta \mathbf{u}^T(t) \quad \Delta \mathbf{u}^T(t+1) \quad \cdots \quad \Delta \mathbf{u}^T(t+m-1)]^T \quad (6.25)$$

$$\Delta \mathbf{u}(t) = [\Delta u_1(t) \quad \Delta u_2(t) \quad \cdots \quad \Delta u_{n_u}(t)]^T$$

$$\bar{\mathbf{Y}}(t) = \mathbf{M}_I \bar{\mathbf{Y}}(t-1) + \mathbf{s}^u \Delta \mathbf{u}(t-1) \quad (6.26)$$

$$\boldsymbol{\varepsilon}(t+1|t) = \mathbf{R}(t+1) - \mathbf{M}_p \bar{\mathbf{Y}}(t) - \mathbf{W}(t+1|t) \quad (6.27)$$

$$\mathbf{M}_I = \begin{bmatrix} \mathbf{0}_{n_y \times n_y} & \mathbf{I}_{n_y} & \mathbf{0}_{n_y \times n_y} & \cdot & \mathbf{0}_{n_y \times n_y} \\ \mathbf{0}_{n_y \times n_y} & \mathbf{0}_{n_y \times n_y} & \mathbf{I}_{n_y} & \cdot & \mathbf{0}_{n_y \times n_y} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \mathbf{I}_{n_y} \\ \mathbf{0}_{n_y \times n_y} & \mathbf{0}_{n_y \times n_y} & \cdot & \mathbf{0}_{n_y \times n_y} & \mathbf{I}_{n_y} \end{bmatrix}_{n n_y \times m n_y} \quad (6.28)$$

$$\mathbf{M}_p = \begin{bmatrix} \mathbf{I}_{p n_y \times p n_y} & \mathbf{0} \end{bmatrix}_{p n_y \times m n_y} \mathbf{M}_I \quad (6.29)$$

$$\mathbf{W}(t+1|t) = \begin{bmatrix} \mathbf{w}(t+1|t) \\ \mathbf{w}(t+2|t) \\ \dots \\ \mathbf{w}(t+p|t) \end{bmatrix} = \begin{bmatrix} \mathbf{w}(t|t) \\ \mathbf{w}(t|t) \\ \dots \\ \mathbf{w}(t|t) \end{bmatrix} = \mathbf{N}_2 [\mathbf{y}(t) - \bar{\mathbf{y}}(t)] \quad (6.30)$$

$$\mathbf{N}_2 = \begin{bmatrix} \mathbf{I}_{n_y} & \cdot & \mathbf{I}_{n_y} \end{bmatrix}_{n_y \times p n_y}^T$$

$$\begin{aligned} \boldsymbol{\Lambda}_{m n_u \times m n_u} &= \text{diag}\{\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2, \dots, \boldsymbol{\Lambda}_m\}, \boldsymbol{\Lambda}_i = \text{diag}\{\Lambda_1, \Lambda_2, \dots, \Lambda_{n_u}\} \\ \boldsymbol{\Gamma}_{p n_y \times p n_y} &= \text{diag}\{\boldsymbol{\Gamma}_1, \boldsymbol{\Gamma}_2, \dots, \boldsymbol{\Gamma}_p\}, \boldsymbol{\Gamma}_i = \text{diag}\{\Gamma_1, \Gamma_2, \dots, \Gamma_{n_y}\} \end{aligned} \quad (6.31)$$

In analogy with the derivations for the SISO system, the MPC controller for a MIMO system can be solved with the algorithm in section 6.1.3, substituting all the vectors defined above in this section, and then the MPC controller's manipulated variable moves can be calculated as follows:

$$\begin{aligned} \Delta \mathbf{u}(t) &= \mathbf{K}_{MPC} \boldsymbol{\varepsilon}(t+1|t) \\ \mathbf{K}_{MPC} &= \begin{bmatrix} \mathbf{I}_{n_u} & \mathbf{0}_{n_u} & \dots & \mathbf{0}_{n_u} \end{bmatrix}_{n_u \times m n_u} (\mathbf{S}_p^u T \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \mathbf{S}_p^u + \boldsymbol{\Lambda}^T \boldsymbol{\Lambda})^{-1} \mathbf{S}_p^u T \boldsymbol{\Gamma}^T \boldsymbol{\Gamma} \end{aligned} \quad (6.32)$$

The model predictive control method can be advantageous for MIMO control problems when the process outputs exhibit dynamic interaction or when it is crucial to meet constraints on the manipulated and/or controlled variables. When constraints exist, a constrained optimization problem has to be solved to calculate the optimal control moves. However, in this work, as a preliminary study, only unconstrained case is considered.

6.1.4 Unconstrained MIMO MPC control law in state-space form

Since all LMIs based tests developed in Chapter 4 for robust stability and robust performance are only applicable to state-space formulation, a state-space version of the MPC controller is developed in this section.

Following Zanovello and Budman (1999) on SISO MPC controller, a MIMO state-space MPC controller representation can be obtained as follows. Assuming there is no measured disturbance, a controller state vector $\mathbf{U}(t-1)$ is defined as follows:

$$\mathbf{U}^T(t-1) = \begin{bmatrix} \mathbf{u}^T(t-1) & \mathbf{u}^T(t-2) & \dots & \mathbf{u}^T(t-n) \end{bmatrix}_{1 \times mn_u} \quad (6.33)$$

and the controller output $\mathbf{u}(t)$ is defined as follows:

$$\begin{aligned} \mathbf{u}(t) &= \mathbf{u}(t-1) + \Delta\mathbf{u}(t) \\ &= \mathbf{e}_1 \mathbf{U}(t-1) + \Delta\mathbf{u}(t) \end{aligned} \quad (6.34)$$

where the matrix \mathbf{e}_1 is given by:

$$\mathbf{e}_1 = \begin{bmatrix} \mathbf{I}_{n_u} & \mathbf{0}_{n_u \times n_u} & \dots & \mathbf{0}_{n_u \times n_u} \end{bmatrix}_{n_u \times mn_u} \quad (6.35)$$

Using the relation between the step response and the impulse response coefficients:

$$\mathbf{S}_{l,k,j} = \sum_{i=1}^j \mathbf{h}_{l,k,i}, j = 1, 2, \dots, n \quad (6.36)$$

where $\mathbf{h}_{l,k,i}$ is the i^{th} impulse response coefficient describing the effect of k^{th} input on l^{th} output, and assuming the system reaches steady-state in n sampling periods, the model update vector $\bar{\mathbf{Y}}(t)$ defined by equation (6.26) can also be calculated from the following vector equality:

$$\bar{\mathbf{Y}}(t) = \begin{bmatrix} \bar{\mathbf{y}}(t) \\ \bar{\mathbf{y}}(t+1) \\ \cdot \\ \cdot \\ \bar{\mathbf{y}}(t+n-1) \end{bmatrix} = \begin{bmatrix} \mathbf{S}_1^u & \mathbf{h}_2 & \cdot & \cdot & \mathbf{h}_n \\ \mathbf{S}_2^u & \mathbf{h}_3 & \cdot & \mathbf{h}_n & \mathbf{0}_{n_y \times n_u} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{S}_n^u & \mathbf{0}_{n_y \times n_u} & \cdot & \cdot & \mathbf{0}_{n_y \times n_u} \end{bmatrix} \begin{bmatrix} \mathbf{u}(t-1) \\ \mathbf{u}(t-2) \\ \cdot \\ \cdot \\ \mathbf{u}(t-n) \end{bmatrix} \quad (6.37)$$

or, in compact form:

$$\bar{\mathbf{Y}}(t) = \mathbf{H}_{n_y \times n n_u} \mathbf{U}(t-1) \quad (6.38)$$

where \mathbf{S}_i^u is defined in equation (6.20), and \mathbf{h}_i is defined as follows:

$$\mathbf{h}_i = \begin{bmatrix} h_{1,1,i} & h_{1,2,i} & \cdot & \cdot & h_{1,n_u,i} \\ h_{2,1,i} & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ h_{n_y,1,i} & h_{n_y,2,i} & \cdot & \cdot & h_{n_y,n_u,i} \end{bmatrix}_{n_y \times n_u} \quad (6.39)$$

Equations (6.26) and (6.37) can be shown to be equivalent. For example, consider the 2nd row of $\bar{\mathbf{Y}}(t)$ from equation (6.26):

$$\begin{aligned} \bar{\mathbf{y}}(t+1) &= \mathbf{y}(t+1), (\text{computed for } \Delta \mathbf{u}(t+i) = 0, i \geq 0) & (6.40) \\ \text{and } \Delta \mathbf{u}(t) = 0 &\Rightarrow \mathbf{u}(t) = \mathbf{u}(t-1) + \Delta \mathbf{u}(t) = \mathbf{u}(t-1) \end{aligned}$$

and, from equation (6.37):

$$\begin{aligned} \bar{\mathbf{y}}(t+1) &= \mathbf{S}_2^u \mathbf{u}(t-1) + \mathbf{h}_3 \mathbf{u}(t-2) + \cdots + \mathbf{h}_n \mathbf{u}(t-n+1) \xrightarrow{\mathbf{S}_2^u = \mathbf{h}_1 + \mathbf{h}_2} & (6.41) \\ &= \mathbf{h}_1 \mathbf{u}(t-1) + \mathbf{h}_2 \mathbf{u}(t-1) + \mathbf{h}_3 \mathbf{u}(t-2) + \cdots + \mathbf{h}_n \mathbf{u}(t-n+1) \xrightarrow{\mathbf{u}(t) = \mathbf{u}(t-1)} \\ &= \mathbf{h}_1 \mathbf{u}(t) + \mathbf{h}_2 \mathbf{u}(t-1) + \mathbf{h}_3 \mathbf{u}(t-2) + \cdots + \mathbf{h}_n \mathbf{u}(t-n+1) \xrightarrow{\mathbf{y}(t+1) = \sum_{i=1}^n \mathbf{h}_i \mathbf{u}(t+1-i)} \\ &= \mathbf{y}(t+1), (\text{computed for } \Delta \mathbf{u}(t+i) = 0, i \geq 0) \end{aligned}$$

Equations (6.40) and (6.41) are equivalent and this shows that the equations (6.26) and (6.37) are equivalent. Therefore, equation (6.37) will be used instead of equation (6.26) because it favors the following development of the MIMO state-space MPC controller.

The general case with unmeasured disturbances and/or error due to model/plant mismatch is considered, i.e., $\mathbf{W}(t+1/t)$ is different than zero. Then, assuming $\mathbf{R} = \mathbf{0}$ without loss of generality, the following is obtained from equations (6.33), (6.32) and (6.27):

$$\mathbf{U}(t) = \mathbf{T}_2 \mathbf{U}(t-1) + \mathbf{T}_1 \mathbf{K}_{MPC} [-\mathbf{M}_p \bar{\mathbf{Y}}(t) - \mathbf{W}(t+1|t)] \quad (6.42)$$

where the matrices $\mathbf{T}_1, \mathbf{T}_2$ are given as follows:

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{I}_{n_u} \\ \mathbf{0}_{n_u \times n_u} \\ \cdot \\ \cdot \\ \mathbf{0}_{n_u \times n_u} \end{bmatrix}_{m_u \times n_u} \quad \mathbf{T}_2 = \begin{bmatrix} \mathbf{I}_{n_u} & \mathbf{0}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \cdot & \mathbf{0}_{n_u \times n_u} \\ \mathbf{I}_{n_u} & \mathbf{0}_{n_u \times n_u} & \mathbf{0}_{n_u \times n_u} & \cdot & \mathbf{0}_{n_u \times n_u} \\ \mathbf{0}_{n_u \times n_u} & \mathbf{I}_{n_u} & \mathbf{0}_{n_u \times n_u} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{0}_{n_u \times n_u} & \cdot & \cdot & \mathbf{I}_{n_u} & \mathbf{0}_{n_u \times n_u} \end{bmatrix}_{nn_u \times nn_u} \quad (6.43)$$

As explained in the previous sections, the MPC controller depends on the current output measurement for the calculations of output predictions and control moves. This

requirement limits the application of the stability and performance analysis of the closed-loop system. Fortunately, the explicit dependence on the output measurement can be removed if a process model can be used to estimate the measured value of the output. The state-affine model used in the previous chapters in this work is also used in this chapter to represent the nonlinear process. This model has been shown before to account for the nonlinear behavior and time-varying dynamics of the process. The process output measurement will be approximated by the state-affine model output. Following a robust control approach, the state-affine model is made up of a nominal linear model plus some model error due to the nonlinear terms of the model.

The state-affine model of the process with unmeasured disturbance is as follows:

$$\begin{aligned}
\mathbf{x}(t+1) &= \{\mathbf{F}_0 + \sum_{i=1}^{K-1} \mathbf{F}_i \delta_{i,t}\} \mathbf{x}(t) + \{\mathbf{G}_1 + \sum_{i=1}^{K-1} \mathbf{G}_{i+1} \delta_{i,t}\} \mathbf{u}(t) & (6.44) \\
\mathbf{y}(t) &= \mathbf{H}_0 \mathbf{x}(t) + \mathbf{W}_f d(t) \\
d(t+1) &= BWd(t) + (1-BW)v(t) \\
\text{where } \delta_{i,t} &= u_{j_1}(t)^{k_1} u_{j_2}(t)^{k_2}, j_1 = 1, 2, \dots, n_u, k_1 = 0, 1, \dots, \\
& j_2 = 1, 2, \dots, n_u, k_2 = 0, 1, \dots, j_1 \neq j_2, k_1 + k_2 > 0
\end{aligned}$$

For a MIMO system with n_x states, n_u inputs and n_y outputs, the dimensions of the matrices in the above process model are $\mathbf{F}_i \in \mathfrak{R}^{n_x \times n_x}$, $\mathbf{G}_i \in \mathfrak{R}^{n_x \times n_u}$, $\mathbf{H}_i \in \mathfrak{R}^{n_y \times n_x}$ and $\mathbf{W}_f \in \mathfrak{R}^{n_y \times 1}$ respectively. The uncertain parameter $\delta_{i,t}$ is redefined as shown above to include cross-products between two different inputs. $\mathbf{y}(t)$ is the measured process output; thus the predicted output and the measured output are given respectively as follows:

$$\begin{aligned}
\bar{\mathbf{y}}(t) &= \begin{bmatrix} \mathbf{I}_{n_y} & \mathbf{0}_{n_y \times n_y} & \mathbf{0}_{n_y \times n_y} \end{bmatrix}_{n_y \times n_{ny}} \bar{\mathbf{Y}}(t) = \mathbf{e}_2 \mathbf{H} \mathbf{U}(t-1) & (6.45) \\
\text{and} & \\
\mathbf{y}(t) &= \mathbf{H}_0 \mathbf{x}(t) + \mathbf{W}_f d(t)
\end{aligned}$$

From equations (6.38), (6.30) and (6.45), equation (6.42) can be rewritten as:

$$\begin{aligned}
\mathbf{U}(t) &= \mathbf{T}_2 \mathbf{U}(t-1) + \mathbf{T}_1 \mathbf{K}_{MPC} \{-\mathbf{M}_p \bar{\mathbf{Y}}(t) - \mathbf{W}(t+1/t)\} \\
&= \mathbf{T}_2 \mathbf{U}(t-1) + \mathbf{T}_1 \mathbf{K}_{MPC} \{-\mathbf{M}_p \mathbf{H} \mathbf{U}(t-1) - \mathbf{N}_2 [\mathbf{y}(t) - \bar{\mathbf{y}}(t)]\} \\
&= \mathbf{T}_2 \mathbf{U}(t-1) + \mathbf{T}_1 \mathbf{K}_{MPC} \{-\mathbf{M}_p \mathbf{H} \mathbf{U}(t-1) - \mathbf{N}_2 [\mathbf{H}_0 \mathbf{x}(t) + \mathbf{W}_f d(t) - \mathbf{e}_2 \mathbf{H} \mathbf{U}(t-1)]\} \\
&= \mathbf{E}_2 \mathbf{U}(t-1) - \mathbf{T}_1 \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{H}_0 \mathbf{x}(t) - \mathbf{T}_1 \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{W}_f d(t)
\end{aligned} \tag{6.46}$$

where the matrix \mathbf{E}_2 is given as follows:

$$\mathbf{E}_2 = \mathbf{T}_2 - \mathbf{T}_1 \mathbf{K}_{MPC} \mathbf{M}_p \mathbf{H} + \mathbf{T}_1 \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{e}_2 \mathbf{H} \tag{6.47}$$

and the current control action can be calculated from the following expression:

$$\begin{aligned}
\mathbf{u}(t) &= \mathbf{u}(t-1) + \mathbf{K}_{MPC} [-\mathbf{M}_p \bar{\mathbf{Y}}(t) - \mathbf{W}(t+1/t)] \\
&= \mathbf{C}_{u1} \mathbf{U}(t-1) - \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{H}_0 \mathbf{x}(t) - \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{W}_f d(t)
\end{aligned} \tag{6.48}$$

where

$$\mathbf{C}_{u1} = [\mathbf{e}_1 - \mathbf{K}_{MPC} \mathbf{M}_p \mathbf{H} + \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{e}_2 \mathbf{H}] \tag{6.49}$$

Then, the state-space representation of the MPC controller can be found by combining equations (6.46) and (6.48) as follows:

$$\begin{bmatrix} \mathbf{U}(t) \\ \mathbf{u}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{E}_2 & -\mathbf{T}_1 \mathbf{K}_{mpc} \mathbf{N}_2 \\ \mathbf{C}_{u1} & -\mathbf{K}_{mpc} \mathbf{N}_2 \end{bmatrix} \begin{bmatrix} \mathbf{U}(t-1) \\ \mathbf{y}(t) \end{bmatrix} \tag{6.50}$$

From equations (6.44) and (6.50), the closed-loop system is obtained by combining the state-affine model and the MPC controller equations into the following equation:

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{U}(t) \\ \frac{d(t+1)}{\mathbf{y}(t)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}(\delta_t) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{U}(t-1) \\ \frac{d(t)}{v(t)} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ 1-BW \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{H}_0^T \\ \mathbf{0}^T \\ \mathbf{W}_f^T \end{bmatrix}^T, \mathbf{D} = [\mathbf{0}] \quad (6.51)$$

where

$$\mathbf{A}(\delta_t) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ -\mathbf{T}_1 \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{H}_0 & \mathbf{E}_2 & -\mathbf{T}_1 \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{W}_f \\ \mathbf{0} & \mathbf{0} & BW \end{bmatrix} \quad (6.52)$$

$$\mathbf{A}_{11} = (\mathbf{F}_0 + \sum_{i=1}^{K-1} \mathbf{F}_i \delta_{i,t}) - (\mathbf{G}_1 + \sum_{i=1}^{K-1} \mathbf{G}_{i+1} \delta_{i,t}) \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{H}_0$$

$$\mathbf{A}_{12} = (\mathbf{G}_1 + \sum_{i=1}^{K-1} \mathbf{G}_{i+1} \delta_{i,t}) \mathbf{C}_{u1}$$

$$\mathbf{A}_{13} = -(\mathbf{G}_1 + \sum_{i=1}^{K-1} \mathbf{G}_{i+1} \delta_{i,t}) \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{W}_f$$

The above state-space system representation can be used for robust performance analysis. For robust stability, the disturbance $d(t)$ does not have to be considered, so the system matrix \mathbf{A} to be considered for the robust stability analysis is as follows:

$$\mathbf{A}(\delta_t) = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ -\mathbf{T}_1 \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{H}_0 & \mathbf{E}_2 \end{bmatrix} \quad (6.53)$$

The closed-loop systems for robust stability analysis, given by equation (6.53) and for robust performance analysis, given by equations (6.51) and (6.52), both have affine-parameter dependence with respect to the uncertain parameters $\delta_{i,t}$'s. This allows the application of the robust stability and performance conditions developed in Chapter 4 to the design of MPC controllers given by equation (6.50).

6.2 Design and Optimization using Quadratic Lyapunov Functions

6.2.1 Selection of MPC tuning parameters

The MPC technique presented in the previous section includes a number of design parameters which can be adjusted to give the desired response as well as an appropriate amount of manipulated variable movement. For open-loop stable plants, stability and performance of the closed-loop system depends only on \mathbf{K}_{MPC} , which in turn is a function of the MPC design parameters, m , p , $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$, and step response coefficients. Systematic guidelines to select these parameters to obtain closed-loop stability are not available in the literature, but the following guidelines have been generally followed by practitioners:

1. The control horizon m is the number of future control actions that are calculated in the optimization step to reduce the predicted sum of squares errors. Too large a value of m results in excessive control action. A smaller value of m leads to a robust controller that is relatively insensitive to model errors. The parameter m is also the dimension of the matrix in equation (6.17) that must be inverted. Therefore, the computational effort increases as m is increased.
2. The parameter p is the number of future output predictions that are used in the optimization calculations. Increasing p results in more conservative control action which has a stabilizing effect but also increases the computational effort. In general, decreasing m relative to p makes the control action less aggressive and tends to stabilize a system.
3. The weighting matrices $\mathbf{\Lambda}$ and $\mathbf{\Gamma}$ contain a potentially large number of design parameters, possibly time-varying. However, for SISO systems, for simplicity it is possible to select $\mathbf{\Gamma} = \mathbf{I}$ and a diagonal $\mathbf{\Lambda} = \lambda \mathbf{I}$, with λ as a design parameter. Larger

values of Λ penalize the magnitude of the control moves $\Delta\mathbf{U}(t)$, thus resulting in less aggressive control.

In general practice, Λ is used as the main tuning parameter because its effect on the performance is straightforward. For SISO systems, it has been mentioned above that it is sufficient to select $\Lambda = \lambda\mathbf{I}$. For MIMO systems, this is equivalent to the following choice:

$$\Lambda_{m n_u \times m n_u} = \text{diag}\{\Lambda, \Lambda, \dots, \Lambda\}, \Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_{n_u}\} \quad (6.54)$$

For a system with n_u inputs, there are n_u design parameters $\lambda_1, \lambda_2, \dots, \lambda_{n_u}$. In this work, an approach for the design of Λ has been developed, based on the proposed robust stability and performance tests presented in Chapter 4. This will be discussed in detail in the following section.

6.2.2 Design and optimization of gain-scheduled MPC

The linear MPC (LMPC) algorithms are being widely used in industry because of their straightforward model representation, i.e., by using step or impulse response model directly identified from data. These advantages can be realized for nonlinear systems by modifying the linear algorithm. To understand the changes required, consider the effect of the system being nonlinear in the equations and variables used by LMPC.

As shown in the previous section, the MPC control calculation is based on output predictions obtained using a model. If the model is nonlinear, the prediction has to be calculated from a nonlinear function. Clearly, this will result in a nonlinear optimization problem that in many cases is difficult to solve.

In this work, to avoid the nonlinear optimization formulation, an alternative simpler approach for prediction and control calculation is proposed. It is proposed to do predictions with step response models as done for the linear case. However, to account

for the process nonlinearity, instead of using one step response model, a family of step or equivalently impulse response models will be defined for different sub-ranges based on the values of the manipulated variable u . Then, for each of these models a linear MPC calculation can be conducted based on the current value of u . This approach results in a simple gain-scheduled MPC strategy that somewhat resembles the traditional gain-scheduled approach based on local linearization. The key difference is that in this work, global closed-loop stability and performance will be tested instead of testing only the local closed-loop stability and performance as proposed by practitioners for the traditional gain-scheduling approach.

The impulse response models and consequently the control actions are scheduled with respect to the input \mathbf{u} . The input weight, for simplicity, will be assumed as the only tuning parameter scheduling against \mathbf{u} . For a SISO system, the overall range of change of the input variable $u(t)$ is discretized into k sub-ranges. For example, for the input variable over the range of $[-1, 1]$, two evenly split sub-ranges can be selected to be $u_1 = [-1, 0]$ and $u_2 = [0, 1]$. For a discretization into multiple sub-ranges $[u_j], j = 1, 2, \dots, k$, an MPC controller will be designed satisfying the robust stability and performance conditions for all the sub-ranges. Step responses and impulse responses are calculated for each sub-range, and $\mathbf{K}_{MPC}(u_j)$ will be obtained based on the optimization of the parameters of $\lambda(u_j)$ that composes the weight Λ .

This gain-scheduled MPC controller design approach can be also applied to the MIMO case. For a MIMO system with n_u inputs, the overall range of change of each input variable $u_i(t), i = 1, 2, \dots, n_u$ is discretized into $k_i, i = 1, 2, \dots, n_u$ sub-ranges. For example, for a 2×2 system, assuming the operation range of each input will be discretized into two sub-ranges, i.e., $k_i = 2$, the sub-ranges of u are as follows:

$$u_1 \in [-1, 1] \rightarrow \begin{cases} u_{1_r} = [-1, 0], r = 1 \\ u_{1_r} = [0, 1], r = 2 \end{cases}, u_2 \in [-1, 1] \rightarrow \begin{cases} u_{2_g} = [-1, 0], g = 1 \\ u_{2_g} = [0, 1], g = 2 \end{cases} \quad (6.55)$$

As a result, the whole operation range will be discretized into four sub-ranges, corresponding to the combinations of the input sub-ranges, i.e., $[r, g] = [1,1], [1,2], [2,1], [2,2]$, referred to as sub-ranges set rg in the sequel. Accordingly, the weighting matrix Λ penalizing the control moves is as follows

$$\Lambda_{4 \times 4} = \begin{bmatrix} \lambda_{1r} & & & \\ & \lambda_{2g} & & \\ & & \lambda_{1r} & \\ & & & \lambda_{2g} \end{bmatrix}, \begin{matrix} r = 1,2 \\ g = 1,2 \end{matrix} \quad (6.56)$$

Which contains 4 design parameters $\lambda_{11}, \lambda_{21}, \lambda_{12}, \lambda_{22}$, where r denotes the sub-range of operation related to the first input and g denotes the sub-range related to the second input. In general, for a MIMO system with n_u inputs, when each input is discretized into $k_i, i = 1, 2, \dots, n_u$ sub-ranges, the corresponding total number of design parameters λ 's will be $\sum_{i=1}^{n_u} k_i$. These parameters can be defined as a parameter vector $\theta = \{\lambda_{i,j}\}, i = 1, 2, \dots, n_u; j = 1, 2, \dots, k_i$. The impulse response model $\mathbf{H}(u)$ will be identified in each of the sub-ranges defined above.

In Chapter 4, it has been shown that for systems that can be represented in the form of equation (6.51), LMIs-based tests have been developed for the analysis of robust stability and performance. Based on these conditions, the robust gain-scheduled MPC controllers proposed in this section can be designed and optimized. The objective of this optimization problem is to minimize the parameter γ according to the following GEVP problem in MATLAB:

$$\begin{aligned}
& \min_{\mathbf{P}} \gamma^2 && (6.57) \\
& \text{subject to} && \begin{bmatrix} \mathbf{A}(\omega)^T \mathbf{P} \mathbf{A}(\omega) - \mathbf{P} & \mathbf{A}(\omega)^T \mathbf{P} \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P} \mathbf{A}(\omega) & \mathbf{B}^T \mathbf{P} \mathbf{B} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \gamma^2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
& \text{for all } \omega \in \mathbf{W}
\end{aligned}$$

which is an alternative form of the equation in Theorem 4.2.

To summarize, the procedure to design an optimal robust gain-scheduled MPC controller is as follows:

1. Set $k_i, i = 1, 2, \dots, n_u$ for each input. Select values of m, p . Set a range and a discrete grid of values in that range for the controller design parameters set $\boldsymbol{\theta}$.
2. Choose values for $\boldsymbol{\theta}$ according to the grid values within the parameter range.
3. Substitute values of $\boldsymbol{\theta}$ into the equation of Theorem 4.2.
4. Minimize γ subject to the equation of Theorem 4.2 (GEVP problem in MATLAB).
5. If a feasible solution exists for the above equation, accept values chosen in step 2 and the optimized performance index γ , otherwise, discard the current values.
6. Go to step 2 until $\gamma_{optimal} = \min(\gamma)$ is obtained over the whole parameter range.

The problem of searching for the optimal performance index, $\gamma_{optimal}$, is not quadratic in terms of the controller parameters $\lambda_{i,j}$'s and the optimization matrix variable \mathbf{P} simultaneously. Thus, the resulting problem is a nonlinear matrix inequality for all of these parameters. For example, equation (4.33) includes higher-order terms like

$\lambda_{i,j} \mathbf{P} \lambda_{i,j}, \lambda_{i,j} \mathbf{P} \lambda_{a,b}$. Thus, the optimization in terms of all of these parameters may be near optimal instead of a global optimal solution. Branch and bound methods have been proposed to solve LMIs that are not convex with respect to certain variables (Fukuda and Kojima, 2001; Braatz, VanAntwerp & Sahinidis, 1997). This is beyond the scope of the current study.

6.3 Case Study Results and Conclusions

For the SISO CSTR process, the state-affine model obtained in Chapter 3 is used in this section for the design of gain-scheduled MPC controllers in section 6.3.1. In section 6.3.2, a simple 2×2 process will be used as an illustration of the MIMO design.

6.3.1 Design results for SISO processes

6.3.1.1 Design results for the SISO CSTR process

For the state-affine model of the CSTR example, as a preliminary study, a simple gain-scheduled MPC controller will be designed according to the following even discretization of the scheduling variable, i.e., the manipulated variable u :

$$\begin{aligned} \text{for } -1 \leq u \leq 0 \quad & \text{MPC}_1(\mathbf{K}_{\text{MPC}_1}, \lambda_1) \\ \text{for } 0 < u \leq 1 \quad & \text{MPC}_2(\mathbf{K}_{\text{MPC}_2}, \lambda_2) \end{aligned} \quad (6.58)$$

In summary, the operation range of $u \in [-1,1]$ is discretized evenly into two sub-ranges, $-1 \leq u \leq 0$, and $0 < u \leq 1$. Then two MPC controllers are designed, one controller for each sub-range. If u is within the first sub-range, the controller $\text{MPC}_1(\mathbf{K}_{\text{MPC}_1}, \lambda_1)$ is applied, which is based on the step response coefficients corresponding to a step input changing from -1 to 0. If u is within the second sub-range, the controller $\text{MPC}_2(\mathbf{K}_{\text{MPC}_2}, \lambda_2)$ is applied, which is based on the step response coefficients corresponding to the step input changing from 0 to 1.

For one specific sub-range of u , the matrix \mathbf{A} of the closed-loop system, given by equation (6.53), has to be calculated with the λ value corresponding to the sub-range. As a result, over the whole operation range, a family of matrices \mathbf{A} will be obtained, each corresponding to one sub-range. The two bounds of the manipulated variable u for each sub-range, represent two vertices of the uncertain parameter box. The LMIs robust stability test of the closed-loop system will be checked against all the matrices \mathbf{A} at each vertex of the uncertain parameter box.

Additional vertices have to be added to account for the boundaries between the sub-ranges of \mathbf{u} . For example, for the controller given by equation (6.58), the vertices of $u = 0^+, 0^-$, using $\delta_i = u^i = 0$ in the state-affine model will be added. These vertices are necessary to account for the discontinuity of the controller at the discretization point 0. The discontinuity is due to the different values of λ and the step response from the two sides of 0. For example, when there is only one uncertain parameter $\delta_1 = u$, the parameter box without a controller discontinuity at 0 will be represented by the dotted line shown in Figure 6.3, while the parameter box that accounts for all the cases in the gain-scheduled MPC controller consists of the two triangles in Figure 6.3, which are outside the range of the dotted line. This parameter box is accounted for only when the two additional vertices corresponding to $u = 0^+, 0^-$ are considered. This is also true for the case when there is more than one uncertain parameter. Consequently, for the controller given by equation (6.58), all the vertices required for robust stability test are as follows:

$$\begin{aligned} \begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{U}(t) \end{bmatrix} &= \mathbf{A}(\mathbf{K}_{MPC1}, \lambda_1) \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{U}(t-1) \end{bmatrix}, \text{ for } u = 1, 0^+, 0 \\ \begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{U}(t) \end{bmatrix} &= \mathbf{A}(\mathbf{K}_{MPC2}, \lambda_2) \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{U}(t-1) \end{bmatrix}, \text{ for } u = -1, 0^- \end{aligned} \quad (6.59)$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ -\mathbf{T}_1 \mathbf{K}_{MPC} \mathbf{N}_2 \mathbf{H}_0 & \mathbf{E}_2 \end{bmatrix} \quad (6.53)$$

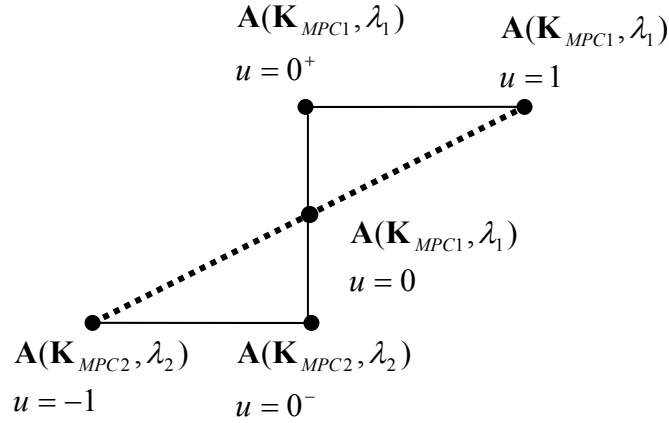


Figure 6.3 Vertices of the parameter box to be tested for robust stability

For the robust performance test of the closed-loop system, the system matrix \mathbf{A} given by equation (6.51) is used instead. This analysis can be easily extended to the case where a more complex scheduling with a larger number of sub-ranges will be used. In that case, additional vertices, corresponding to the connection point of every two adjacent sub-ranges, have to be added into the LMIs tests.

Table 6.1 Gain-scheduled MPC controller optimization (SISO)

GSMPC k	$[\lambda_1, \lambda_2, \dots, \lambda_k]$	$\gamma_{optimal}$
1	[0.7287]	0.5928
2	[0.2732, 0.9499]	0.4926
3	[0.3297, 0.8219, 1.1513]	0.4907
4	[0.8303, 0.8743, 0.8446, 1.0287]	0.6068
5	[0.9369, 1.0456, 1.0464, 1.0038, 1.0821]	0.6152

The gain-scheduled MPC controllers are referred to as GSMPC k in the sequel, where k indicates the number of sub-ranges specified for scheduling. A linear MPC is obtained

when $k = 1$. The optimal input weights $[\lambda_1, \lambda_2, \dots, \lambda_k]$ are obtained based on the optimization procedure described in the previous section and shown in Table 6.1. For different number of evenly discretized sub-ranges from 1 to 5, the optimal performance index $\gamma_{optimal}$ is also shown in the table.

The results for $\gamma_{optimal}$ in Table 6.1 are very close to each other among all the cases. Therefore, it is hard to make significant conclusions on what the best choice of k is. $\gamma_{optimal}$ decreases from $k = 1$ to $k = 3$, and then increases from $k = 3$ to $k = 5$. This shows that the system performance depends on the number of sub-range separations, and in this case, the GSMPC3 controller gives the best robust performance with a $\gamma_{optimal}$ of 0.4907. The reason that $\gamma_{optimal}$ increases from $k = 3$ to $k = 5$ is that more LMI terms and vertices are added to the problem thus increasing the problem conservatism. In addition, $\gamma_{optimal}$ also depends on what are the limits between the sub-ranges in terms of the values of the variable u . Some *a priori* knowledge about the process nonlinearity may be helpful to guide the separation, i.e., more sub-ranges are needed if the system in a particular operation range is highly nonlinear. This point is further explained in Chapter 7 as one of the future research directions.

Table 6.2 Gain-scheduled MPC controller simulation (SISO)

Controller name	$[\lambda_1, \lambda_2, \dots, \lambda_k]$	$\gamma_{optimal}$	$\gamma_{simulation}$	$\sum_{t=1}^{100} \mathbf{u}^T(t)\mathbf{u}(t)$
GSMPC5-1	[0.9369,1.0456,1.0464,1.0038,1.0821]	0.6152	0.3108	2.1345
GSMPC5-2	[5,5,5,5,5]	0.7230	0.3295	1.0426

To show the effect of the input weights optimization, one case is taken from the optimization results in Table 6.1, and compared to another case which has the same number of sub-ranges but non-optimized input weights. This comparison in Table 6.2 shows that the optimization of the input weights results in a reduction of the performance

index and thus results in a better robust performance. Simulation results of the two controllers are obtained and summarized in Table 6.2.

$\gamma_{simulation}$ is the performance index obtained from the simulation, calculated using $\|e\|_{L_2} = \gamma_{simulation} \|v\|_{L_2}$. Different disturbance signals were used in the simulations, including for example step signals, sinusoidal signals, white noise and combinations of them. A multi-spike disturbance signal was selected to be used in the following simulations, because it is resulting in the worst performance among different cases and the results are clear for comparison reasons. Then for the worst case found from simulation, $\gamma_{simulation}$ was calculated. The comparison in Table 6.2 shows that γ is a reliable performance index, in that $\gamma_{optimal} \geq \gamma_{simulation}$ for both controllers GSMPC5-1 and GSMPC5-2. Figure 6.4 shows the simulation results of the gain-scheduled MPC controllers for a specific disturbance signal shown in Figure 6.5.

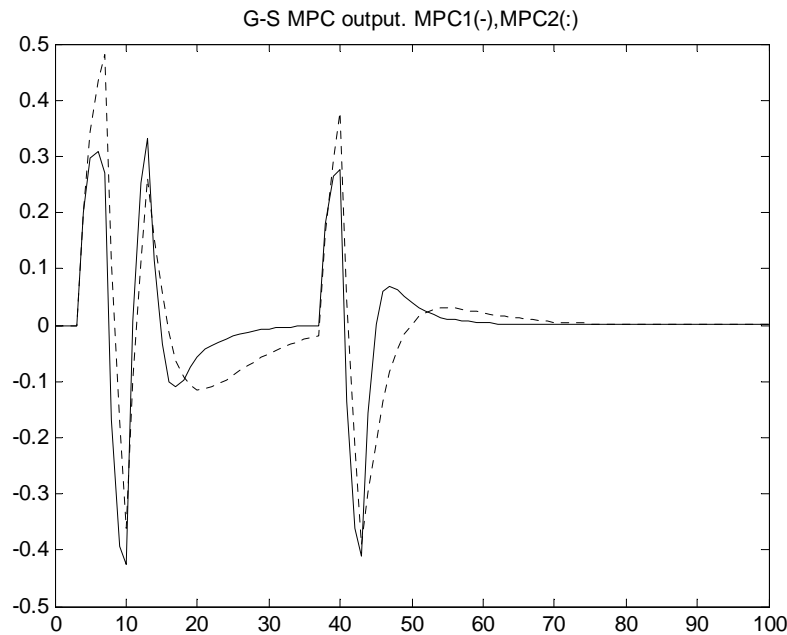


Figure 6.4 GSMPC5-1 (solid line) and GSMPC5-2 (dotted line) simulation

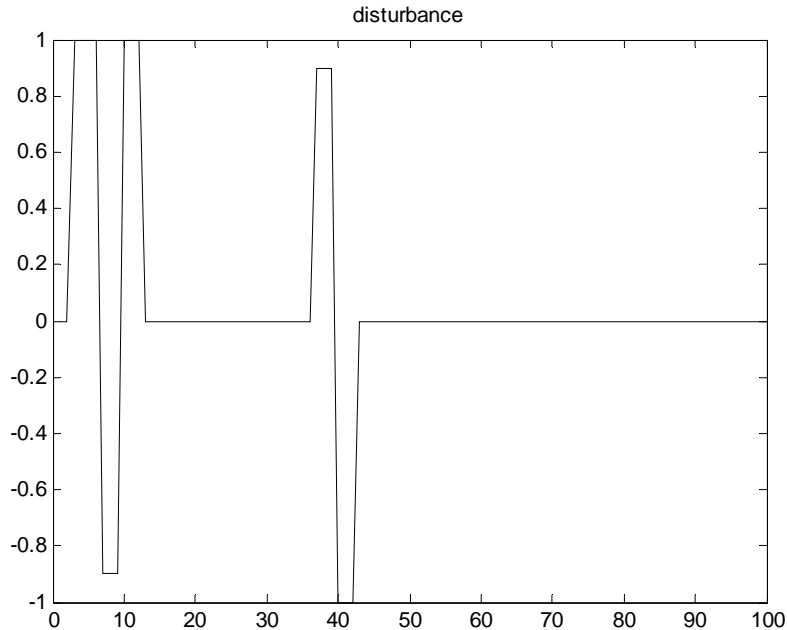


Figure 6.5 Disturbance signal used for the results in Table 6.2

6.3.1.2 Comparison of an optimal gain-scheduled MPC controller and an optimal gain-scheduled PI controller for the SISO CSTR process

For SISO processes, the robust performance analysis has been applied to design the optimal gain-scheduled MPC controllers, shown in Table 6.1. The optimal gain-scheduled MPC controller has turned out to be the GSMPC3 controller with a $\gamma_{optimal}$ of 0.4907. This controller is then compared to the optimized gain-scheduled PI controller 5-OPT-GS-PI-2, which has a $\gamma_{optimal}$ of 0.3204 from Chapter 5 and will be referred to as GSPI for simplicity in the sequel.

In the following, these two controllers are compared, in terms of analysis and simulation, and the comparison results are summarized in Table 6.3. The simulation results of both controllers, GSMPC3 and GSPI, are shown in Figure 6.6, and these results were obtained for the same disturbance as the one shown in Figure 6.5.

Table 6.3 Compare: Gain-scheduled PI and Gain-scheduled MPC

Controller	Parameters: θ	$\gamma_{optimal}$	$\gamma_{simulation}$	$\sum_{t=1}^{100} \mathbf{u}^T(t)\mathbf{u}(t)$
GSMPC3	$[\lambda_1, \dots, \lambda_k] = [0.3297, 0.8219, 1.1513]$	0.4907	0.2998	2.5524
GSPI	$[K_c, \tau_I, W_c, W_d]$ $= [1.2168, 1.9309, 0.1802, 0.0009]$	0.3204	0.2007	2.6360

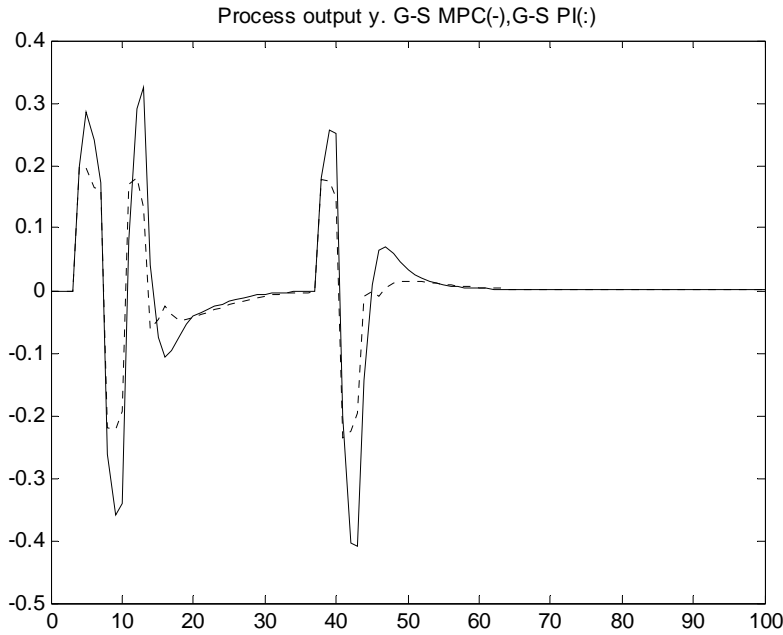


Figure 6.6 GSPI (dotted line) and GSMPC3 (solid line) simulation

The results in Figure 6.6 and Table 6.3 show that the GSPI controller gives better robust performance than the optimal GSMPC3 controller, with a slightly less aggressive control action. Both designs are based on the same robust performance condition proposed in Chapter 4, and the process model is also the same SISO state-affine model. The difference between the design results can only originate from the difference between the controller structures. The gain-scheduled PI controller given by equation (5.2) has only one controller state $\xi(t)$, and it is much simpler than the MPC controller given by equation (6.50), which has $n > 1$ states. As a result, the closed-loop system with the MPC controller, given by equations (6.51) and (6.52), is more complex than the one with PI

controller given by equations (5.14) and (5.15). The addition of controller states makes it more difficult for the robust performance condition to be met because the robust performance condition requires the stability of all the closed-loop system states, including the controller states. This increases the conservatism of the LMIs analysis for the gain-scheduled MPC controller. As a result, the difference between the analysis and simulation, i.e., the difference between $\gamma_{optimal}$ and $\gamma_{simulation}$, is bigger for the MPC controller than that for the PI controller.

This also shows the importance of reducing the conservatism of the robust performance analysis for the MPC controller design. However, these results should not be interpreted necessarily as to favor the PI controller over the MPC controller since the structure of these controllers and the uncertainty considered in the analysis are significantly different in the two cases, and the performance index is only an upper bound.

6.3.2 Design results for MIMO processes

In this section, the design of a gain-scheduled MPC controller for MIMO processes will be shown. To illustrate the technique, a simple 2-input-2-output example is used. The 2×2 state-affine model example is as follows:

$$\begin{aligned} \mathbf{x}(t+1) &= \{\mathbf{F}_0 + \mathbf{F}_1 \delta_{1,t}\} \mathbf{x}(t) + \{\mathbf{G}_1 + \mathbf{G}_2 \delta_{1,t} + \mathbf{G}_3 \delta_{2,t}\} \mathbf{u}(t) \\ d(t+1) &= BWd(t) + (1-BW)v(t) \\ \mathbf{y}(t) &= \mathbf{H}_0 \mathbf{x}(t) + \mathbf{W}_t d(t) \\ \text{where } \delta_{1,t} &= u_1(t), \delta_{2,t} = u_2(t) \end{aligned} \quad (6.60)$$

where the model coefficient matrices are as follows:

$$\mathbf{F}_0 = \begin{bmatrix} 0.1188 & -0.0346 \\ -2.3416 & 0.0937 \end{bmatrix}, \mathbf{F}_1 = \begin{bmatrix} 0.1076 & 0 \\ 1.2289 & 0 \end{bmatrix} \quad (6.61)$$

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{G}_2 = \begin{bmatrix} 0 & 0.1 \\ 0.1 & 0 \end{bmatrix}, \mathbf{G}_3 = \begin{bmatrix} -0.01 & -0.0159 \\ -0.0508 & -0.0928 \end{bmatrix}$$

$$\mathbf{H}_0 = \begin{bmatrix} 0.1755 & -0.0382 \\ 0 & 0.1 \end{bmatrix}, \mathbf{W}_f = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, BW = 0.8$$

First, a simple linear MPC controller will be designed, which has the following form:

$$\text{for } \begin{bmatrix} -1 \leq u_1 \leq 1 \\ -1 \leq u_2 \leq 1 \end{bmatrix} \quad MPC(\mathbf{K}_{MPC}, \mathbf{\Lambda}(\lambda_1, \lambda_2)) \quad (6.62)$$

Then, a simple gain-scheduled MPC controller will also be designed as follows:

$$\begin{aligned} \text{for } \begin{bmatrix} -1 \leq u_1 \leq 0 \\ -1 \leq u_2 \leq 0 \end{bmatrix} & \quad MPC_{11}(\mathbf{K}_{MPC11}, \mathbf{\Lambda}_{11}) & (6.63) \\ \text{for } \begin{bmatrix} -1 \leq u_1 \leq 0 \\ 0 < u_2 \leq 1 \end{bmatrix} & \quad MPC_{12}(\mathbf{K}_{MPC12}, \mathbf{\Lambda}_{12}) \\ \text{for } \begin{bmatrix} 0 < u_1 \leq 1 \\ -1 \leq u_2 \leq 0 \end{bmatrix} & \quad MPC_{21}(\mathbf{K}_{MPC21}, \mathbf{\Lambda}_{21}) \\ \text{for } \begin{bmatrix} 0 < u_1 \leq 1 \\ 0 < u_2 \leq 1 \end{bmatrix} & \quad MPC_{22}(\mathbf{K}_{MPC22}, \mathbf{\Lambda}_{22}) \end{aligned}$$

where $MPC_{ij}(\mathbf{K}_{MPCij}, \mathbf{\Lambda}_{ij})$ refers to the ij^{th} MPC controller, when u_1 is in its i^{th} sub-range, and u_2 is in its j^{th} sub-range. Then, according to equation (6.56), $\mathbf{\Lambda}_{ij}$ has the following form:

$$\mathbf{\Lambda}_{ij} = \begin{bmatrix} \lambda_{1i} & & & \\ & \lambda_{2j} & & \\ & & \lambda_{1i} & \\ & & & \lambda_{2j} \end{bmatrix}, \begin{matrix} i = 1,2 \\ j = 1,2 \end{matrix} \quad (6.64)$$

\mathbf{K}_{MPCij} will be calculated based on the step response corresponding to each of the sub-ranges defined by equation (6.63) using equation (6.32). For example, \mathbf{K}_{MPC12} will be calculated using step response corresponding to $[-1 \leq u_1 \leq 0, 0 < u_2 \leq 1]$.

The gain-scheduled MPC controller and the linear MPC controller are referred to as GSMPC-M and LMPC-M in the sequel. The robust performance analysis results of the controllers GSMPC-M and LMPC-M are summarized in Table 6.4. Assuming that the input weights are not optimized and set all equal to one, the $\gamma_{optimal}$ is calculated and tabulated in Table 6.4. If the input weights are not optimal, the linear MPC controller seems to provide a better robust performance.

Table 6.4 MPC controller analysis (comparing LMPC-M and GSMPC-M)

LMPC-M		
$\delta_1 : \omega_1 \in [\underline{\delta}_1, \bar{\delta}_1] \parallel \delta_2 : \omega_2 \in [\underline{\delta}_2, \bar{\delta}_2]$	$[\lambda_1, \lambda_2]$	$\gamma_{optimal}$
$\delta_1 : \omega_1 \in [-0.5, 0.5] \parallel \delta_2 : \omega_2 \in [-0.5, 0.5]$	$[1, 1]$	0.7698
GSMPC-M		
$\delta_1 : \begin{matrix} \omega_{11} \in [\underline{\delta}_{11}, \bar{\delta}_{11}] \\ \omega_{12} \in [\underline{\delta}_{12}, \bar{\delta}_{12}] \end{matrix} \parallel \delta_2 : \begin{matrix} \omega_{21} \in [\underline{\delta}_{21}, \bar{\delta}_{21}] \\ \omega_{22} \in [\underline{\delta}_{22}, \bar{\delta}_{22}] \end{matrix}$	$\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix}$	$\gamma_{optimal}$
$\delta_1 : \begin{matrix} \omega_{11} \in [-0.3, 0] \\ \omega_{12} \in [0, 0.3] \end{matrix} \parallel \delta_2 : \begin{matrix} \omega_{21} \in [-0.3, 0] \\ \omega_{22} \in [0, 0.3] \end{matrix}$	$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$	0.9350

Following the guidelines of Chapter 3, and according to the MIMO state-affine model given by equation (6.60), two uncertain parameters $\delta_{1,t} = u_1(t), \delta_{2,t} = u_2(t)$ are considered in this section. Table 6.4 shows the values of ω_1, ω_2 that denote the bounds of $u_1(t)$ and $u_2(t)$ respectively. In general, the uncertainty bounds will be described as follows:

$$\mathbf{W} := \{(\omega_1, \omega_2, \dots, \omega_n) : \omega_i \in [\underline{\delta}_i, \bar{\delta}_i]\} \quad (6.65)$$

The linear and gain-scheduled MPC controllers in Table 6.4 are simulated against a disturbance signal. The corresponding performance index $\gamma_{simulation}$, and the sum of squares of the control moves are summarized in Table 6.5.

Table 6.5 MPC controller simulation (comparing LMPC-M and GSMPC-M)

Controller name	$\gamma_{optimal}$	Simulation	
		$\gamma_{simulation}$	$\sum_{t=1}^{500} \mathbf{u}^T(t)\mathbf{u}(t)$
LMPC-M	0.7698	0.3523	22.0861
GSMPC-M	0.9350	0.3410	10.7434

The results in Table 6.5 show that there is inconsistency between the analysis results and the simulation results with respect to the comparison between the two controllers. The analysis shows that the LMPC-M controller is better than the GSMPC-M controller, because the worst-case performance index $\gamma_{optimal}$ of the LMPC-M controller is smaller. However, the simulation results show that the GSMPC-M controller gives a smaller performance index $\gamma_{simulation}$. The simulations have been carried out against a few different disturbance signals, and the worst results, i.e., the ones that gave the largest $\gamma_{simulation}$, are shown here. It should be remembered that the simulation results can not be exactly the same as the analysis results, because it is impossible to find the specific disturbance that achieves the worst-case performance index, and also, the computation of the theoretical performance bound assumes an infinite period of operation, which is also impossible to achieve with computer simulations.

It is believed that the inconsistency between the analysis and simulation is also partly because of the inherent conservatism of the robust analysis approach. The approaches proposed in Chapter 4, for example the analysis using the parameter-dependent Lyapunov functions, may be used in the future for the design of MPC controllers with the purpose

of reducing the conservatism. This is beyond the scope of the current work and is left for future research.

The controller simulation results are shown in Figure 6.7 and Figure 6.8. The two figures are showing the same results, only that Figure 6.7 shows part of the results for the period of $t = [1,100]$, and Figure 6.8 shows the results over the whole simulation period $t = [1,500]$. The disturbance signal used in this simulation is shown in Figure 6.9.

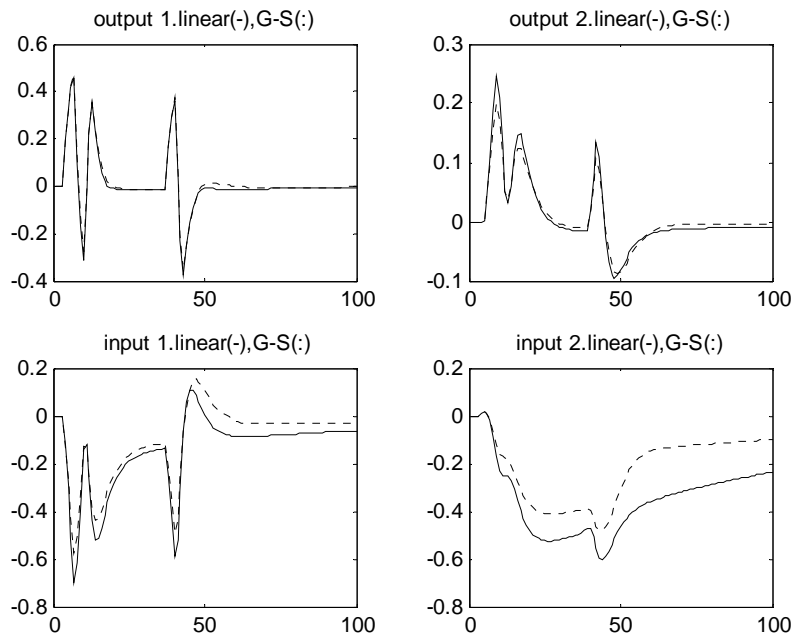


Figure 6.7 LMPC-M (solid line) and GSMPC-M (dotted line) simulation ($t = [1,100]$)

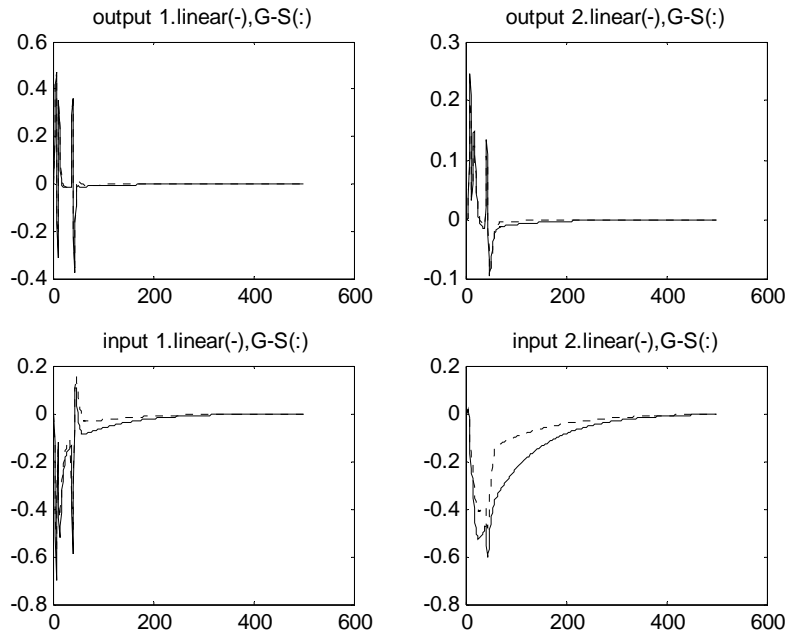


Figure 6.8 LMPC-M (solid line) and GSMPC-M (dotted line) simulation ($t = [1,500]$)

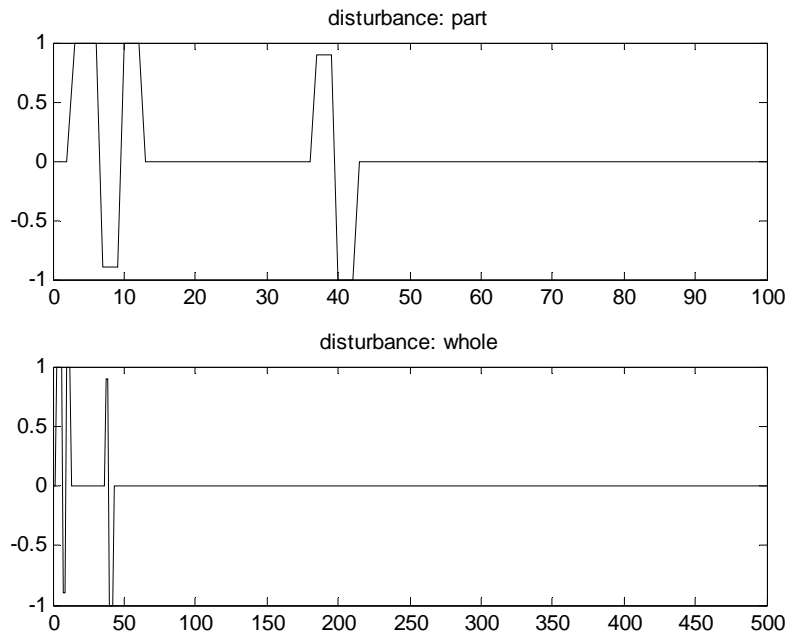


Figure 6.9 Disturbance signal used for the simulation

It is interesting to notice from Table 6.5 that the LMPC-M controller effort, represented by $\sum_{t=1}^{500} \mathbf{u}^T(t)\mathbf{u}(t)$, is much bigger (2.0558 times) than that of the GSMPC-M controller.

This can also be observed from the lower two plots in both Figure 6.7 and Figure 6.8, for the inputs u_1 and u_2 respectively. It means that the linear MPC controller may provide a slightly better output but at the cost of a large control effort. Large control actions are undesirable since they may imply large wear of actuators.

This also shows that the performance index $\gamma_{optimal}$ could be augmented by some term to reflect the effect of the controller effort in the robust performance analysis of the MPC controllers. This might also be a direction for future research which may lead to less conservative designs of controllers.

The input weights can be also optimized to improve the performance of the MPC controllers. For linear MPC controllers, optimized weights $[\lambda_1, \lambda_2]$ are obtained, and for gain-scheduled MPC controllers, optimized weights $\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix}$ can also be obtained.

The robust performance analysis shows that a smaller $\gamma_{optimal}$ is achieved for both the linear MPC controllers and the gain-scheduled MPC controllers.

The MPC controllers with these optimized weights are referred to as LMPC-M-OPT and GSMPC-M-OPT respectively. The optimized controllers LMPC-M-OPT and GSMPC-M-OPT are simulated against the same disturbance shown in Figure 6.9, which was used in the simulations of the LMPC-M and GSMPC-M controllers. The simulation results of the LMPC-M-OPT and GSMPC-M-OPT controllers are shown in Figure 6.10 and Figure 6.11. The analysis and simulation results of all the four MPC controllers for the MIMO process given by equation (6.60), are summarized in Table 6.6.

Table 6.6 MPC controller optimization

Linear MPC (equation (6.62))				
Controller name	$[\lambda_1, \lambda_2]$	$\gamma_{optimal}$	$\gamma_{simulation}$	$\sum_{t=1}^{500} \mathbf{u}^T(t)\mathbf{u}(t)$
LMPC-M	[1,1]	0.7698	0.3523	22.0861
LMPC-M-OPT	[0.5009,0.4983]	0.7472	0.3396	42.5782
Gain-scheduled MPC (equation (6.63))				
Controller name	$\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix}$	$\gamma_{optimal}$	$\gamma_{simulation}$	$\sum_{t=1}^{500} \mathbf{u}^T(t)\mathbf{u}(t)$
GSMPC-M	$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$	0.9350	0.3410	10.7434
GSMPC-M-OPT	$\begin{bmatrix} 0.5164 & 0.5029 \\ 0.4980 & 0.5034 \end{bmatrix}$	0.8694	0.3238	62.1820

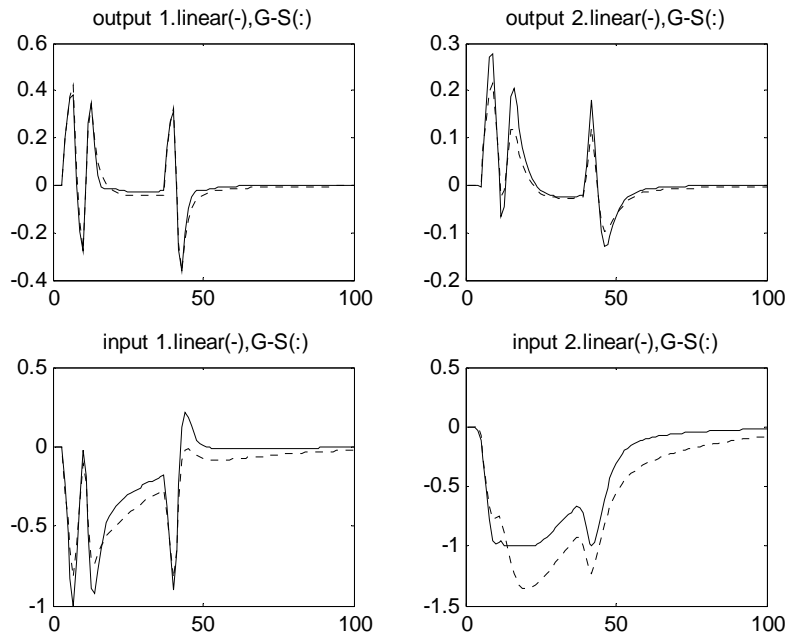


Figure 6.10 LMPC-M-OPT (solid line) and GSMPC-M-OPT (dotted line) simulation

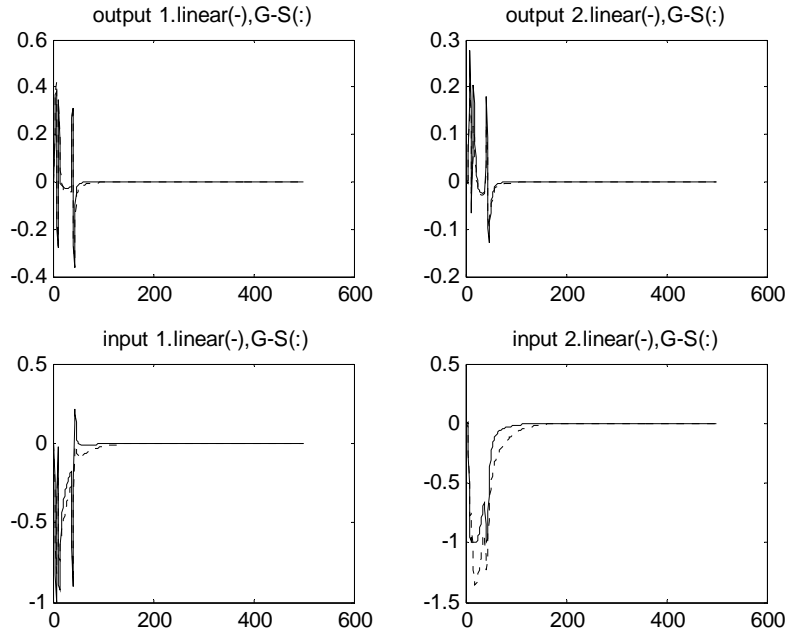


Figure 6.11 LMPC-M-OPT (solid line) and GSMPC-M-OPT (dotted line) simulation

It can be seen from Table 6.6 that the performance index $\gamma_{optimal}$ for gain-scheduled MPC controllers has been reduced by 7.02% from 0.9350 for GSMPC-M to 0.8494 for GSMPC-M-OPT, while the reduction for the linear case is only 2.94% from 0.7698 for LMPC-M to 0.7472 for LMPC-M-OPT. In summary, the optimization of the input weights has reduced the conservatism of the analysis by a larger amount for the gain-scheduled MPC controllers than for the linear MPC controllers. This is because there are four parameters to be optimized for the gain-scheduled case, which is twice the number of parameters to be optimized for the linear case.

The results in Table 6.6 also show that the performance index $\gamma_{optimal}$ for the LMPC-M controller is bigger than the $\gamma_{optimal}$ for the LMPC-M-OPT controller, and the performance index from simulation, i.e., $\gamma_{simulation}$, of the two controllers shows the same trend. This is also true for the other pair of controllers, i.e., GSMPC-M and GSMPC-M-OPT. As a result, it can be concluded that γ is a reliable performance index, in terms of differentiating robust performance among different controllers. γ showed some

inconsistency in Table 6.5 when it is compared among controllers which have different structures, i.e., a linear MPC controller and a gain-scheduled MPC controller.

In Table 6.6, the value $\sum_{t=1}^{500} \mathbf{u}^T(t)\mathbf{u}(t)$ of the optimized controllers are larger than the controllers with non-optimized input weights. This is because the optimized input weights are smaller compared to the non-optimized ones, for example, [0.5009,0.4983] smaller than [1,1] for the LMI-M-OPT controller. These smaller weights impose a smaller penalty on the controller moves than the weights of [1,1], and as a result, $\sum_{t=1}^{500} \mathbf{u}^T(t)\mathbf{u}(t)$ will increase. Smaller input weights will always result in more aggressive control.

6.3.3 Conclusions

In this chapter, the LMIs based robustness analysis techniques are applied to the design and optimization of gain-scheduled MPC controllers. For the application of the LMIs based approaches, the state-space formulation of the MPC controllers have been obtained in this work, based on the previous results obtained by Zanovello and Budman (1999). Design and optimization procedures for gain-scheduled MPC controllers are proposed in Section 6.2.2, and then all the techniques and approaches developed in this chapter are extended to MIMO processes. The optimal robust gain-scheduled MPC controllers are designed for both the SISO CSTR case study process and a simple 2x2 MIMO process. The analysis and simulations results of all the designed controllers are show in section 6.3.

First for the CSTR process, gain-scheduled MPC controllers are designed using different number of operation range discretization. The results in Table 6.1 show that the system performance depends on the number of sub-range separations and $\gamma_{optimal}$ also depends on what are the limits between the sub-ranges in terms of the values of the variable u . Some *a priori* knowledge about the process nonlinearity may be helpful to guide this

discretization step, i.e., more sub-ranges are needed if the system in a particular operation range is highly nonlinear. This point is further explained in Chapter 7 as one of the future research directions. The results in Table 6.2 show that the optimization of the input weights reduces the performance index and thus results in a better performance. It also shows that γ is a reliable performance index, since $\gamma_{optimal} \geq \gamma_{simulation}$ for both controllers GSMPC5-1 and GSMPC5-2.

The optimal gain-scheduled MPC controller and the optimal gain-scheduled PI controller are compared. The results in Figure 6.6 and Table 6.3 show that the GSPI controller gives better robust performance than the optimal GSMPC3 controller, with a less aggressive control action. This also indicates the importance of reducing the conservatism of the robust performance analysis for the MPC controller design.

For MIMO processes, a gain-scheduled MPC controller is designed for a simple 2x2 system. In this section, the operation range for each input is discretized evenly into two sub-ranges. If the input weights are not optimized, the linear MPC controller seems to provide a better robust performance. The simulation results also show that the linear MPC controller provides a slightly better output, but at the cost of a large undesirable control effort. This suggests that the performance index $\gamma_{optimal}$ could be augmented by some term to reflect the effect of the controller effort in the robust performance analysis of the MPC controllers.

If the input weights are optimized, the performance index from the analysis has been reduced by a larger amount for the gain-scheduled MPC controllers than for the linear MPC controllers. This is because there are four parameters to be optimized for the gain-scheduled case, which is twice the number of parameters to be optimized for the linear case.

In summary, the results in this chapter have shown that the procedures proposed here are efficient in obtaining linear and gain-scheduled MPC controllers, which guarantee closed-loop system's robust stability and performance. Optimization of the input weights

has improved the robust performance, showing the importance of optimal tuning of the MPC controller parameters. The simulation results show that γ is a reliable performance index, in terms of differentiating robust performance among different controllers of the same structure. Gain-scheduled MPC controllers are designed with a purpose to compensate for the process nonlinearity, and it is expected to provide a better performance than the linear MPC controllers for nonlinear processes. Most of the design results and all of the simulation results in this chapter show that the gain-scheduled MPC controllers achieve better performance at a cost of less control action. However, there are some counter examples for this point in the designed results. This may be due to the conservatism of the analysis, and as a result, a few directions for future research have been suggested to reduce the conservatism of the design, and they will be explained in detail in Chapter 7.

7 Conclusions and Future Work

7.1 Conclusions

Chemical or biochemical processes are highly nonlinear, especially when operated over a wide range of operating conditions. It is of a great significance to design high-performance nonlinear controllers for efficient control of these nonlinear processes to achieve closed-loop system's stability and good performance. However, there are not many general design procedures to deal with this task, and there are many difficulties to design such controllers because of the system nonlinearity.

For model-based control design problems for highly nonlinear processes, the first difficulty is to obtain a good simple model of the process under study. Two available options are first-principles models obtained from conservation equations, and empirical models identified from process input/output data. The nonlinearity of the processes is generally related to reaction kinetics or nonlinearity of physical properties, thus making it often difficult to obtain a first-principles model which can be used as the basis of control design task. As a result, in this work, relatively simple empirical models are chosen to represent the nonlinear process for the design of controllers.

First, a Volterra series model is identified using least squares algorithm from process input/output data. Then, a state-affine model that is nonlinear with respect to the manipulated variable is obtained through mathematical transformations of the Volterra series model coefficients, based on the algorithm proposed by Sontag (1978). Knapp and Budman (2000) have used this technique to design linear controllers for nonlinear processes. This state-affine model is especially suitable for robust control design since it can be easily partitioned into a nominal linear model and a nonlinear part. If the controller design is based on a nominal linear model, it is valid to assume that the model nonlinearity is the main source of model uncertainty, and the uncertainty can be directly quantified based on the information of process nonlinearity. Since the model nonlinearity is a function of the current manipulated variable only, the model uncertainty can be easily

quantified as compared to other studies in the literature where the uncertainty identification requires the solution of difficult optimization problems. The manipulated variable is naturally bounded due to, for example, actuator saturation limits. The problem of actuator saturation is also explicitly accounted for in Chapter 5 by defining an input-saturation factor and reformulating the gain-scheduled PI controller.

A state-affine model was identified for a CSTR case example, which gave the minimal sum of squares errors comparing to the real process output. The difference between the model output and the real process output is due to truncation of the Volterra series model and due to the transformation of the Volterra series model to the state-affine model. This modeling error has been effectively accounted for as an additional uncertain parameter in the design. This state-affine model was used throughout this work for the model-based robust control design of the CSTR example. The work summarized here is explained in detail in Chapter 3.

Since the state-affine models used in this work can be easily approximated by a nominal linear part and model uncertainty, robust control theory is a natural choice to analyze this type of models. Also the robust control approach is easier to apply and more general than a pure nonlinear analysis that relies on the finding of an appropriate Lyapunov function. Therefore, robust control theory has been applied for the design of gain-scheduled Proportional-Integral (PI) control in Chapter 5, and gain-scheduled Model Predictive Control (MPC) in Chapter 6.

The gain-scheduling formulations proposed in this work are different from the traditional ones reported in the literature. For example, the gain-scheduled PI controller parameters are changed as a continuous function of the scheduling variable, i.e., the manipulated variable, instead of switching these parameters at discrete values of the scheduling variable as generally proposed in the literature. Also, the traditional gain-scheduling (Bequette, 1997) approach is based on the analysis of local linear models, such that the overall designs cannot guarantee closed-loop system's global stability and performance (Shamma and Athans, 1990). The gain-scheduled controllers proposed in this work are all

designed based on the nonlinear state-affine model, which represents the nonlinear process over the whole operation range. The application of this model has also made it possible to incorporate model uncertainty into the gain-scheduled designs. To guarantee the global closed-loop system's stability and performance with the designed controllers, robustness analysis has been applied into the design approach.

The design procedure is based on the robust stability and performance conditions proposed in Chapter 4, a large part of the work shown in this chapter has been previously reported by Gao and Budman (2004). For time-varying uncertain parameters, robust stability and performance conditions are proposed in Theorems 4.1 and 4.2 using fixed Lyapunov functions, and in Theorems 4.3 and 4.4 using parameter-dependent Lyapunov functions. The results in Theorems 4.3 and 4.4 represent the main contributions of this work for discrete-time systems. The case of constant uncertain parameters is also summarized in Theorems 4.5 and 4.6. The comprehensive procedures for the design and optimization of robust gain-scheduled PI controllers are proposed in Chapter 5 and for MPC controllers in Chapter 6.

The closed-loop system modeled by combining the state-affine model and the controller is found to have an affine dependence on the uncertain parameters, and as a result, two important conclusions can be drawn for the closed-loop systems. The first conclusion is that all the uncertain parameters are valued in a convex parameter box, with the uncertainty bounds as the vertices of the box. The basis for this conclusion is found in the method for quantifying the model uncertainty from experimental data, shown in Chapter 3. The second conclusion is that each of the possible closed-loop system matrices within the uncertainty description is a fixed affine function of the uncertain parameters. These two conclusions reduce all the above robust stability and performance conditions proposed in Theorems 4.1 and 4.2, Theorems 4.3 and 4.4, which are originally an infinite set of Lyapunov inequalities, to a finite number of Linear Matrix Inequalities (LMIs). Thus, the final problems are numerically solvable. This is explained in Chapter 4.

One of the inherent problems with robust control is that the design is conservative. Two approaches have been proposed in this work to reduce the conservatism. The first one is based on parameter-dependent Lyapunov functions, and it is applied when the rate of change of the time-varying uncertainty parameters is available *a priori*. The robustness conditions based on this approach are summarized in Theorems 4.3 and 4.4. The second one is based on the relaxation of the lower bound of the input-saturation factor ψ defined in Chapter 5, to reduce the conservatism, which is proposed in Method 5.1.

For the case study CSTR example, gain-scheduled PI controllers were designed and optimized in Chapter 5. The design results are summarized in Chapter 5, which showed that the performance index γ is an efficient indicator for designing, comparing and optimizing the tuning parameters of gain-scheduled PI controllers. It was also found that the performance index $\gamma_{simulation}$ from the simulation is always significantly smaller than the $\gamma_{optimal}$ from the analysis for the designed controllers. For example, for the GS-PI-1 controller, $\gamma_{simulation}$ is 0.3495 while $\gamma_{optimal}$ is 0.5890. Based on these comparisons with simulations, the analysis has been found to be conservative to some degree.

To reduce the conservatism of the above design results, the first approach used in this work is based on parameter-dependent Lyapunov functions. Regions of robust stability and performance in the gain-scheduled PI controller parameter space have been obtained based on this approach in Chapter 5, and compared to the regions based on fixed-parameter Lyapunov functions. The results showed that the application of parameter-dependent Lyapunov functions has enlarged the design regions defined in terms of the tuning parameters. This reduction in conservatism was especially significant for the design of robust gain-scheduled PI controllers as compared to the design of linear PI controllers.

The second novel method that has been proposed in this work is the relaxation of the input-saturation factor ψ to reduce the conservatism. This approach has been very efficient in reducing the conservatism of the CSTR designs. For example, for the linear PI

controller with parameters of $K_c = 1.4, \tau_I = 2.5$, it is impossible to meet the robust performance condition without the relaxation of the input-saturation factor. That is, when $\underline{\psi} = 0$, $\gamma_{optimal} = \infty$. When the lower bound of the input-saturation factor was relaxed to be 0.8376 using Method 5.1, the performance index was reduced to a finite value of 0.4280. These results are given in Chapter 5.

For comparison, extensions of Structured Singular Value (SSV) approach have been reviewed and summarized in this work. In Chapter 4, the general procedures to obtain the Linear Fractional Transformation (LFT) for an uncertain system are given, and the robust stability and performance conditions are reviewed for time-varying uncertainties. Under the same set of conditions, the SSV approach is compared to the quadratic Lyapunov approach in the last section of Chapter 4. Case study results were obtained for the CSTR process in Chapter 5 and the results showed that the stability and performance regions obtained with the SSV approach are smaller than those obtained with the quadratic Lyapunov approach. SSV analysis is based on the upper bound of μ , and when the uncertainty structure has repeated scalar blocks, as is the case in this work, the upper bound of μ will not equal μ . Then, the conclusions drawn from the upper bound of μ will be conservative. In addition, the Lyapunov approach was found to be more versatile than the SSV approach to deal with the issues of input-saturation and for reducing conservatism by using parameter-dependent Lyapunov functions. As a result, the SSV approach is not pursued further beyond the basic comparisons described in Chapters 4 and 5.

The results in Chapters 3, 4 and 5 are obtained for SISO processes. In Chapter 6, the robustness analysis conditions developed in Chapter 4 are extended to MIMO processes. MPC controllers are designed instead of PI controllers, since MPC controllers are especially suitable to handle MIMO systems in the chemical industry. To compensate for the nonlinearity of the processes, gain-scheduled MPC controllers are designed, instead of the nonlinear MPC controllers reported in the literature based on nonlinear optimization. The operation range of the manipulated variable is discretized into a

number of sub-ranges. The controller tuning parameters and the step response matrix, are scheduled based on the input variable. This approach is different from the local-linearization design approach reported in the literature since the design in this work guarantees global stability and performance.

For the SISO CSTR example, gain-scheduled MPC controllers were designed based on different number of sub-ranges. Simulation results of these controllers show that the design procedure is effective because the designed controllers guarantee closed-loop system stability and performance in terms of disturbance rejection. The best gain-scheduled MPC controller (GSMPC3) from these designs, was compared to an optimal gain-scheduled PI controller (GSPI), and both the analysis and simulation results showed that GSMPC3 is more conservative than the GSPI controller. GSMPC3 has a $\gamma_{simulation}$ of 0.2998 and a $\gamma_{optimal}$ of 0.4907, while the GSPI controller has a smaller $\gamma_{simulation}$ of 0.2007 and a smaller $\gamma_{optimal}$ of 0.3204. First, these results show the importance of reducing the conservatism of the analytical approach. Second, these results should not be interpreted necessarily as to favor the PI controller over the MPC controller since the structure of these controllers and the uncertainty considered in the analysis are significantly different in the two cases, and the performance index is only an upper bound.

For a simple MIMO process with two inputs and two outputs, gain-scheduled MPC controllers were also designed. When the input weights were not optimized, the linear MPC controller showed a smaller $\gamma_{optimal}$ than the gain-scheduled MPC controller. However, the simulations of these two controllers showed that the gain-scheduled MPC controller showed a smaller $\gamma_{optimal}$ at only 50% of the linear controller's control effort. If the input weights are optimized, the performance index $\gamma_{optimal}$ for gain-scheduled MPC controllers has been reduced by 7.02%, while the reduction for the linear case is only 2.94%. In summary, the optimization of the input weights leads to improvement of the robust performance index by more than twice for the gain-scheduled MPC controllers as compared to linear MPC controllers. This is because there are four parameters to be

optimized for the gain-scheduled case, which is twice the number of parameters to be optimized for the linear case.

Gain-scheduled MPC controllers are designed with a purpose to compensate for the process nonlinearity, and it is expected to provide a better performance than the linear MPC controllers for nonlinear processes. Most of the design results and all of the simulation results in Chapter 6 show that the gain-scheduled MPC controllers achieve better performance at a cost of less control action. However, there are some counter examples for this point in the designed results. This may be due to the conservatism of the analysis, and as a result, a few directions for future research have been suggested to reduce the conservatism of the design. One example is that the analysis conservatism could be reduced by incorporating the control effort into the analysis.

For practical application of the proposed design procedures in industry, the complicated mathematical analysis could be carried out offline. The design results, i.e., the robust stability and performance regions in the controller parameter space, are produced by the procedures and can be applied online easily. In summary, the robustness analysis has been found to be efficient, but inherently conservative, and it is desired in the future to further reduce this conservatism, such that the design results will approach the simulation results. In the following section, a few future directions are suggested that focus mainly on reducing the conservatism of the analysis and design.

7.2 Future work

The main focus of the future work is suggested to be the reduction of the conservatism of the robustness analysis and the design based on the analysis. In Chapter 6, it has been shown by the design and simulation results that the LMIs based analysis for robust stability and performance is even more conservative for gain-scheduled MPC controllers than for the gain-scheduled PI controllers.

7.2.1 Reducing conservatism of the gain-scheduled MPC design

In Chapter 6, a gain-scheduled MPC controller design procedure has been proposed. The procedure consists of discretely scheduling the step response matrix and input weights based on discretization of the manipulated variable. In Chapter 6, it has been studied for the case that the whole operation range is discretized into even sub-ranges. However, the discretization of the operation range is not necessarily to be even, and the number of sub-ranges is itself a parameter that could be optimized. An MIMO process example with 2 inputs and 2 outputs will be used in this section to illustrate the importance of proper discretization of the scheduling variable for the design of gain-scheduled MPC controllers. Simulation results will be shown in this section

The 2×2 state-affine model is:

$$\begin{aligned} \mathbf{x}(t+1) &= \{\mathbf{F}_0 + \mathbf{F}_1 \delta_{1,t}\} \mathbf{x}(t) + \{\mathbf{G}_1 + \mathbf{G}_2 \delta_{1,t} + \mathbf{G}_3 \delta_{2,t}\} \mathbf{u}(t) \\ d(t+1) &= BWd(t) + (1-BW)v(t) \\ \mathbf{y}(t) &= \mathbf{H}_0 \mathbf{x}(t) + \mathbf{W}_r d(t) \\ \text{where } \delta_{1,t} &= u_1(t), \delta_{2,t} = u_2(t) \end{aligned} \quad (7.1)$$

where the model coefficient matrices are as follows:

$$\begin{aligned} \mathbf{F}_0 &= \begin{bmatrix} 0.1188 & -0.0346 \\ -2.3416 & 0.0937 \end{bmatrix}, \mathbf{F}_1 = \begin{bmatrix} 0.1076 & 0 \\ 1.2289 & 0 \end{bmatrix} \\ \mathbf{G}_1 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{G}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \mathbf{G}_3 = \mathbf{G}_2, \\ \mathbf{H}_0 &= \begin{bmatrix} 0.1755 & -0.0382 \\ 0.1755 & -0.0382 \end{bmatrix}, \mathbf{W}_r = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, BW = 0.8 \end{aligned} \quad (7.2)$$

The operation range for this process is defined to be $-1 \leq u_1 \leq 1, -1 \leq u_2 \leq 1$, which will be discretized into two sub-ranges at the middle limits a and b for u_1 and u_2 respectively. In Chapter 6, a similar gain-scheduled MPC controller was designed for the

case of $a = 0$ and $b = 0$ because an even discretization was used in that section. In this section, a and b will be designed to achieve better control performance, and the resulting gain-scheduled MPC controller is as follows:

$$\begin{aligned}
 & \text{for } \begin{bmatrix} -1 \leq u_1 \leq a \\ -1 \leq u_2 \leq b \end{bmatrix} \quad \text{MPC}_{11}(\mathbf{K}_{\text{MPC11}}, \mathbf{\Lambda}_{11}) & (7.3) \\
 & \text{for } \begin{bmatrix} -1 \leq u_1 \leq a \\ b < u_2 \leq 1 \end{bmatrix} \quad \text{MPC}_{12}(\mathbf{K}_{\text{MPC12}}, \mathbf{\Lambda}_{12}) \\
 & \text{for } \begin{bmatrix} a < u_1 \leq 1 \\ -1 \leq u_2 \leq b \end{bmatrix} \quad \text{MPC}_{21}(\mathbf{K}_{\text{MPC21}}, \mathbf{\Lambda}_{21}) \\
 & \text{for } \begin{bmatrix} a < u_1 \leq 1 \\ b < u_2 \leq 1 \end{bmatrix} \quad \text{MPC}_{22}(\mathbf{K}_{\text{MPC22}}, \mathbf{\Lambda}_{22})
 \end{aligned}$$

where $\text{MPC}_{ij}(\mathbf{K}_{\text{MPCij}}, \mathbf{\Lambda}_{ij})$ refers to the ij^{th} MPC controller, when u_1 is in its i^{th} sub-range, and u_2 is in its j^{th} sub-range. Then, the input weight matrix $\mathbf{\Lambda}_{ij}$ has the following form:

$$\mathbf{\Lambda}_{ij} = \begin{bmatrix} \lambda_{1i} & & & \\ & \lambda_{2j} & & \\ & & \lambda_{1i} & \\ & & & \lambda_{2j} \end{bmatrix} \begin{matrix} i = 1,2 \\ j = 1,2 \end{matrix} \quad (7.4)$$

$\mathbf{K}_{\text{MPCij}}$ will be calculated based on the step response corresponding to each of the sub-ranges defined by equation (6.63). For example, $\mathbf{K}_{\text{MPC12}}$ will be calculated using step responses corresponding to $[-1 \leq u_1 \leq a, b < u_2 \leq 1]$. In this section, a and b will be designed based on the step response of the process given by equations (7.1) and (7.2). It has been found from simulations that the step responses from input u_2 to both outputs change abruptly at the point of $b = -0.8$. The step responses corresponding to $a = 0$ and $b = -0.8$ are shown in Figure 7.1 and Figure 7.2. Based on this observation, it will be

chosen that $a = 0$ and $b = -0.8$ for the gain-scheduled MPC controller in the sequel, and this MPC controller, given by equation (6.63), will be referred to as GS-MPC-1.

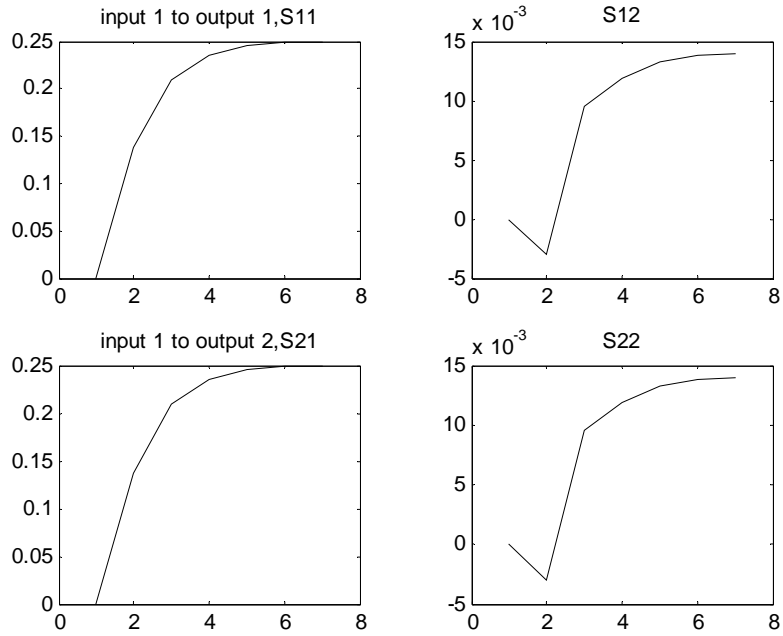


Figure 7.1 Step response ($u_1 \in [0,1], u_2 \in [-0.8,1]$)

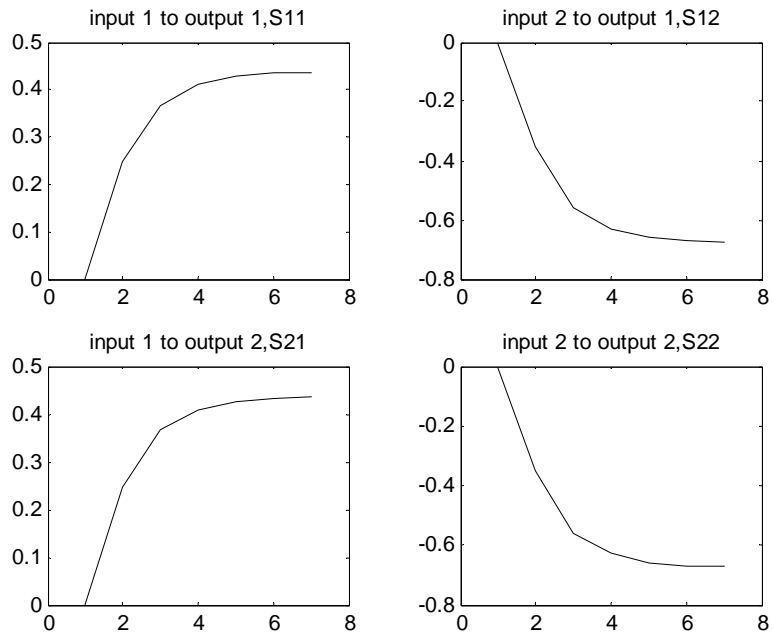


Figure 7.2 Step response ($u_1 \in [-1,0], u_2 \in [-1,-0.8]$)

For the purpose of comparison between even discretization and proper designed discretization, a second gain-scheduled MPC controller based on $a = 0$ and $b = 0$ is also simulated, and this MPC controller, given by equation (6.63), will be referred to as GS-MPC-2. The simulation results are shown in Figure 7.3 with the solid line corresponds to the GS-MPC-1 controller and dotted line to the GS-MPC-2 controller. The two controllers have the same parameter's values, for example the input weights are equal to one, and are simulated against the same disturbance, shown in Figure 7.4. In Figure 7.3, the upper two plots show the process outputs, and the lower two plots show the corresponding inputs.

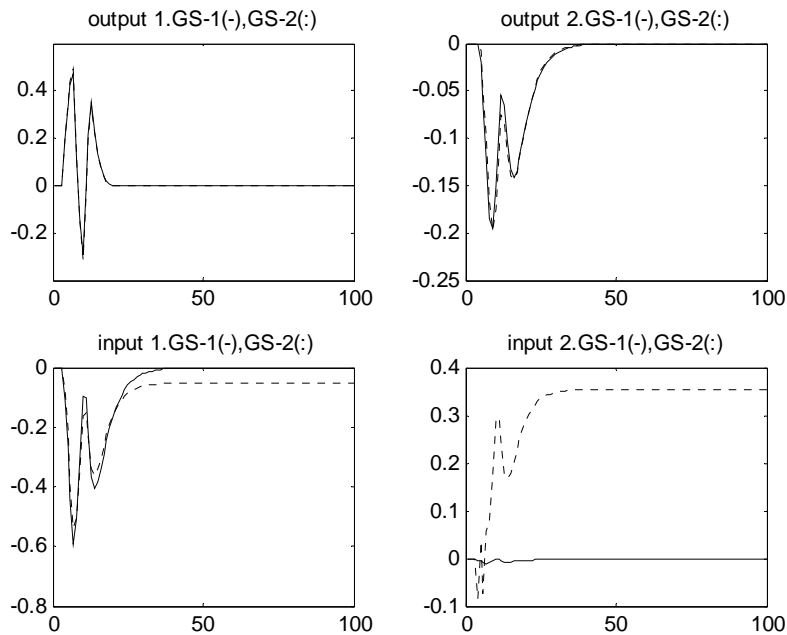


Figure 7.3 GS-MPC-1 (solid line) and GS-MPC-2 (dotted line) simulation

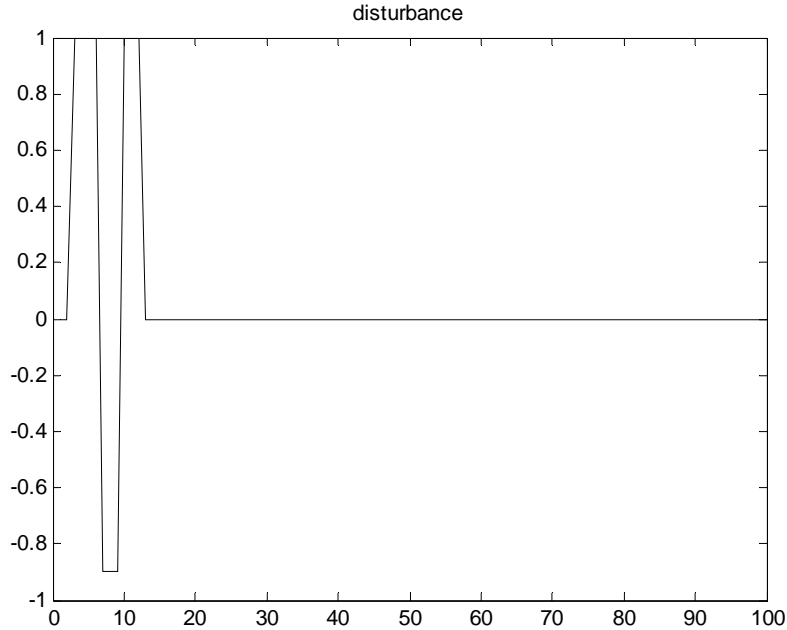


Figure 7.4 Disturbance signal used for the simulation results in Figure 7.3 and Table 7.1

Table 7.1 Simulation results of two MPC controllers

Controller name	Discretization	$\gamma_{simulation}$	$\sum_{t=1}^{100} \mathbf{u}^T(t)\mathbf{u}(t)$
GS-MPC-1	$\begin{bmatrix} u_1 \in [-1,0], u_2 \in [-1,-0.8] \\ u_1 \in [0,1], u_2 \in [-0.8,1] \end{bmatrix}$	0.3517	1.8716
GS-MPC-2	$\begin{bmatrix} u_1 \in [-1,0], u_2 \in [-1,0] \\ u_1 \in [0,1], u_2 \in [0,1] \end{bmatrix}$	0.3582	12.4613

Observing the upper two plots in Figure 7.3, the outputs of GS-MPC-1 and GS-MPC-2 are very close to each other, and it is difficult to conclude which controller is better than the other. However, the inputs are very different showing by the lower two plots in Figure 7.3, especially by the plots at the lower right corner. In the plot at the lower right corner of Figure 7.3, the solid line shows that the GS-MPC-1 controller results in a much less aggressive control action to achieve the similar output, than the GS-MPC-2 controller that is based on a different discretization of the input range.

The simulation results show that the GS-MPC-1 controller based on properly designed discretization of the input, results in a much less aggressive control action to achieve a desired performance than the GS-MPC-2 controller based on an even input discretization. This motivates further study in the design of gain-scheduled MPC controllers, including not only the design of the input weights studied in Chapter 6, but also the discretization of the operation range. The number of the discretized sub-ranges and the discretization points are both important design parameters towards a less conservative gain-scheduled MPC controller design.

In addition, it can be concluded from the above simulation results that the control action, represented as the sum of squares inputs, is also an important measure of robust performance. It is suggested for future work that this control action is incorporated into the design in addition to the performance index γ that reflects the sum of squares errors only. This will also reduce the conservatism of the analysis, by differentiating between controllers which will take different control action to give a similar γ .

Two approaches have been proposed in Chapters 4 and 5 to reduce conservatism of the design, which are based on parameter-dependent Lyapunov functions, and relaxation of the input-saturation factor. In the future, these two approaches should be applied to the design of robust gain-scheduled MPC controllers as well, for further reduction of the conservatism.

Last but not the least, MPC controllers are known to handle hard constraints with success, and the future application of the MPC controller design will be focused on MPC controllers with constraints.

7.2.2 Reducing conservatism of the robustness analysis

The LMIs based robust stability and performance analysis is inherently conservative because it considers cases that will not actually happen during operation. The more

conservative the analysis is, the less reliable the design results are. In other words, the success of the robust designs of gain-scheduled controllers relies heavily on the less conservative analysis of the closed-loop system's robust stability and performance. It is of extreme importance to reduce the conservatism of the robustness analysis conditions proposed in this work. For future work, the following issues are suggested towards less conservative analysis and thus more reliable designs.

7.2.2.1 Alternative robust performance condition formulation

In this section, the same system which has been considered in this work is applied to illustrate an alternative robust performance formulation. The system is given as follows:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\eta}(t+1) \\ e(t) \end{bmatrix} &= \begin{bmatrix} \mathbf{A}(\boldsymbol{\delta}_t) & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \boldsymbol{v}(t) \end{bmatrix} \\ \boldsymbol{\eta}(0) &= \boldsymbol{\eta}_0 \end{aligned} \quad (7.5)$$

where $\boldsymbol{\delta}_t = (\delta_1, \delta_2, \dots, \delta_n) \in \mathbf{R}^n$ is a vector of uncertain and time-varying real parameters. For this class of systems, the definition for quadratic Lyapunov robust performance has been given by Definition 4.2 in Chapter 4. It will be reformulated in this section to reduce the conservatism. If the process will be assumed to be always operated for disturbances within a specific time-dependent envelope, tighter robust performance bounds could be obtained. This formulation is based on the assumption that the disturbance will always evolve with time along some *a priori* known set of process disturbance trajectories. This is expected to reduce conservatism of the robustness analysis, such that the resulting LMIs performance index γ is as close to the simulation performance index $\gamma_{simulation}$ as possible.

The robust performance condition is formulated in Definition 4.2 as follows:

$$V(t+1) - V(t) + e^T(t)e(t) - \gamma^2 \boldsymbol{v}^T(t)\boldsymbol{v}(t) < 0 \quad (7.6)$$

An alternative robust performance condition is suggested here, by summing up equation (7.6) over the time interval $[t \ t + n - 1]$, as follows:

$$V(t+n) - V(t) + \sum_{i=0}^{n-1} e^T(t+i)e(t+i) - \gamma^2 \sum_{i=0}^{n-1} v^T(t+i)v(t+i) < 0 \quad (7.7)$$

As $n \rightarrow \infty$, the above equation (7.7) is equivalent to the condition $\|\mathbf{e}\|_{L_2} < \gamma \|\mathbf{v}\|_{L_2}$. For a finite n , equation (7.7) represents the new formulation of the robust performance condition proposed in this section. To illustrate this condition for a finite n , the formulations for $n = 2$ are given in the following.

$$\begin{aligned} & \boldsymbol{\eta}(t+2)^T \mathbf{P} \boldsymbol{\eta}(t+2) - \boldsymbol{\eta}(t)^T \mathbf{P} \boldsymbol{\eta}(t) + \\ & \sum_{i=0}^1 e^T(t+i)e(t+i) - \gamma^2 \sum_{i=0}^1 v^T(t+i)v(t+i) < 0 \end{aligned} \quad (7.8)$$

By substitution of the closed-loop system given by equation (4.15) into equation (7.8), it can be rewritten as the sum of the following three parts:

$$\begin{aligned} & \boldsymbol{\eta}(t+2)^T \mathbf{P} \boldsymbol{\eta}(t+2) - \boldsymbol{\eta}(t)^T \mathbf{P} \boldsymbol{\eta}(t) = [\boldsymbol{\eta}^T(t) \quad v^T(t) \quad v^T(t+1)] \mathbf{M} \begin{bmatrix} \boldsymbol{\eta}(t) \\ v(t) \\ v(t+1) \end{bmatrix} \quad (7.9) \\ 1. & \mathbf{M} = \begin{bmatrix} \mathbf{W}_{\boldsymbol{\eta}}^T (\mathbf{A}_t^T \mathbf{A}_{t+1}^T \mathbf{P} \mathbf{A}_{t+1} \mathbf{A}_t - \mathbf{P}) \mathbf{W}_{\boldsymbol{\eta}} & \mathbf{W}_{\boldsymbol{\eta}}^T (\mathbf{A}_t^T \mathbf{A}_{t+1}^T \mathbf{P} \mathbf{A}_1 \mathbf{B}) \mathbf{W}_{v_0} & \mathbf{W}_{\boldsymbol{\eta}}^T (\mathbf{A}_t^T \mathbf{A}_{t+1}^T \mathbf{P} \mathbf{B}) \mathbf{W}_v \\ \mathbf{W}_{v_0}^T (\mathbf{B}^T \mathbf{A}_{t+1}^T \mathbf{P} \mathbf{A}_{t+1} \mathbf{A}_t) \mathbf{W}_{\boldsymbol{\eta}} & \mathbf{W}_{v_0}^T (\mathbf{B}^T \mathbf{A}_{t+1}^T \mathbf{P} \mathbf{A}_{t+1} \mathbf{B}) \mathbf{W}_{v_0} & \mathbf{W}_{v_0}^T (\mathbf{B}^T \mathbf{A}_{t+1}^T \mathbf{P} \mathbf{B}) \mathbf{W}_{v_1} \\ \mathbf{W}_{v_1}^T (\mathbf{B}^T \mathbf{P} \mathbf{A}_{t+1} \mathbf{A}_t) \mathbf{W}_{\boldsymbol{\eta}} & \mathbf{W}_{v_1}^T (\mathbf{B}^T \mathbf{P} \mathbf{A}_{t+1} \mathbf{B}) \mathbf{W}_{v_0} & \mathbf{W}_{v_1}^T (\mathbf{B}^T \mathbf{P} \mathbf{B}) \mathbf{W}_{v_1} \end{bmatrix} \end{aligned}$$

and

$$\begin{aligned}
& \sum_{i=0}^1 e^T(t+i)e(t+i) = \begin{bmatrix} \boldsymbol{\eta}^T(t) & \boldsymbol{\nu}^T(t) & \boldsymbol{\nu}^T(t+1) \end{bmatrix} \mathbf{M}_e \begin{bmatrix} \boldsymbol{\eta}(t) \\ \boldsymbol{\nu}(t) \\ \boldsymbol{\nu}(t+1) \end{bmatrix} \quad (7.10) \\
2. & \mathbf{M}_e = \begin{bmatrix} \mathbf{W}_\eta^T (\mathbf{A}_t^T \mathbf{C}^T \mathbf{C} \mathbf{A}_t + \mathbf{C}^T \mathbf{C}) \mathbf{W}_\eta & \mathbf{W}_\eta^T (\mathbf{A}_t^T \mathbf{C}^T \mathbf{C} \mathbf{B} + \mathbf{C}^T \mathbf{D}) W_{\nu 0} & \mathbf{W}_\eta^T (\mathbf{A}_t^T \mathbf{C}^T \mathbf{D}) W_{\nu 1} \\ W_{\nu 0}^T (\mathbf{B}^T \mathbf{C}^T \mathbf{C} \mathbf{A}_t + \mathbf{D}^T \mathbf{C}) \mathbf{W}_\eta & W_{\nu 0}^T (\mathbf{B}^T \mathbf{C}^T \mathbf{C} \mathbf{B} + \mathbf{D}^T \mathbf{D}) W_{\nu 0} & W_{\nu 0}^T (\mathbf{B}^T \mathbf{C}^T \mathbf{D}) W_{\nu 1} \\ W_{\nu 1}^T (\mathbf{D}^T \mathbf{C} \mathbf{A}_t) \mathbf{W}_\eta & W_{\nu 1}^T (\mathbf{D}^T \mathbf{C} \mathbf{B}) W_{\nu 0} & W_{\nu 1}^T (\mathbf{D}^T \mathbf{D}) W_{\nu 1} \end{bmatrix}
\end{aligned}$$

and

$$\begin{aligned}
& \gamma^2 \sum_{i=0}^1 \boldsymbol{\nu}^T(t+i)\boldsymbol{\nu}(t+i) = \begin{bmatrix} \boldsymbol{\eta}^T(t) & \boldsymbol{\nu}^T(t) & \boldsymbol{\nu}^T(t+1) \end{bmatrix} \mathbf{M}_\nu \begin{bmatrix} \boldsymbol{\eta}(t) \\ \boldsymbol{\nu}(t) \\ \boldsymbol{\nu}(t+1) \end{bmatrix} \quad (7.11) \\
3. & \mathbf{M}_\nu = \begin{bmatrix} \mathbf{W}_\eta^T(0) \mathbf{W}_\eta & 0 & 0 \\ 0 & W_{\nu 0}^T(\gamma^2) W_{\nu 0} & 0 \\ 0 & 0 & W_{\nu 1}^T(\gamma^2) W_{\nu 1} \end{bmatrix}
\end{aligned}$$

where $\mathbf{A}_t = \mathbf{A}(\boldsymbol{\delta}(t))$, $\mathbf{A}_{t+1} = \mathbf{A}(\boldsymbol{\delta}(t+1))$. \mathbf{W}_η is the weight for the state $\boldsymbol{\eta}(t)$ at $t=0$, and it is zero for zero initial states, $W_{\nu 0}, W_{\nu 1}$ define the bounds of an envelope of possible disturbances represented by $\boldsymbol{\nu}(t), \boldsymbol{\nu}(t+1)$.

The advantage of using this formulation is that the information of the disturbance trajectory can be taken into account in the robustness analysis, by using the values of $W_{\nu 0}, W_{\nu 1}$ to represent an envelope of possible disturbances. The key difference is that the disturbance weight considered in previous chapters was constant whereas here it is proposed to vary this weight with respect to time in order to consider a narrower envelope of possible disturbances.

In summary, equation (7.8) can be rewritten as the sum of the three equations (7.9), (7.10) and (7.11), as follows:

$$\begin{aligned}
& \boldsymbol{\eta}(t+2)^T \mathbf{P} \boldsymbol{\eta}(t+2) - \boldsymbol{\eta}(t)^T \mathbf{P} \boldsymbol{\eta}(t) + \tag{7.12} \\
& \sum_{i=0}^1 e^T(t+i) e(t+i) - \gamma^2 \sum_{i=0}^1 \mathbf{v}^T(t+i) \mathbf{v}(t+i) < 0 \\
& \Leftrightarrow \\
& \begin{bmatrix} \boldsymbol{\eta}^T(t) & \mathbf{v}^T(t) & \mathbf{v}^T(t+1) \end{bmatrix} (\mathbf{M} + \mathbf{M}_e + \mathbf{M}_v) \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{v}(t) \\ \mathbf{v}(t+1) \end{bmatrix} < 0
\end{aligned}$$

Based on the two assumptions associated with equation (4.15) and the results developed in Chapter 4, equation (7.12) can be formulated as a finite set of LMIs and thus be solved with the FEASP or GEVP problem with MATLAB.

7.2.2.2 Elimination of the convexity condition for the parameter-dependent Lyapunov function analysis

In Chapter 4, the parameter-dependent Lyapunov functions have been used in the LMIs based robust stability and performance conditions to reduce the conservatism of the robustness analysis. The finite LMIs conditions are summarized in Theorem 4.3 and 4.4, where a convexity condition is included in each of them. The reason for this convexity condition is given in detail in Chapter 4. As a reminder, the robust stability condition leads to a term as follows:

$$\mathbf{L}(\boldsymbol{\delta}_t) = \mathbf{A}(\boldsymbol{\delta}_t)^T \mathbf{P}(\boldsymbol{\delta}_{t+1}) \mathbf{A}(\boldsymbol{\delta}_t) - \mathbf{P}(\boldsymbol{\delta}_t) < \mathbf{0} \tag{7.13}$$

For any nonzero vector $\boldsymbol{\lambda}$, clearly $f(\boldsymbol{\delta}_t) = \boldsymbol{\lambda}^T \mathbf{L}(\boldsymbol{\delta}_t) \boldsymbol{\lambda}$ is a scalar function of the following form:

$$\begin{aligned}
f(\boldsymbol{\delta}_t) &= \boldsymbol{\lambda}^T \mathbf{L}(\boldsymbol{\delta}_t) \boldsymbol{\lambda} \tag{7.14} \\
f(\boldsymbol{\delta}_t) &= f(\delta_{i,t}, \delta_{i,t} \delta_{j,t}, \delta_{i,t}^2, \delta_{i,t}^2 \delta_{j,t}, \delta_{i,t}^3)
\end{aligned}$$

In general, the negative sign of $f(\delta_t)$ values at all corners of the parameter box $\mathbf{W} \times \mathbf{S}$, defined in Chapter 3, does not guarantee its negativity over the entire parameter box, because of its 3rd-order dependence with respect to $\delta_{i,t}$'s. However, negativity is obtained when $f(\delta_t)$ is convex in the $\delta_{i,t}$'s, that is, when $\frac{\partial^2 f(\delta_t)}{\partial \delta_{i,t}^2} \geq 0, i = 1, \dots, n$ for all δ_t . The parameter-dependent Lyapunov functions have been proposed to reduce the conservatism of the analysis, but the addition of the convexity conditions will affect the reduction efficiency. It is desired to eliminate these additional convexity conditions to achieve much less conservative designs. This can be realized based on the application of the state-affine model, and it will be explained in the sequel.

The key advantage of using state-affine model in this work is that the process uncertainty is a function of the current input only, as follows:

$$\delta_{i,t} = u(t)^i \quad (7.15)$$

As a result, the 3rd-order terms in equation (7.14) can be transformed into 1st-order terms as follows:

$$\begin{aligned} \delta_{i,t} \delta_{j,t} &= u(t)^i u(t)^j = u(t)^{i+j} = \delta_{i+j,t} \\ \delta_{i,t}^3 &= u(t)^i u(t)^i u(t)^i = u(t)^{i \times 3} = \delta_{i \times 3,t} \end{aligned} \quad (7.16)$$

Based on this, equation (7.14) can be rewritten as follows:

$$\begin{aligned} f(\delta_t) &= \lambda^T \mathbf{L}(\delta_t) \lambda \\ f(\delta_t) &= f(\delta_{i,t}, \delta_{i+j,t}, \delta_{i \times 2,t}, \delta_{i \times 2+j,t}, \delta_{i \times 3,t}) \end{aligned} \quad (7.17)$$

And it is no longer a function which has a 3rd-order dependence with respect to $\delta_{i,t}$'s. Thus, the corresponding convexity condition in Theorems 4.3 and 4.4 is no longer

necessary, and the resulting LMIs may potentially be less conservative by having fewer LMI terms. At the same time, this order-reduction has removed the correlation between the uncertain parameters $\delta_{i,t}$ and $\delta_{i,t}^3$.

However, this may raise another problem related to the addition of more uncertain elements to the uncertainty vector. For example, $\delta_{i+j,t}$, $\delta_{i \times 2,t}$, $\delta_{i \times 2+j,t}$, $\delta_{i \times 3,t}$ are added and additional vertices corresponding to the bounds of these new perturbations have to be integrated into the parameter box. Thus, there is a tradeoff between the elimination of the convexity condition and the addition of uncertain elements. Some future research effort is desired to look into this issue.

7.2.2.3 Selection of the vertices of the uncertain parameter box

The parameter box, defined in chapter 3, represents the range of the uncertain parameters upon which the robust stability and performance conditions have to be tested. The size and shape of the parameter box should be representing the true uncertain parameters as accurate as possible, and this could be manipulated by careful selection of the vertices based on the uncertain parameter bounds. If it is possible to consider a smaller parameter box, it is possible to reduce the conservatism of the analysis. However, it is not always possible to find out the correlation among the uncertain parameters, which may be helpful in determining the smaller parameter box. In this work, based on the application of the state-affine model, the uncertainty is a function of the current input variable, shown by equation (3.6). As a result, all the uncertain parameters can be expressed as a function of other uncertain parameters. For example, in the case of two uncertain parameters δ_1, δ_2 , according to equation (3.6), $\delta_2 = (\delta_1)^2$, the true relationship between the two uncertain parameters is represented by the dotted curve in Figure 7.5.

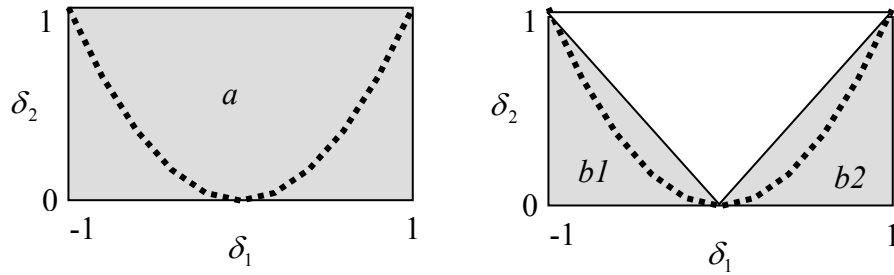


Figure 7.5 Parameter box

In the LMIs based robustness analysis, the bounds of the two uncertain parameters are used as vertices of a parameter box, to approximate this dotted curve. If the bounds of the two uncertain parameters δ_1, δ_2 are $\delta_1 \in [-1, 1]$ and $\delta_2 \in [0, 1]$ respectively, then the four vertices to represent them are $[\delta_1, \delta_2] = [-1, 0], [-1, 1], [1, 0], [1, 1]$. The corresponding parameter box is the shaded rectangle area a in Figure 7.5. It is easy to notice that this area a is unnecessarily conservative for bounding the parabolic functional dependency existent between the uncertainty elements.

For example, a less conservative alternative is to use an additional vertex, $[\delta_1, \delta_2] = [0, 0]$, to define a less conservative parameter box. The new two sets of vertices are $[\delta_1, \delta_2] = [-1, 0], [-1, 1], [0, 0]$ and $[\delta_1, \delta_2] = [1, 0], [1, 1], [0, 0]$, and they define the two triangles $b1$ and $b2$ in Figure 7.5. The total area of the two triangles $b1$ and $b2$ is half of the area of the rectangle area a , thus conservatism is expected to be reduced. However, two more vertices are added to the LMIs and thus the resulting problem will have more LMI conditions. This is another tradeoff scenario that could be studied in the future.

7.2.2.4 Schedule the robustness analysis along the operation range

If the nonlinear process is represented with a state-affine model, the uncertain parameters are functions of the current input according to equation (3.6), and the LMIs tests have been formulated around the process steady state corresponding to $u(t) = 0$. However, the

gain-scheduled controller design has assumed the input variable to change over the operation. For the design of gain-scheduled PI controllers, the controller parameters are designed to change as a continuous function of the variable $u(t)$ over the operation range. For the design of gain-scheduled MPC controllers, the controller tuning parameters are scheduled discretely along the sub-ranges, discretized of the operation range. As a result, it may be less conservative to formulate the LMIs around different steady states, corresponding to a set of values of the input variable over the operation range. This will require that a corresponding empirical model be identified around each of the steady state, and the uncertainty bounds will be obtained based on the corresponding model. The resulting LMIs will be reformulated along the operation range using this approach, and possibly more vertices will be required to cover the entire window of operation in terms of the manipulated variable u . It will also be interesting to integrate a highly nonlinear process example operated around different operating conditions. It requires more future work to find this out.

References

- [1]. Apkarian, P., Gahinet, P. and Becker, G. (1995). Self-scheduled H_∞ Control of Linear parameter-varying systems: a Design Example. *Automatica*. Vol.31, No.9, pp. 1251-1261.
- [2]. Apkarian, P., Gahinet, P. and Becker, G. (1995). Self-scheduled H_∞ Control of Linear parameter-varying systems: a Design Example. *Automatica*. Vol.31, No.9, pp. 1251-1261.
- [3]. Balza, C., Fromageot, A. and Maniere, M. (1967). Four-level pseudo-random Sequences. *Electronics Letters*, Vol.3, No.7, pp. 313-315.
- [4]. Balza, C., Fromageot, A. and Maniere, M. (1967). Four-level pseudo-random Sequences. *Electronics Letters*, Vol.3, No.7, pp. 313-315.
- [5]. Barmish, B., Khargonekar, P., Shi, Z., and Tempo, R. (1990). Robustness margin need not be a continuous function of the problem data. *Systems and Control Letters*. 15(2). pp. 91-98.
- [6]. Becker, G. and Packard, A. (1993). Robust Performance of LPV's using Parametrically Dependent Linear Feedback. *Systems and Control Letters*. Vol. 23, pp. 205-215.
- [7]. Bequette, B.W. (1997). Gain Scheduled Process Control: a Review (1997). (presented at) the NATO ASI on Nonlinear Model Based Control. Antalya, Turkey, August.
- [8]. Boyd, S., Ghaoui, L. EL., Feron, E. and Balakrishnan, V. (1994). *Linear Matrix Inequalities in System and Control Theory*. SIAM (studies in applied mathematics).
- [9]. Budman, H. M. and Knapp, T. D. (2001). Stability Analysis of Nonlinear Processes using Empirical State-affine Models and LMIs. *Journal of Process Control*. 11, pp. 375-386.
- [10]. Budman, H. M. and Knapp, T. D.. Stability of a PI-controlled Nonlinear System.
- [11]. Campo, P.J. and Morari, M. (1987). Robust Model Predictive Control. *Proceedings of American Control Conference*. Vol.2, pp. 1021-1026.

- [12]. Chang, J.A. (1966). Generation of 5-level Maximal-length Sequences. *Electronics Letters*. 2, pp. 258.
- [13]. Chen, H. and Allgower, F. (1998a). A computationally attractive nonlinear predictive control scheme with guaranteed stability for stable systems. *Journal of Process Control*. Vol. 8, No. 5-6. pp. 475-485.
- [14]. Chen, H. and Allgower, F. (1998b). Nonlinear model predictive control schemes with guaranteed stability. *MATO ASI Series E: Applied sciences*. Vol. 353 (Nonlinear Model Based Process Control). pp. 465-494.
- [15]. Chen, H., Scherer, C. W., and Allgower, F. (1997). A game theoretic approach to nonlinear robust receding control of constrained systems. *Proceedings of American Control Conference*. Albuquerque. pp. 3073-3077.
- [16]. Church, R. (1935). Tables of Irreducible Polynomials for the First Four Prime Moduli. *The Annals of Mathematics*. Vol.36, pp. 98-209.
- [17]. Costanza, V., Dickinson, B. and Johnson, E. (1983). Universal Approximations of Discrete-time Control Systems over Finite Time. *IEEE Transactions on Automatic Control*. AC-28, pp. 439-452.
- [18]. Dang Van Mien, H. and Normand-Cyrot, D. (1984). Nonlinear State-affine Identification Methods: Applications to Electrical Power Plants. *Automatica*. 20(2), pp. 175-188.
- [19]. Darnell, M. (1966). Synthesis of Pseudo-random Signals Derived from p -level m sequences. *Electronics Letters*. Vol. 2, pp. 428-430.
- [20]. Davison, D. (2001). ECE788 Robust Control Course Notes. University of Waterloo.
- [21]. Diaz, H. and Desrochers, A. (1988). Modeling of Nonlinear Discrete-time Systems from Input/Output Data. *Automatica*. 24(5), pp. 629-641.
- [22]. Doyle III, F. J. and Morari, M. (1990). A Conic Sector-based Methodology for Nonlinear Control Design. *ACC Proceedings, San Diego, CA, USA*. pp. 2746-2751.
- [23]. Doyle III, F. J., Ogunnaike, B. and Pearson, R. (1995). Nonlinear Model-based Control using Second Order Volterra Models. *Automatica*. Vol. 31, pp. 697-714.
- [24]. Doyle III, F. J., Packard, A. and Morari, M. (1989). Robust Controller Design for a Nonlinear CSTR. *Chemical Engineering Science*. Vol. 44, pp. 1929-1947.

- [25]. Doyle III, F. J., Packard, A., and Zhou, K. (1991). Review of LFTs, LMIs, and μ . Proceedings of the 30th Conference on Decision and Control, Brighton, England. pp. 1227-1232.
- [26]. Doyle, J.C. (1982). Analysis of feedback systems with structured uncertainties. IEEE Proceedings. Vol. 129, Part D, no. 6, pp. 242-250, November.
- [27]. Doyle, J.C. and Packard, A. (1987). Uncertain Multivariable Systems from a State Space Perspective. ACC Proceedings, New Jersey. pp. 2147-2152.
- [28]. Doyle, J.C. and Packard, A. (1988). Robust control of Multivariable and Large Scale Systems. USAF Office of Scientific Research, Washington, DC.
- [29]. Fogler, H.S. (1986). Elements of Chemical Reaction Engineering. Prentice-hall, Englewood Cliffs, NJ.
- [30]. Gahinet, P. and Apkarian, P. (1994). A Linear Matrix Inequality Approach to H_∞ Control. International Journal of Robust and Nonlinear Control. 4, pp. 421-448.
- [31]. Gahinet, P., Apkarian, P., and Chilali, M. (1996). Affine Parameter-dependent Lyapunov Functions and Real Parametric Uncertainty. IEEE Transactions on Automatic control, 41(3), pp. 436-442.
- [32]. Gahinet, P., Nemirovski, A., Laub, A. J. and Chilali, M. (1995). LMI Control Toolbox User's Guide. The MathWorksTM, Inc..
- [33]. Gahinet, P.; Apkarian, P. and Chilali, M. (1994). Affine Parameter-dependent Lyapunov Functions for Real Parametric Uncertainty. Proceedings of the 33rd IEEE Conference on Decision and Control. Lake Buena Vista, Fl, pp. 2026-2031.
- [34]. Gao, J. and Budman, H. M. (2003). Design of Sub-optimal Robust Gain-scheduled PI Controllers. ADCHEM, HongKong.
- [35]. Gao, J. and Budman, H. M. (to be published, 2004). Reducing conservatism in the design of a robust gain-scheduled PI controller for nonlinear chemical processes. International Journal of Control.
- [36]. Garofalo, F., Celentano, G., and Glielmo, L. (1993). Stability Robustness of Interval Matrices via Laypunov Quadratic Forms. IEEE Transactions on Automatic Control. 38(2), pp. 281-284.
- [37]. Genceli, H. and Nikolao, M. (1993). Robust Stability Analysis of Constrained ℓ_1 -

- norm Model Predictive Control. *AIChE Journal*. 39(12), pp. 1954-1965.
- [38]. Godfrey, K.R. (1966). Three-level m Sequences. *Electronics Letters*. Vol. 2, pp. 241-243.
- [39]. Haber, R. and Unbehauen, H. (1990). Structure Identification of Nonlinear Dynamic Systems- a Survey on Input/output Approaches. *Automatica*. Vol. 26, No. 4, pp. 651-677.
- [40]. Hernandez, E. and Arkun, Y. (1993). Control of Nonlinear Systems using Polynomial ARMA Models. *AIChE Journal, Process Systems Engineering*. Vol. 39, No. 3, March.
- [41]. Hoo, K.A. and Kantor, J.C. (1985). An Exothermic Continuously Stirred Tank Reactor is Feedback Equivalent to a Linear System. *Chemical Engineering Communications*. Vol. 37, pp. 1-10.
- [42]. Knapp, T. D. and Budman, H. M. (2000). Robust Control Design of Non-linear Process using Empirical State-affine Models. *International Journal of Control*. 73(17), pp. 1525-1535.
- [43]. Kothare, M.V., Balakrishnan, V. and Morari, M. (1996). Robust Constrained Model Predictive Control using Linear Matrix Inequalities. *Automatica*. Vol. 32, No. 10, pp. 1361-1379.
- [44]. Kravaris, C. and Palanki, S. (1988). Robust Nonlinear State Feedback under Structured Uncertainty. *AIChE Journal*. 7, pp. 1119-1127.
- [45]. Li, J. and et al. (1999). Synthesis of Gain-scheduled Controller for a Class of LPV Systems. *Proceedings of the 38th conference on decision & control*, Pheonix, Arizona, USA, December.
- [46]. MacWilliams, F.J. and Sloane, N.J.A. (1976). Pseudo-random Sequences and Arrays. *Proceedings of the IEEE*, Vol. 64, No. 12, pp. 1715-1728.
- [47]. Marmarelis, P.Z. and Mararelis, V.Z. (1978). *Analysis of Physiological Systems, the White Noise Approach*. New York: Plenum.
- [48]. Morari, M. and Zafiriou, E. (1989). *Robust Process Control*. Prentice-Hall, Englewood Cliffs, NJ.
- [49]. Mutha, R. K., Cluett, W. R. and Penlidis, A. (1997). Nonlinear Model-based Predictive Control of Control Nonaffine Systems. *Automatica*. Vol.33, No.5, pp. 907-

913.

- [50]. Nowak, R. D. and Van Veen, B. D. (1994). Random and Pseudo-random Inputs for Volterra Filter Identification. *IEEE Transactions on Signal Processing*. Vol.42, No.3, pp. 2124-2135.
- [51]. Packard, A. and Doyle, J. (1988). Structured Singular Value with Repeated Scalar Blocks. *ACC Proceedings, Atlanta*. pp. 1213-1218.
- [52]. Packard, A. and Doyle, J. (1990). Quadratic Stability with Real and Complex Perturbations. *IEEE Transactions on Automatic Control*. 35(2), pp. 198-201.
- [53]. Packard, A., Zhou, K., Pandey, P., and Becker, G. (1991). A Collection of Robust Control Problems Leading to LMI's. *Proceedings of the 30th conference on decision and control*, Brighton, England, IEEE, December. pp. 1245-1250.
- [54]. Pearson, R., Ogunnaike, B. and Doyle, III F. J. (1992). Identification of Discrete Convolution Models for Nonlinear Processes. *Proceedings of AIChE Annual Meeting*.
- [55]. Rivera, D.E., Morari, M., and Skogestad, S. (1986). Internal Model Control: 4. PID Controller Design. *Industrial & engineering chemistry process design and development*. 25, pp. 252-265.
- [56]. Sanchez-Pena and Snaier, *Robust Systems: Theory and Applications*, Wiley, 1998.
- [57]. Sandberg, I.W. (1992). Uniform Approximation with Doubly Finite Volterra Series. *IEEE Transactions on Signal Processing*. Vol.40, pp. 1438-1442, June.
- [58]. Schetzen, M. (1989). *The Volterra and Wiener Theories of Nonlinear Systems*. Northeastern University.
- [59]. Seborg, D. E., Edgar, T. F. and Mellichamp, D. A. (1989). *Process Dynamics and Control*, John Wiley & Sons.
- [60]. Shamma, J. S. and Athans, M. (1987). Stability and Robustness of Slowly-varying Linear Systems. *Proceedings of the 26th IEEE Conference on Decision and Control*, Los Angeles, CA, December.
- [61]. Shamma, J. S. and Athans, M. (1990). Analysis of Gain Scheduled Control for Nonlinear Plants. *IEEE Transactions on Automatic Control*. Vol. 35, No.8, August.
- [62]. Shamma, J. S. and Athans, M. (1991). Guaranteed Properties of Gain Scheduled Control for Linear Parameter-varying Plants. *Automatica*. Vol.27, No.3, pp. 559-564.

- [63]. Shamma, J. S. and Athans, M. (1992) Gain Scheduling: potential hazards and Possible Remedies. IEEE Control Systems. pp. 102-107, June.
- [64]. Sivrioglu, S. and Nonami, K. (1996). LMI Approach to Gain scheduled H_∞ Control beyond PID Control for Gyroscopic Rotor-Magnetic Bearing. Proceedings of the 35th Conference on Decision and Control, Kobe, Japan. pp. 3694-3699, December.
- [65]. Slotine, J-J E. and Li, W. (1991). Applied Nonlinear Control. Prentice hall, New Jersey.
- [66]. Sontag, E. (1978). Realization Theory of Discrete-time Nonlinear Systems: Part I. the Bounded Case. IEEE Transactions on Circuits and Systems. CAS-26(4), pp. 342-356.
- [67]. Wang, F. and Balakrishnan V. (1999). Robustness Analysis and Gain-scheduled Controller Synthesis for Rational Parameter-dependent Systems using Parameter-dependent Lyapunov Functions. Proceedings of the 38th IEEE Conference on Decision and Control, Phoenix, Arizona USA. December.
- [68]. Watanabe, R., Uchida, K. and Fujita, M. (1996). A New LMI Approach to Analysis of Linear Systems with Scheduling Parameter-Reduction to Finite Number of LMI Conditions. Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japan. pp. 1663-1665, December.
- [69]. Zafiriou, E. (1990). Robust Model Predictive Control of Processes with Hard Constraints. Computers and Chemical Engineering. Vol.14, No.4/5, pp. 359-371.
- [70]. Zanollo, R. and Budman, H. M. (1999). Model predictive control with soft constraints with application to lime kiln control. Computers and Chemical Engineering. 23. pp. 791-806.
- [71]. Zheng, A. and Morari, M. (1993). Robust Stability of Constrained Model Predictive Control. Proceedings of American Control Conference. Vol.1, San Francisco, CA. pp. 379-383.
- [72]. Zhou, K., Khargonekar, P.P., Stoustrup, J. and Niemann, H.H. (1992). Robust Stability and Performance of Uncertain System in State Space. Proceedings of the 31st IEEE Conference on Decision and Control, Tucson, Arizon. pp. 662-667, December.
- [73]. Zierler, N. (1959). Linear Recurring Sequences. Journal of the Society for Industrial and Applied Mathematics. Vol. 7, pp. 31-48.

8 Appendix A: Nomenclature for CSTR

Table 8.1 Nomenclature for exothermic CSTR

A	heat transfer surface area
B	dimensionless heat of reaction: $B = -\Delta H C_f \gamma / C_p T_{f_0}$
C	reactant concentration
C_f	feed concentration of reactant
C_p	heat capacity
Da	Damkohler number: $Da = (V k_0 e^{-\gamma}) / Q_f$
Ea	activation energy
ΔH	heat of reaction
k_0	reaction rate constant
Q_f	mass feed flow-rate
R	ideal gas constant
T	reaction temperature
T_c	coolant temperature
T_f	feed temperature
T_{f_0}	nominal feed temperature
U	overall heat transfer coefficient
V	reactor volume
x_1	dimensionless concentration: $x_1 = (C_f - C) / C_f$
x_2	dimensionless temperature: $x_2 = (T - T_{f_0}) \gamma / T_{f_0}$
x_c	dimensionless coolant temperature: $x_c = (T_c - T_{f_0}) \gamma / T_{f_0}$
β	dimensionless cooling rate: $\beta = UA / Q_f C_p$
γ	dimensionless activation energy: $\gamma = Ea / RT_{f_0}$

9 Appendix B: MATLAB Code

The MATLAB files used in this work are summarized in this Appendix, in an order similar to the outline of this thesis.

9.1 Model Identification

The following MATLAB files have been used to generate process input/output data, identify the Volterra series model, and obtain the state-affine model.

Table 9.1 MATLAB files for model identification

No.	File name	Description
1.	InputDat	I/O data using PRMS as input
2.	CSTR	Deviated CSTR model for I/O data
3.	Volterra	I/O to Volterra(M=5)
4.	Vol2b5.M	Arrange Volterra kernels into Behavior matrix
5.	StateAff	Volterra to State-affine

Code 1: InputDat.M

```
clear
global U

%Variable list
%
%x10      Initial process conditions
%u        Process input
%x        Process output
%Tc       CSTR cooling water temperature
%length   Simulation time period
%U        Input to differential equation solver
%

% Initial Conditions of Process

x10=0*[0.62195;3.7092];%[0.4759;2.9045];
Tc(1)=0;
x(1)=x10(1,1);
temp=x(1);

% PRMS sequence, the input
baseVal=3;
powerVal=3;
shift=0;
whichSeq=3;
ms=mseq(baseVal,powerVal,shift,whichSeq);
bitNum=baseVal^powerVal-1;
Tc=ms;b=16;
for i=1:bitNum
    Tc((i-1)*b+1:i*b)=ms(i);
```



```

end
length=bitNum*b;

% corresponding process response
ooo(1)=1;
for i=2:length
    ooo(i)=i;
    x(i)=temp;
    U=Tc(i);
    [t,x1]=ode23('cstrd',[0,1],x10);
    A=[t,x1];
    [a,b]=size(A);
    temp=x1(a,1);
    x10=[x1(a,1);x1(a,2)];
end

%plot the input/output data
figure(1)
subplot(2,1,1)
plot(ooo(1:length),Tc(1:length),'k')
title('Input: 3-level PRMS')
axis([0 length -1.2 1.2])
subplot(2,1,2)
plot(ooo(1:length),x(1:length),'k')
title('Output: CSTR')
axis([0 length -0.6 0.4])

%save input/output data to file
save datacstr x Tc length;

```

Code 2:CSTR.M

```

function xlp=cstr(t,x1)

global U
u=U*9+14;
Da=0.072;
B=1;
beta=0.3;
gamma=20;
x11=[0.62195;3.7092];
x1=(x1+x11);
xlp(1,1)=-x1(1)+Da*(1-x1(1))*exp(x1(2)/(1+(x1(2)/gamma)));
xlp(2,1)=-x1(2)+B*Da*(1-x1(1))*exp(x1(2)/(1+(x1(2)/gamma)))-
beta*(x1(2)-u);

```

Code 3:volterra.M

```

%identify Volterra series model using Matlab System identification
toolbox

clear
load datacstr

%Variable list
%start      memory length of Volterra series model

```

```

%length    Simulation time period
%u         Process input
%y         Process output
%coeff     Volterra series model kernels

% Least Square Regression of the Volterra series model
start=6;
y=x(start:length)';
u=Tc(start:length);
ooo=[start:length];
z=[y u Tc(start-1:length-1).*u Tc(start-2:length-2).*u Tc(start-
3:length-3).*u...
    Tc(start-4:length-4).*u Tc(start-5:length-5).*u];
na=[0];nk=[1 1 2 3 4 5];nb=[5 5 4 3 2 1];
th=arx(z,[na nb nk]);
par=th2par(th);
ys=idsim([u Tc(start-1:length-1).*u Tc(start-2:length-2).*u ...
    Tc(start-3:length-3).*u Tc(start-4:length-4).*u Tc(start-
5:length-5).*u],th);
coeff=par;

%Simulation of the Volterra series model to compare to the original ata
for i=start:length,
ooo(i-start+1)=i-start+1;
y1(i-start+1)=coeff(1)*Tc(i-1)+coeff(2)*Tc(i-2)+coeff(3)*Tc(i-
3)+coeff(4)*Tc(i-4)+coeff(5)*Tc(i-5)+...
    coeff(6)*Tc(i-1)*Tc(i-1)+coeff(7)*Tc(i-2)*Tc(i-1)+coeff(8)*Tc(i-
3)*Tc(i-1)...
    +coeff(9)*Tc(i-4)*Tc(i-1)+coeff(10)*Tc(i-5)*Tc(i-1)...
    +coeff(11)*Tc(i-2)*Tc(i-2)+coeff(12)*Tc(i-3)*Tc(i-2)+coeff(13)*Tc(i-
4)*Tc(i-2)...
    +coeff(14)*Tc(i-5)*Tc(i-2)+coeff(15)*Tc(i-3)*Tc(i-3)...
    +coeff(16)*Tc(i-4)*Tc(i-3)+coeff(17)*Tc(i-5)*Tc(i-3)...
    +coeff(18)*Tc(i-4)*Tc(i-4)+coeff(19)*Tc(i-5)*Tc(i-4)...
    +coeff(20)*Tc(i-5)*Tc(i-5);
end

figure(2)
plot(ooo,y,'k',ooo,y1,'k:')
title('Volterra series model(:) and CSTR')
axis([0 length -0.6 0.4])
save volcoefcstrq1 coeff

```

Code 4:vol2b5.m

```

%Arrange Volterra kernels into Behavior matrix

%Variable list
%g        1st-order Volterra kernels
%b        2nd-order Volterra kernels

function [g,b]=vol2b5(coeff)
g(1)=0;
m=5;
g(2:m+1,1)=coeff(1:5);
b=zeros(5,5);

```

```

b(:,1)=coeff(6:10);
b(2:m,2)=coeff(11:14);
b(3:m,3)=coeff(15:17);
b(4:m,4)=coeff(18:19);
b(5:m,5)=coeff(20);
save v1cstrg5 g;
save v2cstrg5 b;

```

Code 5: StataAff.M

```

clear;

%Variable list
%
%u          Process inout
%y          Process output
%g          1st-order Volterra kernels
%b2         2nd-order Volterra kernels
%phi        Submatrix of B(f)
%phi0       Submatrix of B(f) for 1st-order terms
%phi1       Submatrix of B(f) for 2nd-order terms
%F0         State-affine model matrix
%F1         State-affine model matrix
%F2         State-affine model matrix
%G1         State-affine model matrix
%G2         State-affine model matrix
%H0         State-affine model matrix
%ysa1st     Simulation of 1st-order State-affine model
%ysa2nd     Simulation of 2nd-order State-affine model

load volcoefcstrq1;
coeff=coeff;
[g,b]=vol2b5(coeff);
load v1cstrg5;
load v2cstrg5;
load datacstr;
u=(Tc)-mean(Tc);
a=max(abs(u))
u=u/a;
y=(x-mean(x));
y=y/max(abs(y));
b2=b;

% 1st Order State-Affine Model

phi=[g(2) g(3);g(3) g(4)];
phi0=[g(3) g(4);g(4) g(5)];
phi1=zeros(2);
phi2=zeros(2);

for i=1:2,
    F0=inv(phi(1:i,1:i))*phi0(1:i,1:i);
    F1=inv(phi(1:i,1:i))*phi1(1:i,1:i);
    F2=inv(phi(1:i,1:i))*phi2(1:i,1:i);
    G1=inv(phi(1:i,1:i))*phi(1:i,1);
    H0=phi(1,1:i);

```

```

%Simulation of 1st-order State-affine model

    x=zeros(i,length);
    for j=2:length,
        x(:,j)=(F0+F1.*u(j-1)+F2.*u(j-1)^2)*x(:,j-1)+G1.*u(j-1);
        ysalst(i,j)=(H0)*x(:,j);
    end
end

% 2nd Order State-Affine Model

phi=[g(2) b2(1,1) g(3) b2(2,1) b2(2,2);
g(3) b2(2,2) g(4) b2(3,2) b2(3,3);
b2(2,1) 0 b2(3,1) 0 0;
g(4) b2(3,3) g(5) b2(4,3) b2(4,4);
b2(3,1) 0 b2(4,1) 0 0];

phi0=[g(3) b2(2,2) g(4) b2(3,2) b2(3,3);
g(4) b2(3,3) g(5) b2(4,3) b2(4,4);
b2(3,1) 0 b2(4,1) 0 0;
g(5) b2(4,4) g(6) b2(5,4) b2(5,5);
b2(4,1) 0 b2(5,1) 0 0];

phi1=[b2(2,1) 0 b2(3,1) 0 0;
b2(3,2) 0 b2(4,2) 0 0;
0 0 0 0 0;
b2(4,3) 0 b2(5,3) 0 0;
0 0 0 0 0];

phi2=zeros(5);ord=2;
for i=1:ord
    F0=inv(phi(1:i,1:i))*phi0(1:i,1:i);
    F1=inv(phi(1:i,1:i))*phi1(1:i,1:i);
    F2=inv(phi(1:i,1:i))*phi2(1:i,1:i);
    G1=inv(phi(1:i,1:i))*phi(1:i,1);
    G2=inv(phi(1:i,1:i))*phi(1:i,2);
    H0=phi(1,1:i);

%Simulation of 2nd-order State-affine model

    x=zeros(i,length);
    for j=2:length,
        x(:,j)=(F0+F1.*u(j-1)+F2.*u(j-1)^2)*x(:,j-1)+G1.*u(j-
1)+G2.*u(j-1)^2;
        ysa2nd(i,j)=H0*x(:,j-1);
    end
end
for i=1:length,ooo(i)=i;,end
sumerror=0;
for i=1:length
    sumerror=sumerror+(ysa2nd(ord,i)-y(i))^2;
end
sume5=sumerror

figure(1);
plot(ooo,y,'k',ooo,ysa2nd(ord,:),'k:');

```

```
title('State-affine(:) and CTSR')
save cstrmat F0 F1 F2 G1 G2 H0
```

9.2 Gain-scheduled PI Controllers Design

The following MATLAB files have been used to design and simulate gain-scheduled PI controllers, based on Lyapunov functions and parameter-dependent Lyapunov functions, and SSV approach.

9.2.1 Quadratic Lyapunov functions

Table 9.2 MATLAB files for Gain-scheduled PI design: fixed Lyapunov functions

No.	File name	Description
6.	LMIOpt	Optimization of PI parameters K_c, τ_I, W_c, W_d , calls the following function 8
7.	sysRS	The set of LMI for RS, calls 9, 10, and 13
8.	sysRP	The set of LMI for RP, calls 11, 12, and 13
9.	closysRS	Closed-loop system for RS
10.	LMikRS	A single LMI of RS for each vertex
11.	closysRP	Closed-loop system for RP
12.	LMikRP	A single LMI of RP for each vertex
13.	InputSat	Relaxation of input-saturation factor (Method 5.1)
14.	SimuPI	Simulate one Gain-Scheduled PI controller

Code 6: LMIOpt.M

```
x0=[0.38 0.3 0 0];
[xopt,gopt]=fminsearch('sysrp',x0)
```

Code 7: sysRS.M

```
clear

%Variable list
%x                Gain-scheduled PI controller parameters
%ulbnd           Bounds for uncertainty 1
%u2bnd           Bounds for uncertainty 2
%psai            Input-saturation factor
%psailow         Input-saturation factor lower bound (u=-1)
%psailow1        Input-saturation factor lower bound (u=1)
%A0,A1,A2        Closed-loop system matrices
%P0              Lyapunov matrix

x=[3.39 20 0 0]; %controller parameters,x=[Kc,taui,Wc,Wd]

%uncertainty bounds
ulbnd=[-1,1];
[n1,i1]=size(ulbnd);
u2bnd=[0,1];
```

```

[m1,j1]=size(u2bnd);

%LMI formulation
setlmis([])
P0=lmivar(1,[ns+1 1]);%P0 is symmetric block diagonal
k=1;
lmiterm([k 1 1 0],0);%P0>0
lmiterm([-k 1 1 P0],1,1);

%no input-saturation
psai=1;
[A0,A1,A2,ns]=closysrs(x,psai);

for i=1:i1
    for j=1:j1
        k=k+1;
        u1=ulbnd(i);u2=u2bnd(j);
        polA=A0+A1*u1+A2*u2;
        lmikrs(polA,P0,k);
    end
end

%with input-saturation
inpusat=1;
if inpusat==1
    [psailow,psailow1]=inputsat(x);
    i=1;
    [A0,A1,A2,ns]=closysrs(x,psailow);
    for j=1:j1
        k=k+1;
        u1=ulbnd(i);u2=u2bnd(j);
        polA=A0+A1*u1+A2*u2;
        lmikrs(polA,P0,k);
    end
    i=i1;
    [A0,A1,A2,ns]=closysrs(x,psailow1);
    for j=1:j1
        k=k+1;
        u1=ulbnd(i);u2=u2bnd(j);
        polA=A0+A1*u1+A2*u2;
        lmikrs(polA,P0,k);
    end
end

lmilio=getlmis;
[tmin,xffeas]=feasp(lmilio,[]);
P0=dec2mat(lmilio,xffeas,P0);
save Pcons P0

```

Code 8: sysRP.M

```

%Variable list
%x                Gain-scheduled PI controller parameters
%ulbnd            Bounds for uncertainty 1
%u2bnd            Bounds for uncertainty 2
%psai             Input-saturation factor

```

```

%psailow           Input-saturation factor lower bound (u=-1)
%psailow1         Input-saturation factor lower bound (u=1)
%A0,A1,A2         Closed-loop system matrices
%P0               Lyapunov matrix
%gamma           Performance index

function gamma=sysrp(x)
x=[0.38 0.3 0 0]; %controller parameters,x=[Kc,taui,Wc,Wd]

%uncertainty bounds
ulbnd=[-1,1];
[n1,i1]=size(ulbnd);
u2bnd=[0,1];
[m1,j1]=size(u2bnd);

%LMI formulation
setlmis([])
P0=lmivar(1,[ns+1+1 1]);%P0 is symmetric block diagonal
k=1;
lmiterm([k 1 1 0],0);%P0>0
lmiterm([-k 1 1 P0],1,1);

%no input-saturation
psai=1;
[A0,A1,A2,B,C,D,ns]=closysrp(x,psai);
for i=1:i1
    for j=1:j1
        k=k+1;
        u1=ulbnd(i);u2=u2bnd(j);
        polA=A0+A1*u1+A2*u2;
        lmikrp(polA,P0,k,B,C,D);
    end
end

inpusat=1;
if inpusat==1
    [psailow,psailow1]=inputsat(x);

    i=1;
    for j=1:j1
        k=k+1;
        u1=ulbnd(i);u2=u2bnd(j);
        psai=psailow;
        [A0,A1,A2,B,C,D,ns]=closysrp(x,psai);
        polA=A0+A1*u1+A2*u2;
        lmikrp(polA,P0,k,B,C,D);
    end

    i=i1;
    for j=1:j1
        k=k+1;
        u1=ulbnd(i);u2=u2bnd(j);
        psai=psailow1;
        [A0,A1,A2,B,C,D,ns]=closysrp(x,psai);
        polA=A0+A1*u1+A2*u2;
        lmikrp(polA,P0,k,B,C,D);
    end
end

```

```

end

lmilio=getlmis;
[tmin,xfeas]=gevp(lmilio,k-1,[1.0*exp(-6) 0 0 0 0]);
%[tmin,xfeas]=feasp(lmilio);
gamma=sqrt(tmin)

```

Code 9: closysRS.M

```

function [A0,A1,A2,ns]=closysrs(x,psai)

load C:\ggy2003\model\cstrmat
f0=F0;f1=F1;g1=G1;g2=G2;h0=H0;
[ns,nt]=size(f0);% number of states
Wf=1;bw=0.8;%performance weight
Wt=0.025;gam=Wt*eye(ns);%modeling error

[s,c]=size(x);
kc=x(1);
taui=x(2);
Wc=0;Wd=0;
if c==4
Wc=x(3);
Wd=x(4);
end
ac=1;
bc=1;
cc=kc/taui;
dc=kc+kc/taui;

%psai=1;
A0=[f0-g1*dc*psai*h0 g1*cc*psai;-h0*psai psai];
A1=[f1-(g1*Wd+g2*dc)*psai*h0 (g2*cc+g1*Wc)*psai;zeros(1,nt) 0];
A2=[-g2*Wd*psai*h0 g2*Wc*psai;zeros(1,nt) 0];

```

Code 10: LMIKRS.M

```

%kth single LMI, for robust stability

function lmikrs(polA1,P0,k)

lmiterm([k 1 1 P0],polA1',polA1); %A'*P0*A
lmiterm([k 1 1 P0],[-1,1]); %-P0
lmiterm([-k 1 1 0],0); % A'*P0*A-P0<0

```

Code 11: closysRP.M

```

function [A0,A1,A2,B,C,D,ns]=closysrp(x,psai)

load C:\ggy2003\model\cstrmat;
f0=F0;f1=F1;g1=G1;g2=G2;h0=H0;
[ns,nt]=size(f0);% number of states
Wf=1;bw=0.8;%performance weight
Wt=0.025;gam=Wt*eye(ns);%modeling error

[s,c]=size(x);

```



```

kc=x(1);
taui=x(2);
Wc=0;Wd=0;
if c==4
Wc=x(3);
Wd=x(4);
end
ac=1;
bc=1;
cc=kc/taui;
dc=kc+kc/taui;

%psai=1;
A0=[f0-g1*dc*psai*h0    g1*cc*psai    -g1*dc*Wf*psai;-h0*psai    psai    -
Wf*psai;zeros(1,ns) 0 bw];
A1=[f1-(g1*Wd+g2*dc)*psai*h0    (g2*cc+g1*Wc)*psai    -
(g1*Wd+g2*dc)*Wf*psai;zeros(2,nt+2)];
A2=[-g2*Wd*psai*h0 g2*Wc*psai -g2*Wd*Wf*psai;zeros(2,nt+2)];
B=[zeros(ns,1);0;1-bw];
C=[-h0 0 -Wf];
D=[0];

```

Code 12: LMikRP.M

```

%kth single LMI, for robust performance

function lmikrp(polA1,P0,k,B,C,D)

small=exp(-20);[ns1,ns2]=size(polA1);
small1=eye(ns1)*small;
lmiterm([k 1 1 P0],polA1',polA1);%A'*P0*A
lmiterm([k 1 1 P0],[-1,1]);%-P0
lmiterm([k 1 2 P0],polA1',B);%A'*P0*B
lmiterm([k 1 3 0],C');%C'
lmiterm([k 2 2 P0],B',B);%B'*P0*B
lmiterm([k 2 3 0],D');%D'
lmiterm([k 3 3 0],[-1]);%1
lmiterm([-k 1 1 0],small1);
lmiterm([-k 2 2 0],1);
lmiterm([-k 3 3 0],small);

```

Code 13: Inputsat.M

```

function [psailow,psailow1]=inputsat(x)
length=21;
kc=x(1);
taui=x(2);
Wc=x(3);
Wd=x(4);
Ac=1;Bc=1;
Cc=kc/taui;
Dc=kc+kc/taui;
uk=-1;
for i=1:length
    e(i)=(-1+(i-1)*0.1);
    xc(i)=(uk-(Dc+Wd*uk)*e(i))/(Cc+Wc*uk);

```

```

    xc1=xc(i)+e(i);
    u(i)=(Cc*xc1+Dc*e(i))/(1-Wc*xc1-Wd*e(i));
end
usat=max(abs(u));
psailow=1/usat;

uk0=1;
for i=1:length
    xc0(i)=(uk0-(Dc+Wd*uk0)*e(i))/(Cc+Wc*uk0);
    xc01=xc0(i)+e(i);
    u0(i)=(Cc*xc01+Dc*e(i))/(1-Wc*xc01-Wd*e(i));
end
usat0=max(abs(u0));
psailow1=1/usat0;

```

Code 14: simuPI.M

```

function [u1,y1,ooo]=sisosimuPI(xopt,v,d,sumv,length)

%Variable list
%xopt          Gain-scheduled PI controller parameters
%nx            number of states
%length        Simulation time period
%v             Unmeasured disturbance
%d             Filtered disturbance of v
%sumv         Sum of squared disturbance

load cstrmat;
[nx,nx]=size(F0);
%xopt=[1.3723 2.949 -0.004 0.001];

kcl=xopt(1);tauil=xopt(2);Wcl=xopt(3);Wdl=xopt(4);
Ccl=kcl/tauil;
Dcl=kcl+kcl/tauil;
Ac=1;
Bc=1;

bw=0.8;
w=10;

Wf=1; %disturbance weight
e1=[];e2=[];u1=[];u2=[];ooo=[];
amp=0;%setpoint=0 for analysis, so for comparison, set to 0 as well
ooo(1)=1;
u1(1)=0;u2(1)=0;
xcprev1=0;xcprev2=0;
x1(:,1)=zeros(nx,1);x2(:,1)=x1(:,1);
y1(1)=H0*x1(:,1);y2(1)=H0*x2(:,1);
psail=ones(length,1);psail(1)=1;psai2=1;

for i=1:length
    psail(i)=1;
    ooo(i)=i;
    y1(i)=H0*x1(:,i)+d(i); %gain-scheduling output with disturbance
    e1(i)=amp-y1(i);
    u1(i)=(Ccl*xcprev1+Dcl*e1(i))/(1-Wcl*xcprev1-Wdl*e1(i));

```

```

    if u1(i)>1
        psail(i)=abs(1/u1(i));
        u1(i)=1;
    elseif u1(i)<-1
        psail(i)=abs(1/u1(i));
        u1(i)=-1;
    end
    x1(:,i+1)=(F0+F1.*u1(i))*x1(:,i)+G1.*u1(i)+G2.*u1(i)^2;
    xc1=psail(i)*(Ac*xcprev1+Bc*e1(i));
    xcprev1=xc1;
end

sumerrorgs1=0;
for i=1:length
sumerrorgs1=sumerrorgs1+e1(i)^2;
end
PIgamma=sqrt(sumerrorgs1/sumv)
PIsumu=u1*u1'

```

9.2.2 Parameter-dependent Lyapunov functions

Table 9.3 MATLAB files for Gain-scheduled PI design:parameter-dependent Lyapunov functions

No.	File name	Description
15.	LMIOptP	Optimization of PI parameters K_c, τ_I, W_c, W_d , calls the following function 17
16.	sysPRS	RS, calls 18, 9, and 13
17.	sysPRP	RP, calls 19, 11, and 13
18.	LMikPRS	LMI of RS for each vertex
19.	LMikPRP	LMI of RP for each vertex

Code 15: LMIOptP.M

```

x0=[0.38 0.3 0 0];
[xopt,gopt]=fminsearch('sysprp',x0)

```

Code 16: sysPRS.M

```

clear

%Variable list
%ulbnd      Bounds for uncertainty 1
%u2bnd      Bounds for uncertainty 2
%dulbnd     Bounds for rate of uncertainty 1
%du2bnd     Bounds for rate of uncertainty 2

x=[2.44 1.34 0 0]; %controller parameters,x=[Kc,taui,Wc,Wd]
%uncertainty bounds
ulbnd=[-1,1];
[n1,i1]=size(ulbnd);
dulbnd=[2,-2];

```

```

[n2,i2]=size(dulbnd);
u2bnd=[0,1];
[m1,j1]=size(u2bnd);
du2bnd=[1,-1];
[m2,j2]=size(du2bnd);

%no input-saturation
psai=1;
[A0,A1,A2,ns]=closysrs(x,psai);

%LMI formulation
setlmis([])
P0=lmivar(1,[ns+1 1]);%P0 is symmetric block diagonal
P1=lmivar(1,[ns+1 1]);%P1 is symmetric block diagonal
P2=lmivar(1,[ns+1 1]);%P2 is symmetric block diagonal
k=1;
lmiterm([(4*(k-1)+1) 1 1 0],1);%P0>0
lmiterm([- (4*(k-1)+1) 1 1 P0],1,1);
lmiterm([(4*(k-1)+2) 1 1 0],0);%P1>0
lmiterm([- (4*(k-1)+2) 1 1 P1],1,1);
lmiterm([(4*(k-1)+3) 1 1 0],0);%P2>0
lmiterm([- (4*(k-1)+3) 1 1 P2],1,1);

for i=1:i1
for j=1:j1
for l=1:i2
for m=1:j2
k=k+1;
u1=ulbnd(i);u2=u2bnd(j);
dul=dulbnd(l);du2=du2bnd(m);
polA=A0+A1*u1+A2*u2;
lmikprs(pol1,A0,A1,A2,P0,P1,P2,k,u1,du1,u2,du2)
end
end
end
end

lmilio=getlmis;
[tmin,xfeas]=feasp(lmilio,[]);

```

Code 17: sysPRP.M

```

function gamma=sysprp(x);
x=[2.98 20 0.075 0.075]; %controller parameters,x=[Kc,taui,Wc,Wd]
%uncertainty bounds
ulbnd=[-1,1];
[n1,i1]=size(ulbnd);
dulbnd=[2,-2];
[n2,i2]=size(dulbnd);
u2bnd=[0,1];
[m1,j1]=size(u2bnd);
du2bnd=[1,-1];
[m2,j2]=size(du2bnd);

%no input-saturation
psai=1;

```

```

[A0,A1,A2,B,C,D,ns]=closysrp(x,psai);

%LMI formulation
setlmiis([])
P0=lmivar(1,[ns+2 1]);%P0 is symmetric block diagonal
P1=lmivar(1,[ns+2 1]);%P1 is symmetric block diagonal
P2=lmivar(1,[ns+2 1]);%P2 is symmetric block diagonal
k=1;
lmiterm([(4*(k-1)+1) 1 1 0],0);%P0>0
lmiterm([- (4*(k-1)+1) 1 1 P0],1,1);
lmiterm([(4*(k-1)+2) 1 1 0],0);%P1>0
lmiterm([- (4*(k-1)+2) 1 1 P1],1,1);
lmiterm([(4*(k-1)+3) 1 1 0],0);%P2>0
lmiterm([- (4*(k-1)+3) 1 1 P2],1,1);

for i=1:i1
    for j=1:j1
        k=k+1;
        u1=ulbnd(i);u2=u2bnd(j);
        du1=dulbnd(i);du2=du2bnd(j);
        lmikprp1(A0,A1,A2,P0,P1,P2,k,u1,du1,u2,du2) %convexity condition
    end
end

%performance condition, have to be the last
l=0;
for i=1:i1
    for j=1:j1
        for di=1:i2
            for dj=1:j2
                l=l+1;
                u1=ulbnd(i);u2=u2bnd(j);
                du1=dulbnd(di);du2=du2bnd(dj);
                polA=A0+A1*u1+A2*u2;
                lmikprp(polA,A0,A1,A2,B,C,D,P0,P1,P2,l,u1,du1,u2,du2)
            end
        end
    end
end

lmilios=getlmiis;
[tmin,xfeas]=gevp(lmilios,l,[1.0*exp(-6) 0 0 0 0]);
gamma=sqrt(tmin);

```

Code 18: LMikPRS.M

```

function lmikprs(pol1,A0,A1,A2,P0,P1,P2,k,u1,du1,u2,du2)

lmiterm([(4*(k-1)+1) 1 1 0],[-exp(-20)];%condition 3,i=1,j=0,2
lmiterm([- (4*(k-1)+1) 1 1 P1],A1',A1*(3*u1+du1));
lmiterm([- (4*(k-1)+1) 1 1 P1],A1',A0,'s');
lmiterm([- (4*(k-1)+1) 1 1 P0],A1',A1);
lmiterm([- (4*(k-1)+1) 1 1 P1],A2',A1*u2,'s');
lmiterm([- (4*(k-1)+1) 1 1 P2],A1',A1*u2);
lmiterm([- (4*(k-1)+1) 1 1 P2],A2',A2*du2);
lmiterm([(4*(k-1)+4) 1 1 0],[-exp(-20)];%condition 3,i=2,j=0,1

```

```

lmiterm([- (4*(k-1)+4) 1 1 P2],A2',A2*(3*u2+du2));
lmiterm([- (4*(k-1)+4) 1 1 P2],A2',A0,'s');
lmiterm([- (4*(k-1)+4) 1 1 P0],A2',A2);
lmiterm([- (4*(k-1)+4) 1 1 P2],A2',A1*u1,'s');
lmiterm([- (4*(k-1)+4) 1 1 P1],A2',A2*u1);
lmiterm([- (4*(k-1)+4) 1 1 P1],A2',A2*du1);
lmiterm([(4*(k-1)+2) 1 1 0],0);%condition 2
lmiterm([- (4*(k-1)+2) 1 1 P0],1,1);
lmiterm([- (4*(k-1)+2) 1 1 P1],u1,1);
lmiterm([- (4*(k-1)+2) 1 1 P2],u2,1);
lmiterm([(4*(k-1)+3) 1 1 P0],poll',poll1);%condition1
lmiterm([(4*(k-1)+3) 1 1 P0],-1,1);
lmiterm([(4*(k-1)+3) 1 1 P1],poll',poll1*(u1+du1));
lmiterm([(4*(k-1)+3) 1 1 P1],-1,u1);
lmiterm([(4*(k-1)+3) 1 1 P2],poll',poll1*(u2+du2));
lmiterm([(4*(k-1)+3) 1 1 P2],-1,u2);
lmiterm([- (4*(k-1)+3) 1 1 0],0);

```

Code 19: LMikPRP.M

```
function lmikprp(poll1,A0,A1,A2,B,C,D,P0,P1,P2,k,u1,du1,u2,du2)
```

```

small=exp(-20);[ns1,ns2]=size(poll1);
small1=eye(ns1)*small;
lmiterm([(4*(k-1)+3) 1 1 P0],poll1',poll1);%condition1
lmiterm([(4*(k-1)+3) 1 1 P0],-1,1);
lmiterm([(4*(k-1)+3) 1 1 P1],poll1',poll1*(u1+du1));
lmiterm([(4*(k-1)+3) 1 1 P1],-1,u1);
lmiterm([(4*(k-1)+3) 1 1 P2],poll1',poll1*(u2+du2));
lmiterm([(4*(k-1)+3) 1 1 P2],-1,u2);
lmiterm([(4*(k-1)+3) 1 2 P0],poll1',B);%A'*P0*B
lmiterm([(4*(k-1)+3) 1 2 P1],poll1'*(u1+du1),B);%A'*P0*B
lmiterm([(4*(k-1)+3) 1 2 P2],poll1'*(u2+du2),B);%A'*P0*B
lmiterm([(4*(k-1)+3) 1 3 0],C');%C'
lmiterm([(4*(k-1)+3) 2 2 P0],B',B);%B'*P0*B
lmiterm([(4*(k-1)+3) 2 2 P1],B'*(u1+du1),B);%B'*P0*B
lmiterm([(4*(k-1)+3) 2 2 P2],B'*(u2+du2),B);%B'*P0*B
lmiterm([(4*(k-1)+3) 2 3 0],D');%D'
lmiterm([(4*(k-1)+3) 3 3 0],-1);%1
lmiterm([- (4*(k-1)+3) 1 1 0],small1);
lmiterm([- (4*(k-1)+3) 2 2 0],1);%gamma*gamma
lmiterm([- (4*(k-1)+3) 3 3 0],small);%

```

```
function lmikprp1(A0,A1,A2,P0,P1,P2,k,u1,du1,u2,du2)
```

```

lmiterm([(4*(k-1)+1) 1 1 0],-exp(-20));%condition 3,i=1,j=0
lmiterm([- (4*(k-1)+1) 1 1 P1],A1',A1*(3*u1+du1));
lmiterm([- (4*(k-1)+1) 1 1 P1],A1',A0,'s');
lmiterm([- (4*(k-1)+1) 1 1 P0],A1',A1);
lmiterm([(4*(k-1)+4) 1 1 0],-exp(-20));%condition 3,i=2,j=0,1
lmiterm([- (4*(k-1)+4) 1 1 P2],A2',A2*(3*u2+du2));
lmiterm([- (4*(k-1)+4) 1 1 P2],A2',A0,'s');
lmiterm([- (4*(k-1)+4) 1 1 P0],A2',A2);
lmiterm([- (4*(k-1)+4) 1 1 P2],A2',A1*u1,'s');
lmiterm([- (4*(k-1)+4) 1 1 P1],A2',A2*u1);
lmiterm([- (4*(k-1)+4) 1 1 P1],A2',A2*du1);

```

```

lmiterm([(4*(k-1)+2) 1 1 0],0);%condition 2
lmiterm([- (4*(k-1)+2) 1 1 P0],1,1);
lmiterm([- (4*(k-1)+2) 1 1 P1],u1,1);
lmiterm([- (4*(k-1)+2) 1 1 P2],u2,1);

```

9.2.3 Structured Singular Value

Table 9.4 MATLAB files for Gain-scheduled PI design: SSV

No.	File name	Description
20.	tvssvMAIN	Time-varying μ . Optimization of T and D , calls the following functions 21 and 22
21.	tvssvRS	Robust stability
22.	tvssvRP	Robust performance
23.	tvssvMN	Generate the matrices: M for RS and N for RP

Code 20: tvssvMAIN.M

```

clear

%Variable list
% xPI Gain-scheduled PI controller parameters

xPI=[2 1.1545 -0.1 -0.75];
RS=0;RP=0;%1:fix real uncertainty with complex uncertainty
liner=0;%if 1, linear PI; not 1, G-S PI
[M,Mblk,N,Nblk]=tvssvMN(xPI,RS,RP,liner);
RSRP=1; %1 for RS;2 for RP

if RSRP==1
Tsize=abs(Mblk(1,1));
Dsize=abs(Mblk(2,1));
else
Tsize=abs(Nblk(1,1));
Dsize=abs(Nblk(2,1));
end

x0=rand(Tsize*Tsize+Dsize*Dsize,1);

%x=x0;
save x0 x;
load x0
x0=x;
[x,tv]=fminunc('tvssvRP',x0,[],xPI,RS,RP,liner);

```

Code 21: tvssvRS.M

```

function [tvmu]=tvssvRS(x,xPI,RS,RP,liner)

t=x;%weighting matrix

[M,Mblk,N,Nblk]=tvssvMN(xPI,RS,RP,liner);
Tsize=abs(Mblk(1,1));

```

```

Dsize=abs(Mblk(2,1));
Xsize=Tsize+Dsize;

for i=1:Tsize
for j=1:Tsize
    T(i,j)=t((i-1)*Tsize+j);
end
end

for i=1:Dsize
for j=1:Dsize
    D(i,j)=t(Tsize*Tsize+(i-1)*Dsize+j);
end
end

T1=inv(T);
D1=inv(D);

TD=zeros(Xsize);
TD(1:Tsize,1:Tsize)=T;
TD(Tsize+1:Tsize+Dsize,Tsize+1:Tsize+Dsize)=D;

TD1=zeros(Xsize);
TD1(1:Tsize,1:Tsize)=T1;
TD1(Tsize+1:Tsize+Dsize,Tsize+1:Tsize+Dsize)=D1;

TMT=TD*M*TD1;

tvmu=norm(TMT);%max(abs(svd(ma)))

```

Code 22: tvssvRP.M

```

function [tvmu]=tvssvRP(x,xPI,RS,RP,liner)

t=x;%weighting matrix

[M,Mblk,N,Nblk]=tvssvMN(xPI,RS,RP,liner);
Tsize=abs(Nblk(1,1));
Dsize=abs(Nblk(2,1));
Xsize=Tsize+Dsize+1;

for i=1:Tsize
for j=1:Tsize
    T(i,j)=t((i-1)*Tsize+j);
end
end

for i=1:Dsize
for j=1:Dsize
    D(i,j)=t(Tsize*Tsize+(i-1)*Dsize+j);
end
end

T1=inv(T);
D1=inv(D);

```



```

TD=zeros(Xsize);
TD(1:Tsize,1:Tsize)=T;
TD(Tsize+1:Tsize+Dsize,Tsize+1:Tsize+Dsize)=D;
TD(Xsize,Xsize)=1;

TD1=zeros(Xsize);
TD1(1:Tsize,1:Tsize)=T1;
TD1(Tsize+1:Tsize+Dsize,Tsize+1:Tsize+Dsize)=D1;
TD1(Xsize,Xsize)=1;

TNT=TD*N*TD1;

tvmu=norm(TNT);%max(abs(svd(ma)))

```

Code 23: tvssvMN.M

```

%generate the main matrix
%M for RS, N for RP

function [M,Mblk,N,Nblk]=tvssvMN(xPI,RS,RP,liner)
load C:\ggy2003\model\cstrmat

[ns,nt]=size(F0);% number of states
Wf=1;BW=0.8;%performance weight

%LFT of state-affine
k=1;%number of uncertainties
b22=eye(ns);
c22=F1;
d12=zeros(1,ns);
d21=G2;
d22=zeros(ns);

%LFT of g-s PI
x=xPI; %[3.06 14 0 0];
kc=x(1);
taui=x(2);
Wc=x(3);
Wd=x(4);

ac=1;
bc=1;
cc=kc/taui;
dc=kc+kc/taui;

bc22=0;
cc22=Wc;
dc12=1;
dc21=Wd;
dc22=0;

if liner==1 %linear PI
    %M matrix for robust stability, linear PI
    M11=[F0-G1*dc*H0 G1*cc;-bc*H0 ac];
    M12=[b22-G1*dc*d12;-bc*d12];
    M21=[c22-d21*dc*H0 d21*cc];

```

```

M22=[d22-d21*dc*d12];
M=[M11 M12;M21 M22];
% Uncertainty block for RS, M-delta
%states:x,kesai
%diag[z-1*eye(ns+1) u1*eye(ns)]
Mblk=[ns+1 0;-(ns) 0];
else
%M matrix for robust stability, g-s PI
M11=[F0-G1*dc*H0 G1*cc;-bc*H0 ac];
M12=[b22-G1*dc*d12 G1*dc12;-bc*d12 bc22];
M21=[c22-d21*dc*H0 d21*cc;-dc21*H0 cc22];
M22=[d22-d21*dc*d12 d21*dc12;-dc21*d12 dc22];
M=[M11 M12;M21 M22];
% Uncertainty block for RS, M-delta
%states:x,kesai
%diag[z-1*eye(ns+1) u1*eye(ns) u1]
Mblk=[ns+1 0;-(ns+1) 0];
end

if liner==1
%N matrix for robust performance, linear PI
N11=[F0-G1*dc*H0 G1*cc -G1*dc*Wf;-bc*H0 ac -bc*Wf;zeros(1,ns) 0 BW];
N12=[b22-G1*dc*d12;-bc*d12;zeros(1,ns)];
N13=[zeros(ns,1); 0; 1-BW];
N21=[c22-d21*dc*H0 d21*cc -d21*dc*Wf];
N22=[d22-d21*dc*d12];
N23=[zeros(ns,1)];
N31=[-H0 0 -Wf];
N32=[zeros(1,ns)];
N33=0;
N=[N11 N12 N13;N21 N22 N23;N31 N32 N33];
% Uncertainty block for RP, N-delta
%states:x,kesai,d
%diag[z-1*eye(ns+1+1) u1*eye(ns) deltaRP]
Nblk=[ns+1+1 0;-(ns) 0;1 1];
else
%N matrix for robust performance, g-s PI
N11=[F0-G1*dc*H0 G1*cc -G1*dc*Wf;-bc*H0 ac -bc*Wf;zeros(1,ns) 0 BW];
N12=[b22-G1*dc*d12 G1*dc12;-bc*d12 bc22;zeros(1,ns) 0];
N13=[zeros(ns,1); 0; 1-BW];
N21=[c22-d21*dc*H0 d21*cc -d21*dc*Wf;-dc21*H0 cc22 -dc21*Wf];
N22=[d22-d21*dc*d12 d21*dc12;-dc21*d12 dc22];
N23=[zeros(ns,1) ;0];
N31=[-H0 0 -Wf];
N32=[zeros(1,ns+1)];
N33=0;
N=[N11 N12 N13;N21 N22 N23;N31 N32 N33];
% Uncertainty block for RP, N-delta
%states:x,kesai,d
%diag[z-1*eye(ns+1+1) u1*eye(ns) u1 deltaRP]
Nblk=[ns+1+1 0;-(ns+1) 0;1 1];
end

%if it is required to fix the real uncertain block problem
%RS=0;%1 fix
%RP=1;%1 fix
if RS==1

```

```

Tsize=abs(Mblk(1,1));
Dsize=abs(Mblk(2,1));
Xsize=Tsize+Dsize;
%fix real block with complex block
pdim=Xsize;
belta=0.01;
fixl=[eye(pdim);belta*eye(pdim)];
fixr=fixl';
blk=Mblk;
Mblk=[blk;abs(blk)];
M=mmult(fixl,M,fixr);
end
if RP==1
Tsize=abs(Nblk(1,1));
Dsize=abs(Nblk(2,1));
Xsize=Tsize+Dsize+1;
%fix real block with complex block
pdim=Xsize;
belta=0.01;
fixl=[eye(pdim);belta*eye(pdim)];
fixr=fixl';
blk=Nblk;
Nblk=[blk;abs(blk)];
N=mmult(fixl,N,fixr);
end

```

9.3 Gain-scheduled MPC Controllers Design

The following MATLAB files have been used to design gain-scheduled MPC controllers.

9.3.1 SISO processes

Table 9.5 MATLAB files for Gain-scheduled MPC design: SISO

No.	File name	Description
24.	LMIOptMPC1	Optimization of input weights λ , calls the following functions 26
25.	LMImainRS	RS, calls 27 and 28
26.	LMImainRP	RP, calls 29 and 30
27.	LMIsysRS	Closed-loop system for RS, calls 31
28.	LMIsubRS	LMI of RS for each vertex
29.	LMIsysRP	Closed-loop system for RP, calls 31
30.	LMIsubRP	LMI of RP for each vertex
31.	SISOresponse	Step-response of a SISO process
32.	SISOsimu	Simulate one linear MPC
33.	SISOsimuGS	Simulate one gain-scheduled MPC
34.	SISOsimuGSplot	Compare and plot simulations of more than one G-S MPC, and G-S MPC with G-S PI. It calls 14, 32, and 33.

Code 24: LMIOptMPC1.M

```
tot=4;
step=[-1:2/tot:1];
weit0=0.9*ones(1,tot);

weit=weit0;
save weit0 weit;
load weit0
weit0=weit;
[Weit,gopt]=fminsearch('lmimainrp',weit0,[],tot,step)
```

Code 25: LMImainRS.M

```
clear

%Variable list
%n          Process settling time
%p          Prediction horizon
%m          Control horizon
%tot       Total number of sub-ranges
%step      sub-ranges
%Weit,weiu Input weights

n=12;p=8;m=2;
tot=5;
step=[-1:2/tot:1];
Weit=1*ones(1,tot);

i=1;
step1=step(i);
step2=step(i+1);
weiu=Weit(i);
[A0,A1,ns]=lmsysrs(step1,step2,n,p,m,weiu);

%LMI formulation
setlmis([])
P0=lmivar(1,[ns 1]);%P0 is symmetric block diagonal
lmiterm([1 1 1 0],0);%P0>0
lmiterm([-1 1 1 P0],1,1);

for i=1:tot
    %steps used for H and step-responses
    step1=step(i);
    step2=step(i+1);
    weiu=Weit(i);
    [A0,A1,ns]=lmsysrs(step1,step2,n,p,m,weiu);
    k=i+1;
    eig1=eig(A0+A1*step1);
    eig2=eig(A0+A1*step2);

    lmisubrs(k,P0,A0,A1,step1,step2);
end

lmilio=getlmis;
[tmin,xfas]=feasp(lmilio,[]);
```

```
tmin
```

Code 26: LMImainRP.M

```
function gamma=lmimainrp(Weit,tot,step)

%tot=3;
%step=[-1:2/tot:1];
%Weit=1*ones(1,tot);

n=6;p=4;m=2;weiy=1;

i=1;
step1=step(i);
step2=step(i+1);
weiu=Weit(i);
[A0,A1,B,C,D,H,Au,Kmpc,M,Mp,ns]=lmisysrp(step1,step2,n,p,m,weiu,weiy);
%LMI formulation
setlmis([])
P0=lmivar(1,[ns 1]);%P0 is symmetric block diagonal
k=1;
lmiterm([k 1 1 0],0);%P0>0
lmiterm([-k 1 1 P0],1,1);
f=1;
for i=1:f:tot
    step1=step(i);
    step2=step(i+f);
    weiu=Weit(i);

    [A0,A1,B,C,D,H,Au,Kmpc,M,Mp,ns]=lmisysrp2(step1,step2,n,p,m,weiu,weiy);
    eig1(:,i)=eig(A0+A1*step1);
    eig2(:,i)=eig(A0+A1*step2);
    k=k+1;
    lmisubrp(k,P0,A0,A1,B,C,D,step1,step2);
end

lmilio=getlmis;
load feasov
tmin0=tmin;
xfeas0=xfeas;
[tmin,xfas]=gevp(lmilio,(k-1)*2,[1.0*exp(-6) 1000 0 0 0],tmin0,xfas0);
save feasov tmin xfeas
gamma=sqrt(tmin)%tmin should be smaller than 1
```

Code 27: LMIsysRS.M

```
%closed-loop system matrices for LMI of MPC and state-affine
```

```
function [A0,A1,ns]=lmisysrs(step1,step2,n,p,m,weiu)
load C:\ggy2003\model\cstrmat
[nx,nx]=size(F0);

[h,su]=SISOresponse(step1,step2,n);
su=su';

%close loop formulation of MPC and state-affine
```

```

Wf=1;BW=0.8;
M=zeros(n,n);
for i=1:n-1
    M(i,i+1)=1;
end
M(n,n)=1;
Mp=M(1:p,:);
Wu=weiu*eye(m);%to be designed
Wy=10*eye(p);
Y(:,1)=zeros(n,1);%initial steady-state
du(1)=0;eklk(:,1)=zeros(p,1);eklk(:,2)=zeros(p,1);
wklk(:,1)=zeros(p,1);wklk(:,2)=zeros(p,1);
SU=zeros(p,m);
for i=1:m
    SU(i:p,i)=[su(1:p-i+1)'];
end
mm=zeros(1,m);mm(1,1)=1;
Kmpc=mm*inv(SU'*Wy'*Wy*SU+Wu'*Wu)*SU'*Wy'*Wy;
N2=ones(p,1);
T1=zeros(n,1);T1(1,1)=1;
T2=zeros(n,n);
for i=2:n
    T2(i,i-1)=1;
end
T2(1,1)=1;
e1=T1';
H=zeros(n,n);H(:,1)=su(1:n)';
for i=2:n
    H(1:n-i+1,i)=h(i:n)';
end
KMH=Kmpc*Mp*H;
KNH0=Kmpc*N2*H0;
KNH=Kmpc*N2*e1*H;
KNW=Kmpc*N2*Wf;
Cu2=e1-KMH;
E=T2-T1*KMH;
E2=E+T1*KNH;
Cu1=(e1-KMH+KNH);
B=[zeros(nx,1);zeros(n,1);1-BW];
C=[H0 zeros(1,n) Wf];
D=zeros(1,1);
deltu=0;
A110=(F0)-(G1)*KNH0;
A11=(F1)-(G2)*KNH0;
A120=(G1)*Cu1;
A12=(G2)*Cu1;
A130=-(G1)*KNW;
A13=-(G2)*KNW;
A21=-T1*KNH0;
A22=E2;
A23=-T1*KNW;
A0=[A110 A120;A21 A22];
A1=[A11 A12;zeros(n,nx) zeros(n,n)];
[ns,ns]=size(A0);

```

Code 28: LMISubRS.m

```

function lmisubrs(k,P0,A0,A1,step1,step2)
small=exp(-20);
%case a)
ul=step1;
polA1=A0+A1*ul;
lmiterm([(k-1)*2 1 1 P0],polA1',polA1);%A'*P0*A
lmiterm([(k-1)*2 1 1 P0],[-1,1]);%-P0
lmiterm([(k-1)*2 1 1 0],[-small]);

%case b),
ul=step2;
polA2=A0+A1*ul;
lmiterm([(k-1)*2+1 1 1 P0],polA2',polA2);%A'*P0*A
lmiterm([(k-1)*2+1 1 1 P0],[-1,1]);%-P0
lmiterm([(k-1)*2+1 1 1 0],[-small]);

```

Code 29: LMISYSRP.M

```

%closed-loop system matrices for LMI of MPC and state-affine

function
[A0,A1,B,C,D,H,Au,Kmpc,M,Mp,ns]=lmisysrp(step1,step2,n,p,m,weiu,weiy)
%step1=-1;step2=-0.8;
%n=12;p=8;m=2;weiu=1;
load C:\ggy2003\model\cstrmat
[nx,nx]=size(F0);
Wf=1;BW=0.8;

[h,su]=SISOresponse(step1,step2,n);
su=su';

%close loop formulation of MPC and state-affine
M=zeros(n,n);
for i=1:n-1
    M(i,i+1)=1;
end
M(n,n)=1;
Mp=M(1:p,:);
Wu=weiu*eye(m);%to be designed
Wy=weiy*eye(p);
Y(:,1)=zeros(n,1);%initial steady-state
du(1)=0;eklk(:,1)=zeros(p,1);eklk(:,2)=zeros(p,1);
wk1k(:,1)=zeros(p,1);wk1k(:,2)=zeros(p,1);
SU=zeros(p,m);
for i=1:m
    SU(i:p,i)=[su(1:p-i+1)'];
end
mm=zeros(1,m);mm(1,1)=1;
Kmpc=mm*inv(SU'*Wy'*Wy*SU+Wu'*Wu)*SU'*Wy'*Wy;
N2=ones(p,1);
T1=zeros(n,1);T1(1,1)=1;
T2=zeros(n,n);
for i=2:n
    T2(i,i-1)=1;
end

```

```

T2(1,1)=1;
e1=T1';
H=zeros(n,n);H(:,1)=su(1:n)';
for i=2:n
    H(1:n-i+1,i)=h(i:n)';
end
KMH=Kmpc*Mp*H;
KNH0=Kmpc*N2*H0;
KNH=Kmpc*N2*e1*H;
KNW=Kmpc*N2*Wf;
Cu2=e1-KMH;
E=T2-T1*KMH;
E2=E+T1*KNH;
Cu1=(e1-KMH+KNH);
deltu=0;
A110=(F0)-(G1)*KNH0;
A11=(F1)-(G2)*KNH0;
A120=(G1)*Cu1;
A12=(G2)*Cu1;
A130=-(G1)*KNW;
A13=-(G2)*KNW;
A21=-T1*KNH0;
A22=E2;
A23=-T1*KNW;
A0=[A110 A120 A130;A21 A22 A23;zeros(1,nx) zeros(1,n) BW];
A1=[A11 A12 A13;zeros(n,nx) zeros(n,n) zeros(n,1);zeros(1,nx) zeros(1,n)
0];
[ns,ns]=size(A0);
B=[zeros(nx,1);zeros(n,1);1-BW];
C=[H0 zeros(1,n) Wf];
D=[0];
Au=[-KNH0 Cu1 -KNW];

%controller state-space
Ac=E2;Bc=-T1*Kmpc*N2;Cc=Cu1;Dc=Kmpc*N2;
mpcsys=ss(Ac,Bc,Cc,Dc,1);
[num,den]=ss2tf(Ac,Bc,Cc,Dc);
[z,p,k]=ss2zp(Ac,Bc,Cc,Dc);
pols=pole(mpcsys);

```

Code 30: LMISubRP.M

```

function lmisubrp(k,P0,A0,A1,B,C,D,step1,step2)

%case a)
u1=step1;
polA1=A0+A1*u1;
small=exp(-20);[ns1,ns2]=size(polA1);
small11=eye(ns1)*small;
lmiterm([(k-1)*2 1 1 P0],polA1',polA1);%A'*P0*A
lmiterm([(k-1)*2 1 1 P0],[-1,1]);%-P0
lmiterm([(k-1)*2 1 2 P0],polA1',B);%A'*P0*B
lmiterm([(k-1)*2 1 3 0],C');%C'
lmiterm([(k-1)*2 2 2 P0],B',B);%B'*P0*B
lmiterm([(k-1)*2 2 3 0],D');%D'
lmiterm([(k-1)*2 3 3 0],[-1]);%1

```



```

lmiterm([- (k-1)*2 1 1 0],small1);
lmiterm([- (k-1)*2 2 2 0],1);
lmiterm([- (k-1)*2 3 3 0],small);%

%case b),
u1=step2;
polA2=A0+A1*u1;
lmiterm([(k-1)*2+1 1 1 P0],polA2',polA2);%A'*P0*A
lmiterm([(k-1)*2+1 1 1 P0],-1,1);%-P0
lmiterm([(k-1)*2+1 1 2 P0],polA2',B);%A'*P0*B
lmiterm([(k-1)*2+1 1 3 0],C');%C'
lmiterm([(k-1)*2+1 2 2 P0],B',B);%B'*P0*B
lmiterm([(k-1)*2+1 2 3 0],D');%D'
lmiterm([(k-1)*2+1 3 3 0],-1);%1
lmiterm([- ((k-1)*2+1) 1 1 0],small1);
lmiterm([- ((k-1)*2+1) 2 2 0],1);
lmiterm([- ((k-1)*2+1) 3 3 0],small);

```

Code 31: SIS0response.M

```

function [h,su]=SIS0response(step1,step2,n)

load C:\ggy2003\model\cstrmat
[nx,nx]=size(F0);
length=n+1;
%obtain the steady-state corresponding to step1
x(:,1)=zeros(nx,1);u=step1;
for i=1:length
    y(i)=H0*x(:,i);
x(:,i+1)=(F0+F1.*u)*x(:,i)+G1.*u+G2.*u^2;
end

%step-responses of state-affine
x(:,1)=x(:,length+1);
y0=H0*x(:,1);%+Wf*d(i);

for i=1:length
    u=step2;
    x(:,i+1)=(F0+F1.*u)*x(:,i)+G1.*u+G2.*u^2;
    if (step2-step1)==0
        map=1;
    else map=1/(step2-step1);
    end

    y(i)=map*H0*x(:,i);
    s1(i)=y(i);
end

ss1=s1(1);
s1=s1-ss1;
su=s1(2:length)';

    for i=2:length-1
        hs(i)=su(i)-su(i-1);
    end
    hs(1)=su(1);

```

```
h=hs;
```

Code 32: SISOSimu.M

```
clear

load C:\ggy2003\model\cstrmat
[nx,nx]=size(F0);
tend=50;n=12;p=8;m=2;
length=15+1;
Wf=1;BW=0.8;
weiu=1;
weiy=1;
step1=0;step2=1;
[h,su]=SISOresponse(step1,step2,length);

%setpoints
r=0+zeros(tend+p,1);
%disturbance
[v,d,sumv]=disturbance(tend,BW);

[A0,A1,B,C,D,H,Au,Kmpc,M,Mp,ns]=lmisysrp(step1,step2,n,p,m,weiu,weiy);

N2=ones(p,1);
T1=zeros(n,1);T1(1,1)=1;
T2=zeros(n,n);
for i=2:n
    T2(i,i-1)=1;
end
T2(1,1)=1;
e1=T1';
KMH=Kmpc*Mp*H;
KNH0=Kmpc*N2*H0;
KNH=Kmpc*N2*e1*H;
KNW=Kmpc*N2*Wf;
Cu2=e1-KMH;
E=T2-T1*KMH;
E2=E+T1*KNH;
Cu1=(e1-KMH+KNH);

%initialize
ooo(1)=1;
uc(1)=0;uc(2)=0;
y(1)=0;y(2)=0;
du(1)=0;du(2)=0;
x(:,1)=zeros(nx,1);x(:,2)=zeros(nx,1);

Y(:,1)=zeros(n,1);
ek1k(:,1)=zeros(p,1);ek1k(:,2)=zeros(p,1);
wk1k(:,1)=zeros(p,1);wk1k(:,2)=zeros(p,1);

%MPC design
for k=2:tend
    ooo(k)=k;
    R(:,k+1)=r(k+1:k+p);
    Y(:,k)=M*Y(:,k-1)+su(1:n)*du(k-1);%update the model
```

```

    y(k)=H0*x(:,k)+Wf*d(k);
    wk1k(:,k+1)=ones(p,1)*(y(k)-Y(1,k));
    ek1k(:,k+1)=R(:,k+1)-Mp*Y(:,k)-wk1k(:,k+1);%
    du(k)=Kmpc*ek1k(:,k+1);
    uc(k)=uc(k-1)+du(k);
    u=uc(k);
    x(:,k+1)=(F0+F1.*u)*x(:,k)+G1.*u+G2.*u^2;
end

sumerror=y*y';
LinMPCgamma=sqrt(sumerror/sumv)

```

Code 33: SISOSimuGS.M

```

function [uc,y2,opo]=sisosimugs(step,tot,weit,v,d,sumv,tend)

load C:\ggy2003\model\cstrmat
[nx,nx]=size(F0);
length=15+1;
Wf=1;BW=0.8;weiy=1;
n=12;p=8;m=2;

%tot=5;
%step=[-1:2/tot:1];
%weit=1*ones(1,tot);

%step-responses of state-affine
for j=1:tot
    step1=step(j);%unit step response
    step2=step(j+1);%unit step response
    [hj,suj]=SISOresponse(step1,step2,length);
    hu(j,:)=hj;
    su(j,:)=suj';
end

% MPC design
r=0+zeros(tend+p,1);%setpoints

M=zeros(n,n);
for i=1:n-1
    M(i,i+1)=1;
end
M(n,n)=1;
Mp=M(1:p,:);
Wy=weiy*eye(p);
Y(:,1)=zeros(n,1);%initial steady-state
Y1(:,1)=zeros(n,1);%initial steady-state
du(1)=0;ek1k(:,1)=zeros(p,1);ek1k(:,2)=zeros(p,1);
wk1k(:,1)=zeros(p,1);wk1k(:,2)=zeros(p,1);

SU=zeros(p,m);
for j=1:tot
    for i=1:m
        SU(i:p,i)=[su(j,1:p-i+1)'];
    end
end

```

```

mm=zeros(1,m);mm(1,1)=1;
Wu=weit(j)*eye(m);
Kmpc1(j,:)=mm*inv(SU'*Wy'*Wy*SU+Wu'*Wu)*SU'*Wy'*Wy;
end

ooo(1)=1;
uc(1:n)=zeros(1,n);y2(1:n)=zeros(1,n);
du(1:n)=zeros(1,n);d(1:n)=zeros(1,n);
x(:,1:n)=zeros(nx,n);

U(:,1)=zeros(n,1);
gs=zeros(tend,1);
%MPC design
for k=2:tend
    ooo(k)=k;
    R(:,k+1)=r(k+1:k+p);
    m2=(uc(k-1));
    for i=1:tot
        if (step(i))<=m2 & m2<=(step(i+1))
            gs(k)=i;j=i;
        end
    end
    end
        Kmpc=Kmpc1(j,:);
        j1=j;jj=1;j2=j1+1;

            for i1=2:length-2
                hs(i1)=su(j,i1)-su(j,i1-1);
            end
            hs(1)=su(j,1);
            h=hs;
            H=zeros(n,n);H(:,1)=su(j,1:n)';
            for i2=2:n
                H(1:n-i2+1,i2)=h(i2:n)';
            end

while j1~=j2&jj<10
    j=j1;
    jj=jj+1;
    j2=j1;

Y(:,k)=H*U(:,k-1);

    d(k)=BW*d(k-1)+(1-BW)*v(k-1);%filter disturbance
    y2(k)=H0*x(:,k)+Wf*d(k);
    opo(k)=k;
    wk1k(:,k+1)=ones(p,1)*(y2(k)-Y(1,k));
    ek1k(:,k+1)=R(:,k+1)-Mp*Y(:,k)-wk1k(:,k+1);%-SD*dd(k);
    du(k)=Kmpc*ek1k(:,k+1);
    uc(k)=uc(k-1)+du(k);

    if uc(k)>1 uc(k)=1;end
    if uc(k)<-1 uc(k)=-1;end

    m1=(uc(k));
        for i=1:tot
            if (step(i))<=m1 & m1<=(step(i+1))
                j1=i; Kmpc=Kmpc1(i,:);
            end
        end
end

```

```

        end
    end

    h=hu(j1,:);
    H=zeros(n,n);H(:,1)=su(j1,1:n)';
    for i2=2:n
        H(1:n-i2+1,i2)=h(i2:n)';
    end

    U(2:n,k)=U(1:n-1,k-1);
    U(1,k)=uc(k); dul(k)=uc(k)-uc(k-1);
end
    u=uc(k);
    x(:,k+1)=(F0+F1*u)*x(:,k)+G1*u+G2*u^2;
end
    sumerror=y2*y2';
    MPCgamma=sqrt(sumerror/sumv)
    MPCsumu=uc*uc'
    gssumdu=dul*dul';
    gssum=sumerror+MPCsumu;

figure(1)
plot(opo,uc,'k:',opo,y2,'k')
title('G-S MPC input(:) and output')
axis([0 tend -1 1]);
figure(2)
plot(opo,y2,'k:',opo,v(1:tend),'k')
title('G-S MPC output(:) and disturbance')
axis([0 tend -1 1]);

```

Code 34: sisosimuGSplot.M

```

tend=100;BW=0.8;

[v,d,sumv]=disturbance(tend,BW);
%save dist v d sumv
%load dist

%Simulation of first MPC
tot=4;
step=[-1:2/tot:1];
weit1=[0.6426,0.6616,0.7410,0.7509];
[u1,y1,t1]=sisosimugs(step,tot,weit1,v,d,sumv,tend);
%Simulation of second MPC
%weit2=[10 10 10 10 10];
%[u2,y2,t2]=sisosimugs(step,tot,weit2,v,d,sumv,tend);

%Simulation of gain-scheduled PI
xopt=[1.2168,1.9309,0.1802,0.0009];%[1.4023,3.2087,0.1033,0.0721];
[u2,y2,t2]=sisosimuPI(xopt,v,d,sumv,tend);

figure(1)
plot(t1,u1,'k:',t1,u2,'k')
title('G-S MPC input. MPC1(:),MPC2(-)')
%title('input. G-S MPC(:),G-S PI(-)')

```

```

%axis([0 tend -1 1]);
figure(2)
plot(t1,y1,'k:',t1,y2,'k')
title('G-S MPC output. MPC1(:),MPC2(-)')
%title('output.G-S MPC(:),G-S PI(-)')
%axis([0 tend -1 1]);
figure(3)
plot(v,'k')
title('disturbance')

```

9.3.2 MIMO processes

Table 9.6 MATLAB files for Gain-scheduled MPC design: MIMO

No.	File name	Description
35.	LMIoptMPC2	Optimization of input weights λ , calls the following function 37
36.	MIMOGSLMImainRS	RS, calls 38 and 39
37.	MIMOGSLMImainRP	RP, calls 40 and 41
38.	MIMOLMIsysRS	Closed-loop system for RS
39.	MIMOLMIsubRS	LMI of RS for each vertex
40.	MIMOLMIsysRP	Closed-loop system for RP
41.	MIMOLMIsubRP	LMI of RP for each vertex
42.	MIMOmodel	2*2 state-affine model
43.	MIMOresponse	Step-response of a MIMO process
44.	MIMOsimu	Simulate one linear MPC
45.	MIMOGSsimu	Simulate one linear MPC and one G-S MPC (2-switch)

Code 35: LMIoptMPC2.M

```

nu=2;
weit0=1*ones(1,nu);%for mimolmimainrp
%weit0=0.8*ones(2,nu);%for mimogslmimainrp

weit=weit0;
save weit0 weit;
load weit0
weit0=weit;
[Weit,gopt]=fminsearch('mimolmimainrp',weit0,[],nu)
%[Weit,gopt]=fminsearch('mimogslmimainrp',weit0,[],nu)

```

Code 36: MIMOGSLMImainRS.M

```

clear

nu=2;
Weightu=(0.6)*ones(2,nu);
Weightu
ny=2;ulstep1=0;ulstep2=1;u2step1=0;u2step2=1;
n=6;p=4;m=2;

```

```

weiy=1*ones(1,ny);Wf=[1;0];

MIMOmodel;
weiu=Weightu(1,:);
[A0,A1,A2,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOlmsysrs(nu,ny,n,p,m,weiu,weiy,Wf,
... ulstep1,ulstep2,u2step1,u2step2);
%LMI formulation
setlmis([])
P0=lmivar(1,[ns 1]);%P0 is symmetric block diagonal
Q=lmivar(1,[1 1]);%Q is symmetric block diagonal
k=1;lmitag=[];
l1=1;
lmiterm([1 1 1 0],0);%P0>0
lmiterm([-1 1 1 P0],1,1);

rang1=[-0.3 0;0 0.3];%for uncertain parameter u1
rang2=[-0.3 0;0 0.3];%for uncertain parameter u2
mrang1=[-1 0;0 1];% for model of step-response
mrang2=[-1 0;0 1];
[r g]=size(rang1);

%G-S robust stability
for i=1:r
for j=1:r
k=k+1;
ulstep1=rang1(i,1);
ulstep2=rang1(i,g);
weiu(1)=Weightu(i,1);
u2step1=rang2(j,1);
u2step2=rang2(j,g);
weiu(2)=Weightu(j,2);
mulstep1=mrang1(i,1);
mulstep2=mrang1(i,g);
mu2step1=mrang2(j,1);
mu2step2=mrang2(j,g);

%robust stability

[A0,A1,A2,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOlmsysrs(nu,ny,n,p,m,weiu,weiy,Wf,
...mulstep1,mulstep2,mu2step1,mu2step2);
mimolmisubrs((k-1)*4+1,P0,A0,A1,A2,ulstep1,u2step2);
mimolmisubrs((k-1)*4+2,P0,A0,A1,A2,ulstep2,u2step1);
mimolmisubrs((k-1)*4+3,P0,A0,A1,A2,ulstep2,u2step2);
mimolmisubrs((k-1)*4+4,P0,A0,A1,A2,ulstep1,u2step1);
k=k+1;
end
end

lmilio=getlmis;
[tmin,xfeas]=feasp(lmilio);

```

Code 37: MIMOGSLMImainRP.M

```

clear

function gamma=mimogslmimainrp(Weightu,nu)

```

```

%nu=2;
%Weightu=(0.6)*ones(2,nu);
ny=2;ulstep1=0;ulstep2=1;u2step1=0;u2step2=1;
n=6;p=4;m=2;
weiy=1*ones(1,ny);Wf=[1;0];

MIMOmodel;
weiu=Weightu(1,:);
[A0,A1,A2,B,C,D,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOlmisysrp(nu,ny,n,p,m,weiu,weiy,Wf,...ulstep1,ulstep2,u2step1,u2step2);
%LMI formulation
setlmis([])
P0=lmivar(1,[ns 1]);%P0 is symmetric block diagonal
Q=lmivar(1,[1 1]);%Q is symmetric block diagonal
k=1;limitag=[];
l1=1;
lmiterm([1 1 1 0],0);%P0>0
lmiterm([-1 1 1 P0],1,1);

rangel=[-0.3 0;0 0.3];%for uncertain parameter u1
range2=[-0.3 0;0 0.3];%for uncertain parameter u2
mrangel=[-1 0;0 1];% for model of step-response
mrange2=[-1 0;0 1];
[r g]=size(rangel);

%G-S robust stability
for i=1:r
for j=1:r

    k=k+1;
    ulstep1=rangel(i,1);
    ulstep2=rangel(i,g);
    weiu(1)=Weightu(i,1);
    u2step1=range2(j,1);
    u2step2=range2(j,g);
    weiu(2)=Weightu(j,2);
    mulstep1=mrangel(i,1);
    mulstep2=mrangel(i,g);
    mu2step1=mrange2(j,1);
    mu2step2=mrange2(j,g);

%robust performance

[A0,A1,A2,B,C,D,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOlmisysrp(nu,ny,n,p,m,weiu,weiy,Wf,...mulstep1,mulstep2,mu2step1,mu2step2);
mimolmisubrp((k-1)*4,P0,Q,A0,A1,A2,B,C,D,ulstep1,u2step1);
mimolmisubrp((k-1)*4+1,P0,Q,A0,A1,A2,B,C,D,ulstep1,u2step2);
mimolmisubrp((k-1)*4+2,P0,Q,A0,A1,A2,B,C,D,ulstep2,u2step1);
mimolmisubrp((k-1)*4+3,P0,Q,A0,A1,A2,B,C,D,ulstep2,u2step2);

end
end

lmiterm([(k-1)*4+6 1 1 0],0);%Q>0
lmiterm([- (k-1)*4-6 1 1 Q],1,1);

lmiterm([(k-1)*4+7 1 1 Q],1,1);%Q<gamma^2

```



```

lmiterm([- (k-1)*4-7 1 1 0],1);

lmilio=getlmis;
load feasov %to use initial guess,do not initialize P0
tmin0=tmin;
xfeas0=xfeas;
[tmin,xfeas]=gevp(lmilio,1,[1.0*exp(-2) 50 0 0 0],tmin0,xfeas0);
save feasov tmin xfeas
gamma=sqrt(tmin)

```

Code 38: MIMOLMISysRS.M

```

%MIMO system of 2*2
%closed-loop system matrices for LMI of MPC and state-affine

function
[A0,A1,A2,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOLmisysrs(nu,ny,n,p,m,weiu,weiy,Wf,
...ulstep1,ulstep2,u2step1,u2step2)
%nu=2;ny=2;n=12;p=8;m=2;weiu=ones(1,nu);Wf=[1;0];ulstep1=-
1;ulstep2=0;u2step1=-1;u2step2=0;
load C:\gjjy2003\mpc\MIMOsstateaffine

[nx,nx]=size(F0);
prit=0;
[Su,Hu]=MIMOresponse(prit,nu,ny,n,p,m,ulstep1,ulstep2,u2step1,u2step2);

%sparse diagonal matrix
%M is nny*nny
e0=ones(n*ny,1);
M=spdiags([e0],[ny],n*ny,n*ny);
M(n*ny-1,n*ny-1)=1;
M(n*ny,n*ny)=1;
%Mp is pny*nny
Mp=M(1:p*ny,:);
%Wu is mnu*mnu,weiu is 1*nu,EU is nu*mnu
clear EU;
EU0=eye(nu);EU=EU0;
for i=1:m-1
    EU=[EU EU0];
end
Wu=spdiags([weiu*EU]',[0],m*nu,m*nu);
%Wy is pny*pny
clear EY;
EY0=eye(ny);EY=EY0;
for i=1:p-1
    EY=[EY EY0];
end
Wy=spdiags([weiy*EY]',[0],p*ny,p*ny);
%Y is nny*1
Y(:,1)=zeros(n*ny,1);%initial steady-state

%Su(1:n,1:4) are for s11,s12,s21,s22, step-responses
SU=zeros(n*ny,m*nu);
for i=1:n
    SU((i-1)*ny+1,1)=Su(i,1);
    SU((i-1)*ny+1,nu)=Su(i,2);

```

```

SU(i*ny,1)=Su(i,3);
SU(i*ny,nu)=Su(i,4);
end
for j=2:m
    SU((j-1)*ny+1:n*ny,(j-1)*nu+1:j*nu)=SU(1:(n-(j-1))*ny,1:nu);
end
SU0=SU;
SU=SU(1:p*ny,:);
mm=zeros(nu,m*nu);mm(1:nu,1:nu)=eye(nu);
Kmpc=mm*inv(SU'*Wy'*Wy*SU+Wu'*Wu)*SU'*Wy'*Wy;
N2=EY';
T1=zeros(n*nu,nu);T1(1:nu,1:nu)=eye(nu);
e3=ones(n*nu,1);
T2=spdiags([e3],[-nu],n*nu,n*nu);
T2(1,1)=1;
T2(nu,nu)=1;
e1=T1';
%Hu(1:n,1:4) are for h11,h12,h21,h22, impulse-responses
H=zeros(n*ny,n*nu);H(:,1:nu)=SU0(:,1:nu);
for i=1:n-1
    H((i-1)*ny+1,nu+1)=Hu(i+1,1);
    H((i-1)*ny+1,2*nu)=Hu(i+1,2);
    H(i*ny,nu+1)=Hu(i+1,3);
    H(i*ny,2*nu)=Hu(i+1,4);
end
for j=3:n
    H(1:(n-(j-1))*ny,(j-1)*ny+1:j*nu)=H((j-2)*ny+1:(n-1)*ny,nu+1:2*nu);
end

e2=zeros(ny,n*ny);e2(1:ny,1:ny)=eye(ny);
KMH=Kmpc*Mp*H;
KNH0=Kmpc*N2*H0;
KNH=Kmpc*N2*e1*H;
KN2H=Kmpc*N2*e2*H;
KNW=Kmpc*N2*Wf;
Cu2=e1-KMH;
E=T2-T1*KMH;
E2=E+T1*KN2H;
Cu1=(e1-KMH+KN2H);
Au=[-KNH0 Cu1 -KNW];
deltu=0;
A110=(F0)-(G1)*KNH0;
A11=(F1)-(G2)*KNH0;
A211=F2-G3*KNH0;
A120=(G1)*Cu1;
A12=(G2)*Cu1;
A212=G3*Cul;
A21=-T1*KNH0;
A22=E2;
A0=[A110 A120;A21 A22];
A1=[A11 A12;zeros(n*nu,nx) zeros(n*nu,n*nu)];
A2=[A211 A212;zeros(n*nu,nx) zeros(n*nu,n*nu)];
[ns,ns]=size(A0);

```

Code 39: MIMOLMIsubRS.M

```

function mimolmisubrs(k,P0,A0,A1,A2,u1,u2)
polA1=A0+A1*u1+A2*u2;
small=exp(-20);
lmiterm([k 1 1 P0],polA1',polA1);%A'*P0*A
lmiterm([k 1 1 P0],[-1,1]);%-P0
lmiterm([k 1 1 0],[-small]);

```

Code 40: MIMOLMISysRP.M

```

%MIMO system of 2*2
%closed-loop system matrices for LMI of MPC and state-affine

function
[A0,A1,A2,B,C,D,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOLmisysrp(nu,ny,n,p,m,weiu,weiy,Wf,...
    ulstep1,ulstep2,u2step1,u2step2)
%nu=2;ny=2;n=12;p=8;m=2;weiu=ones(1,nu);Wf=[1;0];ulstep1=-1;ulstep2=0;u2step1=-1;u2step2=0;
load C:\gjy2003\mpc\MIMOsstateaffine

[nx,nx]=size(F0);
prit=0;
[Su,Hu]=MIMOresponse(prit,nu,ny,n,p,m,ulstep1,ulstep2,u2step1,u2step2);

%sparse diagonal matrix
%M is nny*nny
e0=ones(n*ny,1);
M=spdiags([e0],[ny],n*ny,n*ny);
M(n*ny-1,n*ny-1)=1;
M(n*ny,n*ny)=1;
%Mp is pny*nny
Mp=M(1:p*ny,:);
%Wu is mnu*mnu,weiu is 1*nu,EU is nu*mnu
clear EU;
EU0=eye(nu);EU=EU0;
for i=1:m-1
    EU=[EU EU0];
end
Wu=spdiags([weiu*EU]',[0],m*nu,m*nu);
%Wy is pny*pny
clear EY;
EY0=eye(ny);EY=EY0;
for i=1:p-1
    EY=[EY EY0];
end
Wy=spdiags([weiy*EY]',[0],p*ny,p*ny);
%Y is nny*1
Y(:,1)=zeros(n*ny,1);%initial steady-state

%Su(1:n,1:4) are for s11,s12,s21,s22, step-responses
SU=zeros(n*ny,m*nu);
for i=1:n
    SU((i-1)*ny+1,1)=Su(i,1);
    SU((i-1)*ny+1,nu)=Su(i,2);
    SU(i*ny,1)=Su(i,3);
    SU(i*ny,nu)=Su(i,4);

```

```

end
for j=2:m
    SU((j-1)*ny+1:n*ny,(j-1)*nu+1:j*nu)=SU(1:(n-(j-1))*ny,1:nu);
end
SU0=SU;
SU=SU(1:p*ny,:);
mm=zeros(nu,m*nu);mm(1:nu,1:nu)=eye(nu);
Kmpc=mm*inv(SU'*WY'*WY*SU+Wu'*Wu)*SU'*WY'*WY;
N2=EY';
T1=zeros(n*nu,nu);T1(1:nu,1:nu)=eye(nu);
e3=ones(n*nu,1);
T2=spdiags([e3],[-nu],n*nu,n*nu);
T2(1,1)=1;
T2(nu,nu)=1;
e1=T1';
%Hu(1:n,1:4) are for h11,h12,h21,h22, impulse-responses
H=zeros(n*ny,n*nu);H(:,1:nu)=SU0(:,1:nu);
for i=1:n-1
    H((i-1)*ny+1,nu+1)=Hu(i+1,1);
    H((i-1)*ny+1,2*nu)=Hu(i+1,2);
    H(i*ny,nu+1)=Hu(i+1,3);
    H(i*ny,2*nu)=Hu(i+1,4);
end
for j=3:n
    H(1:(n-(j-1))*ny,(j-1)*ny+1:j*ny)=H((j-2)*ny+1:(n-1)*ny,nu+1:2*ny);
end

e2=zeros(ny,n*ny);e2(1:ny,1:ny)=eye(ny);
KMH=Kmpc*Mp*H;
KNH0=Kmpc*N2*H0;
KNH=Kmpc*N2*e1*H;
KN2H=Kmpc*N2*e2*H;
KNW=Kmpc*N2*Wf;
Cu2=e1-KMH;
E=T2-T1*KMH;
E2=E+T1*KN2H;
Cu1=(e1-KMH+KN2H);
Au=[-KNH0 Cu1 -KNW];
deltu=0;
A110=(F0)-(G1)*KNH0;
A11=(F1)-(G2)*KNH0;
A211=F2-G3*KNH0;
A120=(G1)*Cu1;
A12=(G2)*Cu1;
A212=G3*Cu1;
A130=-(G1)*KNW;
A13=-(G2)*KNW;
A213=-(G3)*KNW;
A21=-T1*KNH0;
A22=E2;
A23=-T1*KNW;
A0=[A110 A120 A130;A21 A22 A23;zeros(1,nx) zeros(1,n*nu) BW];
A1=[A11      A12      A13;zeros(n*nu,nx)      zeros(n*nu,n*nu)
zeros(n*nu,1);zeros(1,nx) zeros(1,n*nu) 0];
A2=[A211      A212      A213;zeros(n*nu,nx)      zeros(n*nu,n*nu)
zeros(n*nu,1);zeros(1,nx) zeros(1,n*nu) 0];
B=[zeros(nx,1);zeros(n*nu,1);1-BW];

```

```

C=[H0 zeros(ny,n*nu) Wf];
D=zeros(ny,1);
[ns,ns]=size(A0);

%controller state-space
Ac=E2;Bc=-T1*Kmpc*N2;Cc=Cu1;Dc=Kmpc*N2;
mpcsys=ss(Ac,Bc,Cc,Dc,1);
[num1,den1]=ss2tf(Ac,Bc,Cc,Dc,1);
[num2,den2]=ss2tf(Ac,Bc,Cc,Dc,2);
[z1,p1,k1]=ss2zp(Ac,Bc,Cc,Dc,1);
[z2,p2,k2]=ss2zp(Ac,Bc,Cc,Dc,2);
pols=pole(mpcsys);

```

Code 41: MIMOLMIsubRP.M

```

function mimolmisubrp(k,P0,Q,A0,A1,A2,B,C,D,u1,u2)
polA1=A0+A1*u1+A2*u2;
lmiterm([(k) 1 1 P0],polA1',polA1);%A'*P0*A
lmiterm([(k) 1 1 P0],[-1,1]);%-P0
lmiterm([(k) 1 2 P0],polA1',B);%A'*P0*B
lmiterm([(k) 1 3 0],C');%C'
lmiterm([(k) 2 2 P0],B',B);%B'*P0*B
lmiterm([(k) 2 3 0],D');%D'
lmiterm([(k) 3 3 0],[-1]);%1
lmiterm([- (k) 2 2 Q],1,1);

```

Code 42: MIMOmodel.M

```

load C:\ggy2003\model\cstrmatq1
model01=1;
if model01==1
G1=[G1';0 1]';G2=0.1*[G2';1 0]';G3=[-0.01 -0.0159;-0.0508 -0.0928];
H0=[H0;0 0.1];
else
G1=[G1';0 1]';G2=[G2';1 0]';G3=G2;
H0=[H0;H0];
end

BW=0.8;

save MIMOstateaffine F0 F1 F2 G1 G2 G3 H0 BW

```

Code 43: MIMOresponse.M

```

%closed-loop system matrices for LMI of MPC and state-affine

function
[Su,Hu]=MIMOresponse(prit,nu,ny,n,p,m,ulstep1,ulstep2,u2step1,u2step2)
%prit=3;nu=2;ny=2;n=6;p=4;m=2;
%range1=[-1 1;0 1;-1 0];
%range2=[-1 1;-0.8 1;-1 -0.8];
%ulstep1=range1(prit,1);ulstep2=range1(prit,2);
%u2step1=range2(prit,1);u2step2=range2(prit,2);

MIMOmodel;
load C:\ggy2003\mpc\MIMOstateaffine

```

```

[nx,nx]=size(F0);
length=n+1;

%step-responses of state-affine
% u(1) to y(1),y(2)
%obtain the steady-state corresponding to step1
x(:,1)=zeros(nx,1);u=zeros(nu,1);u(1)=ulstep1;
for i=1:length
    y(:,i)=H0*x(:,i);
    x(:,i+1)=(F0+F1*u(1)+F2*u(2))*x(:,i)+(G1+G2*u(1)+G3*u(2))*u;
end

%step-responses of state-affine
x(:,1)=x(:,length+1);u=zeros(nu,1);u(1)=ulstep2;
for i=1:length
    x(:,i+1)=(F0+F1*u(1)+F2*u(2))*x(:,i)+(G1+G2*u(1)+G3*u(2))*u;
    if (ulstep2-ulstep1)==0
        map=1;
    else
        map=1/(ulstep2-ulstep1);
    end
    y(:,i)=map*H0*x(:,i); %scaled to unit-step response
    Su(i,1)=y(1,i)-y(1,1);
    Su(i,3)=y(2,i)-y(2,1);
end

% u(2) to y(1),y(2)
%obtain the steady-state corresponding to step1
x(:,1)=zeros(nx,1);u=zeros(nu,1);u(2)=u2step1;
for i=1:length
    y(:,i)=H0*x(:,i);
    x(:,i+1)=(F0+F1*u(1)+F2*u(2))*x(:,i)+(G1+G2*u(1)+G3*u(2))*u;
end

x(:,1)=x(:,length+1);u=zeros(nu,1);u(2)=u2step2;
for i=1:length
    x(:,i+1)=(F0+F1*u(1)+F2*u(2))*x(:,i)+(G1+G2*u(1)+G3*u(2))*u;
    if (u2step2-u2step1)==0
        map=1;
    else
        map=1/(u2step2-u2step1);
    end
    y(:,i)=map*H0*x(:,i); %scaled to unit-step response
    Su(i,2)=y(1,i)-y(1,1);
    Su(i,4)=y(2,i)-y(2,1);
end

if prit~=0;
    figure(prit)
    subplot(2,2,1)
    plot(Su(:,1),'k')
    title('Step-response of input 1 to output 1,S11')
    % axis([1 15 -0.2 0.4]);
    subplot(2,2,2)
    plot(Su(:,2),'k')
    title('Step-response of input 2 to output 1,S12')
    % axis([1 15 -0.2 0.4]);
    subplot(2,2,3)

```

```

plot(Su(:,3),'k')
title('Step-response of input 1 to output 2,S21')
% axis([1 15 -0.1 0.2]);
subplot(2,2,4)
plot(Su(:,4),'k')
title('Step-response of input 2 to output 2,S22')
% axis([1 15 -0.1 0.2]);
end

Su(1:n,:)=Su(2:n+1,:);

%impulse response
Hu(1,:)=Su(1,:);
for i=2:n
    Hu(i,:)=Su(i,:)-Su(i-1,:);
end
Hu0(1,:)=zeros(1,4);
Hu0(2:n+1,:)=Hu;
%figure(2)
% subplot(2,2,1)
% plot(Hu0(:,1))
% title('Impulse-response of input 1 to output 1,H11')
% subplot(2,2,2)
% plot(Hu0(:,2))
% title('Impulse-response of input 2 to output 1,H12')
% subplot(2,2,3)
% plot(Hu0(:,3))
% title('Impulse-response of input 1 to output 2,H21')
% subplot(2,2,4)
% plot(Hu0(:,4))
% title('Impulse-response of input 2 to output 2,H22')

```

Code 44: MIMOsimu.M

```

clear

nu=2;ny=2;
n=6;p=4;m=2;
weiu=1*[1 1];weiy=1*ones(1,ny);
Wf=[1;0];pmit=2;ulstep1=-1;ulstep2=1;u2step1=-1;u2step2=1;
MIMOmodel;
load C:\ggy2003\mpc\MIMOstateaffine
[nx,nx]=size(F0);
[A0,A1,A2,B,C,D,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOlmisysrp(nu,ny,n,p,m,weiu,weiy,Wf,ulstep1,ulstep2,u2step1,u2step2);

tend=50;

clear EY;
EY0=eye(ny);EY=EY0;
for i=1:p-1
    EY=[EY EY0];
end
N2=EY';

T1=zeros(n*nu,nu);T1(1:nu,1:nu)=eye(nu);

```

```

e3=ones(n*nu,1);
T2=spdiags([e3],[-nu],n*nu,n*nu);
T2(1,1)=1;
T2(nu,nu)=1;
e1=T1';
e2=zeros(ny,n*ny);e2(1:ny,1:ny)=eye(ny);
KMH=Kmpc*Mp*H;
KNH0=Kmpc*N2*H0;
KNH=Kmpc*N2*e1*H;
KN2H=Kmpc*N2*e2*H;
KNW=Kmpc*N2*Wf;
Cu2=e1-KMH;
E=T2-T1*KMH;
E2=E+T1*KN2H;

r=0+zeros(tend+p,1);%setpoints
for i=3+1:3+n
    r(i)=-0.1;
end
for i=3+n+1:3+2*n
    r(i)=0.1;
end
r=0*r;
%disturbance
[v,d,sumv]=disturbance(tend,BW);

%initialize
Y(:,1)=zeros(n*ny,1);%initial steady-state
du(:,1)=zeros(nu,1);du(:,2)=zeros(nu,1);
ek1k(:,1)=zeros(p*ny,1);ek1k(:,2)=ek1k(:,1);
wk1k(:,1)=zeros(p*ny,1);wk1k(:,2)=wk1k(:,1);
uc(:,1)=zeros(nu,1);uc(:,2)=zeros(nu,1);
x(:,1)=zeros(nx,1);x(:,2)=zeros(nx,1);
y(:,1)=H0*x(:,1);y(:,2)=H0*x(:,2);
ooo(1)=1;

%Simulation of MPC, using the state-affine model from MIMOmodel
for k=2:tend
    ooo(k)=k;
    for i=1:p
        R((i-1)*ny+1:i*ny,k+1)=r(k+1)*ones(ny,1);
    end
    Y(:,k)=M*Y(:,k-1)+SU0(:,1:nu)*du(:,k-1);%update the model
    y(:,k)=H0*x(:,k)+Wf*d(k);
    wk1k(:,k+1)=N2*(y(:,k)-Y(1:ny,k));
    ek1k(:,k+1)=R(:,k+1)-Mp*Y(:,k)-wk1k(:,k+1);
    du(:,k)=Kmpc*ek1k(:,k+1);
    uc(:,k)=uc(:,k-1)+du(:,k);
    u=uc(:,k);
    x(:,k+1)=(F0+F1*u(1))*x(:,k)+(G1+G2*u(1)+G3*u(2))*u;
end

sumel=y(1,:)*y(1,:);
sume2=y(2,:)*y(2,:);
gammasimu=sqrt((sumel+sume2)/sumv)

```



```

    clsume1=ycl(1,:)*ycl(1,:);
    clsume2=ycl(2,:)*ycl(2,:);
    clgammasimu=sqrt((clsume1+clsume2)/sumv)

    figure(1)
    subplot(2,2,1)
    plot(ooo,y(1,:), 'k',ooo,v, 'k:')
    title(' design output 1,disturbance(:)')
    subplot(2,2,2)
    plot(ooo,y(2,:), 'k',ooo,v, 'k:')
    title('design output 2,disturbance(:)')
    subplot(2,2,3)
    plot(uc(1,:), 'k')
    title('design input 1')
    subplot(2,2,4)
    plot(uc(2,:), 'k')
    title('design input 2')

```

Code 45: MIMOGSSimu.M

```

clear

nu=2;ny=2;
n=6;p=4;m=2;weiy=1*ones(1,ny);
Wf=[1;0];pmit=0;
MIMOmodel;
load C:\ggy2003\mpc\MIMOstateaffine
[nx,nx]=size(F0);
tend=500;

clear EY;
EY0=eye(ny);EY=EY0;
for i=1:p-1
    EY=[EY EY0];
end
N2=EY';

r=0+zeros(tend+p,1);%setpoints
for i=3+1:3+n
    r(i)=-0.1;
end
for i=3+n+1:3+2*n
    r(i)=0.1;
end
r=0*r;
%disturbance
%[v,d,sumv]=disturbance(tend,BW);
save compargslin v d sumv
load compargalin
%initialize for linear MPC
Y(:,1)=zeros(n*ny,1);%initial steady-state
du(:,1)=zeros(nu,1);du(:,2)=zeros(nu,1);
eklk(:,1)=zeros(p*ny,1);eklk(:,2)=eklk(:,1);
wk1k(:,1)=zeros(p*ny,1);wk1k(:,2)=wk1k(:,1);
uc(:,1)=zeros(nu,1);uc(:,2)=zeros(nu,1);
x(:,1)=zeros(nx,1);x(:,2)=zeros(nx,1);

```

```

y(:,1)=H0*x(:,1);y(:,2)=H0*x(:,2);
ooo(1)=1;
%initialize for G-S MPC
Yg(:,1)=zeros(n*ny,1);%initial steady-state
dug(:,1)=zeros(nu,1);dug(:,2)=zeros(nu,1);
ek1kg(:,1)=zeros(p*ny,1);ek1kg(:,2)=ek1k(:,1);
wk1kg(:,1)=zeros(p*ny,1);wk1kg(:,2)=wk1k(:,1);
ucg(:,1)=zeros(nu,1);ucg(:,2)=zeros(nu,1);
xg(:,1)=zeros(nx,1);xg(:,2)=zeros(nx,1);
yg(:,1)=H0*xg(:,1);yg(:,2)=H0*xg(:,2);

%linear MPC design,using the state-affine model from MIMOmodel
ulstep1=-1;ulstep2=1;u2step1=-1;u2step2=1;weiu=[1 1];%[0.5009 0.4983];%;
[A0,A1,A2,B,C,D,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOlmisysrp(nu,ny,n,p,m,weiu,we
iy,Wf,ulstep1,ulstep2,u2step1,u2step2);
for k=2:tend
    ooo(k)=k;
    for i=1:p
        R((i-1)*ny+1:i*ny,k+1)=r(k+1)*ones(ny,1);
    end
    Y(:,k)=M*Y(:,k-1)+SU0(:,1:nu)*du(:,k-1);%update the model
    y(:,k)=H0*x(:,k)+Wf*d(k);
    wk1k(:,k+1)=N2*(y(:,k)-Y(1:ny,k));
    ek1k(:,k+1)=R(:,k+1)-Mp*Y(:,k)-wk1k(:,k+1);
    du(:,k)=Kmpc*ek1k(:,k+1);
    uc(:,k)=uc(:,k-1)+du(:,k);

% input-saturation limits
for j=1:nu
    if uc(j,k)>1
        uc(j,k)=1;
    elseif uc(j,k)<-1
        uc(j,k)=-1;
    end
end
% end of input-saturation limits
u=uc(:,k);
x(:,k+1)=(F0+F1*u(1)+F2*u(2))*x(:,k)+(G1+G2*u(1)+G3*u(2))*u;
end

%G-S MPC design,using the state-affine model from MIMOmodel
%ulstep1=-1;ulstep2=1;u2step1=-1;u2step2=1;weiu=1*[1 1];
%
[A0,A1,A2,B,C,D,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOlmisysrp(nu,ny,n,p,m,weiu,..
.weiy,Wf,ulstep1,ulstep2,u2step1,u2step2);
for k=2:tend
    for i=1:p
        R((i-1)*ny+1:i*ny,k+1)=r(k+1)*ones(ny,1);
    end
    Yg(:,k)=M*Yg(:,k-1)+SU0(:,1:nu)*dug(:,k-1);%update the model
    yg(:,k)=H0*xg(:,k)+Wf*d(k);
    wk1kg(:,k+1)=N2*(yg(:,k)-Yg(1:ny,k));
    ek1kg(:,k+1)=R(:,k+1)-Mp*Yg(:,k)-wk1kg(:,k+1);
    dug(:,k)=Kmpc*ek1kg(:,k+1);
    ucg(:,k)=ucg(:,k-1)+dug(:,k);

%input-saturation limits

```

```

%% for j=1:nu
% if ucg(j,k)>1
%     ucg(j,k)=1;
% elseif ucg(j,k)<-1
%     ucg(j,k)=-1;
% end
%end
%end of input-saturation limits

    ug=ucg(:,k);ug1=ucg(1,k);ug2=ucg(2,k);
Weight=[0.5164, 0.5029,0.4980,0.5034];
    xg(:,k+1)=(F0+F1*ug1+F2*ug2)*xg(:,k)+(G1+G2*ug1+G3*ug2)*ug;

    if ug1<0
        ulstep1=-1;ulstep2=0;weiu=Weight(1)*[1 1];
    else
        ulstep1=0;ulstep2=1;weiu=Weight(2)*[1 1];
    end
    if ug2<0
        u2step1=-1;u2step2=0;weiu=Weight(3)*[1 1];
    else
        u2step1=0;u2step2=1;weiu=Weight(4)*[1 1];
    end
[A0,A1,A2,B,C,D,Au,Kmpc,ns,H,M,Mp,SU0]=MIMOlmisysrp(nu,ny,n,p,m,weiu,...
weiy,Wf,ulstep1,ulstep2,u2step1,u2step2);
end

%linear
sumel=y(1,:)*y(1,:);sume2=y(2,:)*y(2,:);
sumul=uc(1,:)*uc(1,:);sumu2=uc(2,:)*uc(2,:);
Lingamma=sqrt((sumel+sume2)/sumv)
Linsumu=sumul+sumu2
%G-S
gsumel=yg(1,:)*yg(1,:);gsume2=yg(2,:)*yg(2,:);
gsumul=ucg(1,:)*ucg(1,:);gsumu2=ucg(2,:)*ucg(2,:);
GSgamma=sqrt((gsumel+gsume2)/sumv)
GSsumu=gsumul+gsumu2

%linear vs gain-scheduled
figure(2)
subplot(2,2,1)
plot(ooo,y(1,:), 'k',ooo,yg(1,:), 'k:')
title('output 1.linear(-),G-S(:)')
subplot(2,2,2)
plot(ooo,y(2,:), 'k',ooo,yg(2,:), 'k:')
title('output 2.linear(-),G-S(:)')
subplot(2,2,3)
plot(ooo,uc(1,:), 'k',ooo,ucg(1,:), 'k:')
title(' input 1.linear(-),G-S(:)')
subplot(2,2,4)
plot(ooo,uc(2,:), 'k',ooo,ucg(2,:), 'k:')
title('input 2.linear(-),G-S(:)')

aaa=100;
figure(1)
subplot(2,2,1)
plot(ooo(1:aaa),y(1,1:aaa), 'k',ooo(1:aaa),yg(1,1:aaa), 'k:')

```

```

title('output 1.linear(-),G-S(:)')
subplot(2,2,2)
plot(ooo(1:aaa),y(2,1:aaa),'k',ooo(1:aaa),yg(2,1:aaa),'k:')
title('output 2.linear(-),G-S(:)')
subplot(2,2,3)
plot(ooo(1:aaa),uc(1,1:aaa),'k',ooo(1:aaa),ucg(1,1:aaa),'k:')
title(' input 1.linear(-),G-S(:)')
subplot(2,2,4)
plot(ooo(1:aaa),uc(2,1:aaa),'k',ooo(1:aaa),ucg(2,1:aaa),'k:')
title('input 2.linear(-),G-S(:)')

figure(3)
subplot(2,1,1)
plot(ooo(1:aaa),v(1:aaa),'k')
    title('disturbance: part')
    subplot(2,1,2)
plot(ooo,v,'k')
    title('disturbance: whole')

%plot gain-scheduled MPC only
gs=0;
if gs==1
figure(4)
    subplot(2,2,1)
    plot(ooo,yg(1,:), 'k:',ooo,v,'k')
    title(' output 1, G-S(:),disturbance(:)')
    subplot(2,2,2)
    plot(ooo,yg(2,:), 'k:',ooo,v,'k')
    title('output 2, G-S(:),disturbance(:)')
    subplot(2,2,3)
    plot(ooo,ucg(1,:), 'k:')
    title(' input 1, G-S(:)')
    subplot(2,2,4)
    plot(ooo,ucg(2,:), 'k:')
    title('input 2, G-S(:)')
end

```

10 Appendix C: Nomenclature

English symbols	
$\mathbf{A}(\delta_t), \mathbf{B}, \mathbf{C}, \mathbf{D}$	Closed-loop system matrices
BW	Bandwidth weight, $0 \leq BW \leq 1$
$d(t)$	The filtered unmeasured disturbance
$e(t)$	The output error
$\mathbf{F}_i, \mathbf{G}_i, \mathbf{H}_i$	State-affine model matrices containing model coefficients
$h_0, h_1, h_2, \dots, h_n$	Impulse response coefficients
K_c	Proportional gain
\tilde{K}_c	Variable gain to deal with input-saturation
\mathbf{K}_{MPC}	MPC controller function
m	Control horizon
n	Settling time
p	Prediction horizon
\mathbf{P}	Lyapunov matrix $\mathbf{P} = \mathbf{P}^T > \mathbf{0}$
$\mathbf{R}(t)$	Set-point trajectory, $\mathbf{R}(t) = [r(t+1) \ r(t+2) \ \dots \ r(t+p)]^T$.
\mathbf{S}	Parameter box $\mathbf{S} := \{(\tau_1, \tau_2, \dots, \tau_n) : \tau_i \in \{\underline{v}_i, \overline{v}_i\}\}$
$S_0^u, S_1^u, S_2^u, \dots, S_n^u$	Step response coefficients
\mathbf{S}^u	Step response matrix
$u(t)$	Control action calculated with saturation limits
$\mathbf{U}(t-1)$	MPC controller state, $\mathbf{U}^T(t-1) = [\mathbf{u}(t-1) \ \mathbf{u}(t-2) \ \dots \ \mathbf{u}(t-n)]_{1 \times nm_u}$
$\hat{u}(t)$	Control action calculated without saturation limits
$V(t)$	Quadratic Lyapunov function $V(t) = \boldsymbol{\eta}(t)^T \mathbf{P} \boldsymbol{\eta}(t)$
\mathbf{W}	Parameter box $\mathbf{W} := \{(\omega_1, \omega_2, \dots, \omega_n) : \omega_i \in \{\underline{\delta}_i, \overline{\delta}_i\}\}$

$\mathbf{W}(t+1/t)$	A vector defined to represent the unmeasured disturbance and model/plant mismatch
W_c, W_d	The controller tuning parameters
W_f	Disturbance weight, $0 \leq W_f \leq 1$
W_t	Magnitude of the modeling error
$\mathbf{x}(t)$	The process states
$y(t)$	The output
$\bar{\mathbf{Y}}(t)$	Model update vector.
$\bar{\mathbf{Y}}(t+1/t)$	p – step-ahead prediction vector

Greek symbols	
γ	Performance index
$\mathbf{\Gamma}$	Weighting matrices y
$\underline{\delta}_i, \bar{\delta}_i$	Lower and upper bounds of uncertain parameter $\delta_{i,t}$, $\delta_{i,t} \in [\underline{\delta}_i, \bar{\delta}_i]$
$\delta_{i,t}$	Uncertain parameter, $\delta_{i,t} = u(t)^i$
δ_t	Modeling error in the output
δ_t	Uncertain parameter vector $\delta_t = (\delta_{1,t}, \delta_{2,t}, \dots, \delta_{n,t}) \in \mathbf{R}^n$
$\Delta \in \mathbf{C}^{n \times n}$	Uncertainty block $\Delta = \{diag[\delta_1 \mathbf{I}_{r_1}, \dots, \delta_s \mathbf{I}_{r_s}, \Delta_1, \dots, \Delta_f] : \delta_i \in \mathbf{C}, \Delta_j \in \mathbf{C}^{m_j \times m_j}\}$
$\Delta \delta_{i,t}$	Uncertain parameter time-variation, $\Delta \delta_{i,t} = \delta_{i,t+1} - \delta_{i,t}$
$\Delta \delta_t$	Rate of variation, $\Delta \delta_t = (\Delta \delta_{1,t}, \Delta \delta_{2,t}, \dots, \Delta \delta_{n,t}) \in \mathbf{R}^n$
$\Delta u(t)$	First control move, $u(t) = u(t-1) + \Delta u(t)$
$\boldsymbol{\varepsilon}(t+1/t)$	Feedback corrected vector of future output deviations from the reference trajectory, $\boldsymbol{\varepsilon}(t+1/t) = [\varepsilon(t+1/t) \quad \varepsilon(t+2/t) \quad \dots \quad \varepsilon(t+p/t)]^T$

$\eta(t)$	Closed-loop system state
Λ	Weighting matrices for u
μ	Structured singular value of a matrix
$v(t)$	The unfiltered unmeasured disturbance
$\underline{v}_i, \bar{v}_i$	Lower and upper bounds of uncertain parameter $\Delta\delta_{i,t}, \Delta\delta_{i,t} \in [\underline{v}_i \quad \bar{v}_i]$
$\xi(t)$	The PI controller state
$\rho(\cdot)$	Spectral radius of a matrix, i.e. the largest absolute value of the matrix's eigenvalues
$\bar{\sigma}(\cdot)$	The largest absolute value of the matrix's singular values
τ_I	Reset time
ψ	Input-saturation factor with $\psi \in [\underline{\psi} \quad \bar{\psi}]$, where $\underline{\psi}$ is the lower bound, and $\bar{\psi}$ is the upper bound.

11 Appendix D: Defense Presentation Slides

Robust Control Design of Gain-scheduled Controllers for Nonlinear Processes

Jianying (Meg) Gao

Supervisor: Professor Hector M. Budman
Department of Chemical Engineering



1

Outline

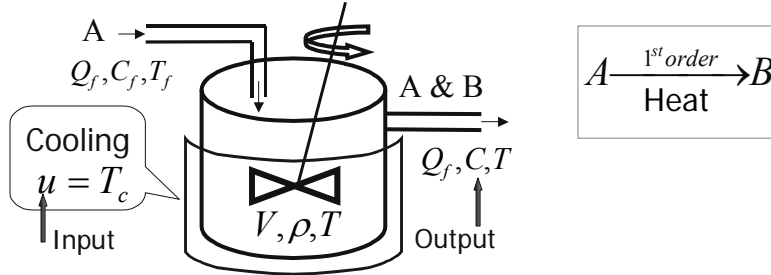
1. Motivation
2. Model Nonlinear Processes
3. Robust Gain-scheduled Proportional-Integral Controller Design (SISO)
4. Robust Gain-scheduled Model Predictive Controller Design (SISO)
5. Conclusions and Future Directions



2

Nonlinear Process: CSTR

- Continuous Stirred Tank Reactor: CSTR



- Mass Balance:
$$V \frac{dC}{dt} = Q_f (C_f - C) - VkC$$

- Nonlinear process: **Arrhenius**

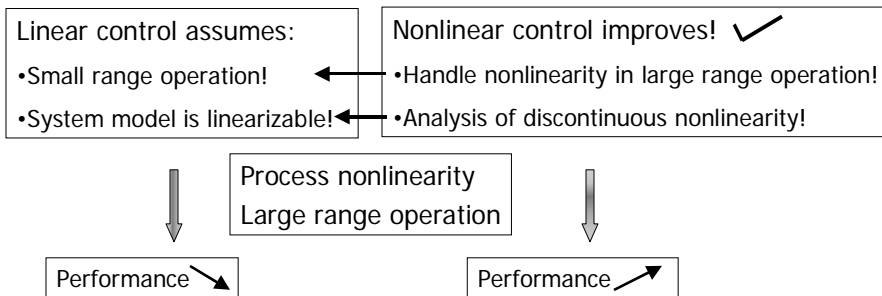
$$k(T) = e^{\frac{-E}{RT}}$$



3

Motivation

- Problem: chemical or bio-chemical processes are highly nonlinear! High-performance controllers are desired!
- Solutions: linear control v.s. nonlinear control



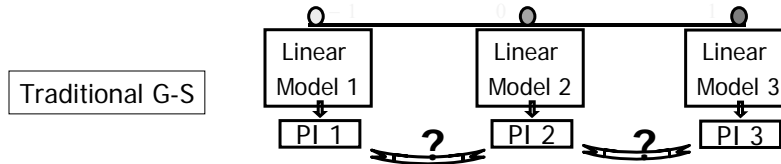
- Conclusion: Nonlinear control, e.g. gain-scheduling!



4

Gain-Scheduling Control

- Successful design approach for nonlinear processes!



- However, no guarantee of global stability and performance! (Shamma & Athans, 1990)
- How to guarantee global stability and performance?
 - Nonlinear approach: Find a Lyapunov function for a nonlinear model!
 - ✓ Robust control approach: nonlinear model=linear+uncertainty!
 - ✓ Non-traditional gain-scheduling: no local linearization!



5

Objective and Novelties

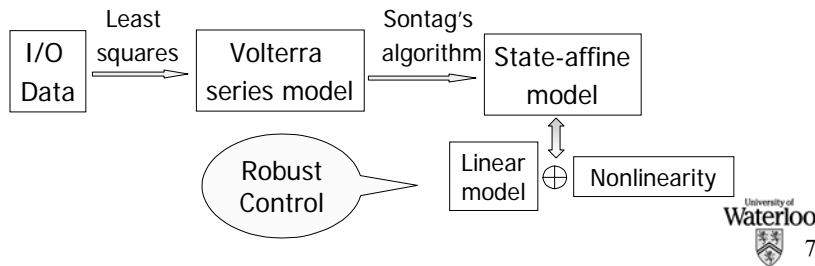
- Objective: Propose a comprehensive design procedure for gain-scheduled controllers, such that Robust Stability and Robust Performance are guaranteed!
- Novelties:
 1. Model nonlinear process as linear+uncertainty!
 2. Guarantee RS & RP
 - RS and RP conditions
 - Reduce conservatism: Parameter-dependent Lyapunov functions
 - Reduce conservatism: Relaxation of input-saturation factor
 - Empirical modeling:
 3. Robust gain-scheduling: PI and MPC



6

Model Nonlinear Process

- First-principles model:
 - Difficult to identify: kinetics, time-variation
 - Impractical to use: dimension, structure
- Empirical model: ✓
 - Easy to identify: experimental I/O data
 - Choose model structure
 - Volterra series model is a classical nonlinear empirical model



Volterra Series Model

- Advantages ✨
 - Black box model for nonlinear process from I/O
 - Linear least squares algorithm

$$y(t) = \sum_{i=1}^{\infty} h_i u(t-i) +$$

\leftarrow
 1st order
 linear

$$\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} h_{ij} u(t-i)u(t-j) +$$

\leftarrow
 2nd order

$$\sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} h_{ijk} u(t-i)u(t-j)u(t-k) + \dots$$

\leftarrow
 3rd order

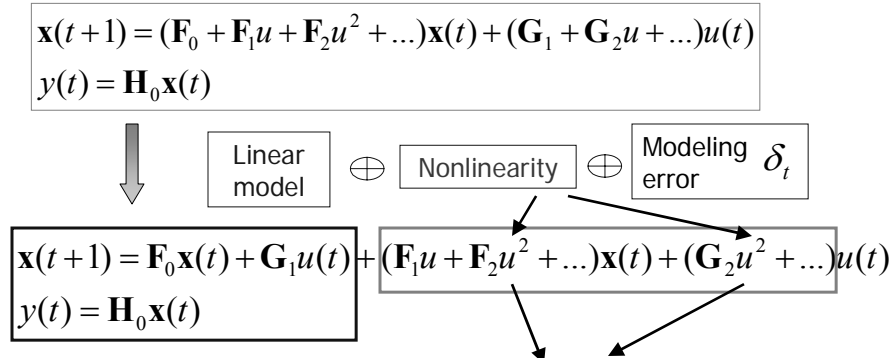
➤ Disadvantages ☾
 • Not suitable for robust control approach

• The output depends on past inputs raised to different powers and in different product combinations, $u(t-1)^2, u(t-2)u(t-3)$

8

State-affine Model

- Nonlinear state-affine model



- Deal with the **Nonlinearity** as **Uncertainty!**

$\delta_{i,t} = u(t)^i, \delta_{1,t} = u(t), \delta_{2,t} = u(t)^2$



9

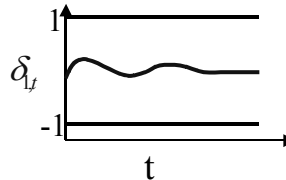
Uncertainty Quantification 1

- **Assumption 1:** Each uncertain parameter is bounded

$\delta_{i,t} = u(t)^i, \delta_{1,t} = u(t), \delta_{2,t} = u(t)^2$

$u(t) \in [-1 \ 1] \rightarrow \delta_{1,t} \in [-1 \ 1]$

$u(t) \in [\underline{u} \ \bar{u}] \rightarrow \delta_{i,t} \in [\underline{\delta}_i, \bar{\delta}_i]$



- Uncertainty can be bounded much easier!
- Convexity: Parameter vector is valued in a hyper-rectangle called the parameter box \mathbf{W}

$\mathbf{W} := \{(\omega_1, \omega_2, \dots, \omega_n) : \omega_i \in [\underline{\delta}_i, \bar{\delta}_i]\}$



10

Uncertainty Quantification 2

- Time-variation of the uncertain parameters is available!

$$\Delta\delta_{i,t} = \delta_{i,t+1} - \delta_{i,t}$$

- **Assumption 2:** Each uncertain parameter-variation is bounded

$$\Delta\delta_{i,t} \in [\underline{v}_i \quad \bar{v}_i]$$

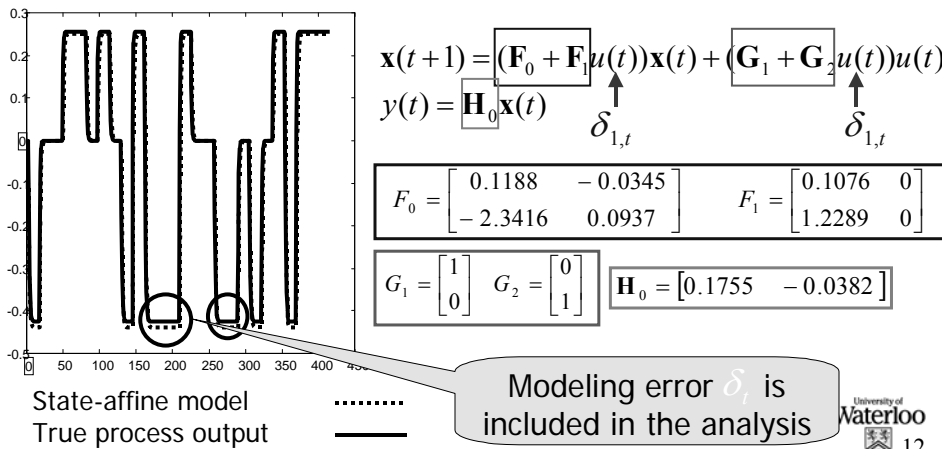
- Convexity: Parameter vector is valued in a hyper-rectangle called the parameter box S

$$S := \{(\tau_1, \tau_2, \dots, \tau_n) : \tau_i \in [\underline{v}_i, \bar{v}_i]\}$$



Results 1 (modeling)

- State-affine model for CSTR $\mathbf{F}_i, \mathbf{G}_i, \mathbf{H}_0$
 - I/O data \rightarrow Volterra series model \rightarrow state-affine model



G-S PI Design

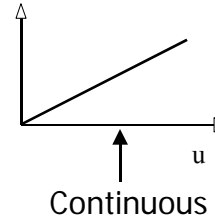
- Gain-Scheduled Proportional-Integral Controller

$$\xi(t+1) = A_c \xi(t) + B_c e(t)$$

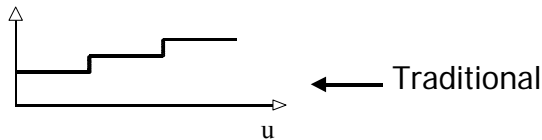
$$\hat{u}(t) = \left(C_c + \sum_{i=1}^m W_{ci} u^i(t) \right) \xi(t) + \left(D_c + \sum_{i=1}^m W_{di} u^i(t) \right) e(t)$$

$$e(t) = y_d(t) - y(t), y_d(t) = 0$$

$$A_c = 1, B_c = 1, C_c = \frac{K_c}{\tau_I}, D_c = K_c + \frac{K_c}{\tau_I}$$



- Design parameters: K_c, τ_I, W_c, W_d
- Traditional gain-scheduling: linearization



13

Closed-loop System: APS

- Closed-loop system: affine parameter-dependent system

$$\begin{bmatrix} \eta(t+1) \\ e(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}(\delta_t) & \mathbf{B}(\delta_t) \\ \mathbf{C}(\delta_t) & \mathbf{D}(\delta_t) \end{bmatrix} \begin{bmatrix} \eta(t) \\ v(t) \end{bmatrix}, \eta(0) = \eta_0$$

Global RS and RP \leftarrow

- **Assumption 3:** Affine dependence on the uncertain parameters

$$\mathbf{A}(\delta_t) = \mathbf{A}_0 + \mathbf{A}_1 \delta_{1,t} + \mathbf{A}_2 \delta_{2,t} \dots + \mathbf{A}_n \delta_{n,t}$$

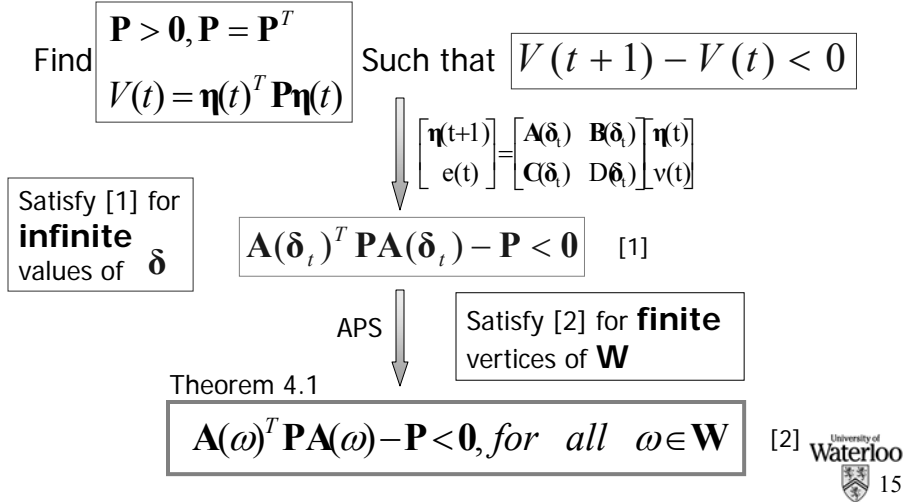
- Important: reduces infinite problem to a finite set of LMIs (Linear Matrix Inequalities)!



14

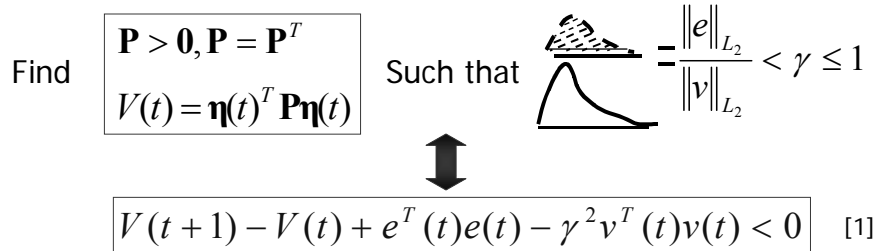
Robust Stability (RS)

➤ Quadratic Lyapunov stability (QLS)



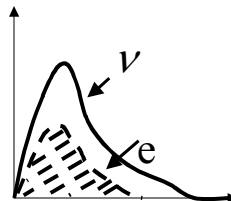
Robust Performance (RP)

➤ Quadratic Lyapunov performance (QLP)



➤ Disturbance rejection

- Performance index: γ
- It can be optimized!



Robust Performance (RP)

➤ Quadratic Lyapunov performance (QLP)

$$\begin{bmatrix} \eta(t+1) \\ e(t) \end{bmatrix} = \begin{bmatrix} A(\delta_t) & B(\delta_t) \\ C(\delta_t) & D(\delta_t) \end{bmatrix} \begin{bmatrix} \eta(t) \\ v(t) \end{bmatrix} \quad \begin{bmatrix} A(\delta_t)^T P A(\delta_t) - P & A(\delta_t)^T P B & C^T \\ B^T P A(\delta_t) & B^T P B - \gamma^2 I & D^T \\ C & D & -I \end{bmatrix} < \mathbf{0} \quad [1]$$

Satisfy [1] for **infinite** values of δ

APS

Satisfy [2] for **finite** vertices of \mathbf{W}

Theorem 4.2

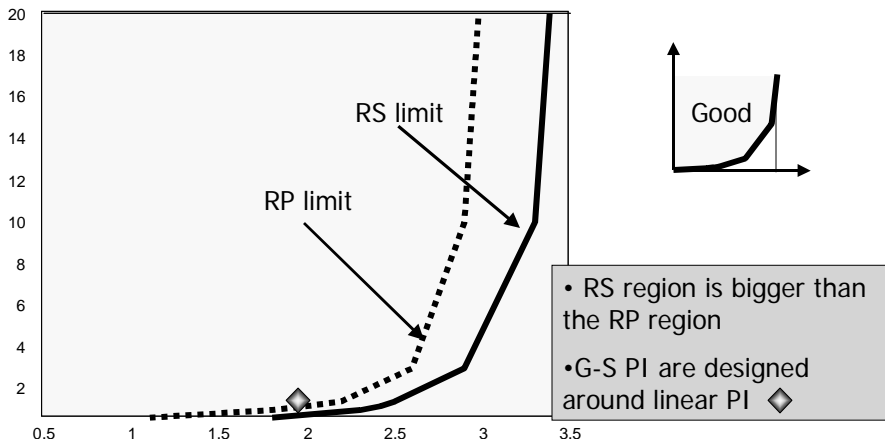
$$\begin{bmatrix} A(\omega)^T P A(\omega) - P & A(\omega)^T P B & C^T \\ B^T P A(\omega) & B^T P B - \gamma^2 I & D^T \\ C & D & -I \end{bmatrix} < \mathbf{0} \quad [2]$$

for all $\omega \in \mathbf{W}$



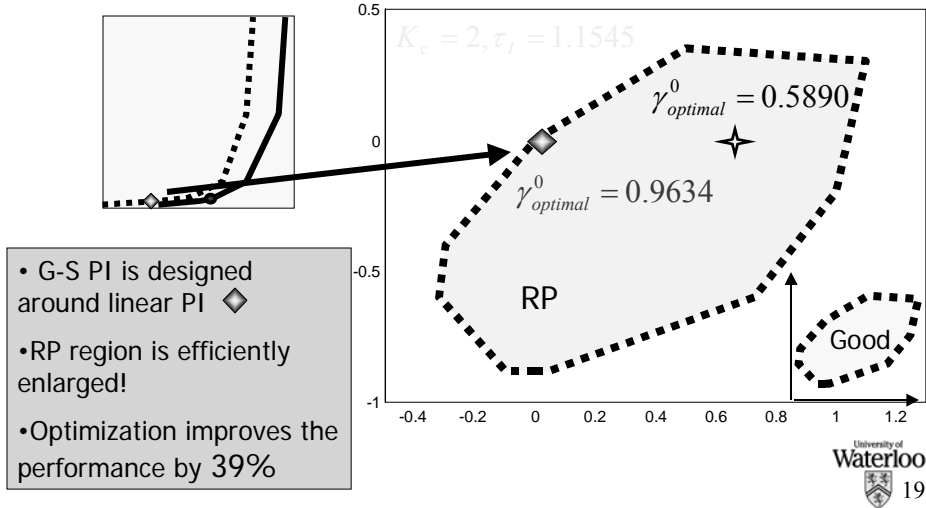
Results 2 (Linear PI RS&RP)

➤ Figure 5.2: RS and RP regions ($W_c = W_d = 0$)



Results 2 (G-S PI RP)

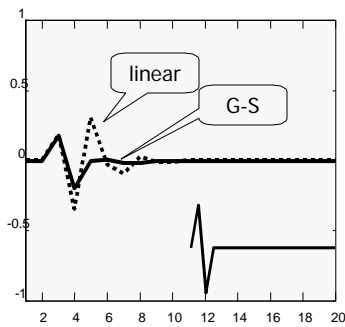
➤ Figure 5.3: Improve over linear PI



Results 2 (PI Simulation)

➤ Table 5.1

PI	K_c	τ_I	W_c	W_d	$\gamma_{optimal}$	$\gamma_{simulation}$
Linear \diamond	2	1.1545	0	0	0.9634	0.3787
G-S PI	1.2168	1.9309	0.1802	0.009	0.3204	0.2025



RS (parameter-dependent)

➤ Affine quadratic Lyapunov stability (AQLS)

Find
$$\begin{aligned} \mathbf{P}(\delta_t) &= \mathbf{P}_0 + \mathbf{P}_1 \delta_{1,t} + \mathbf{P}_2 \delta_{2,t} + \dots + \mathbf{P}_n \delta_{n,t} \\ V(t) &= \boldsymbol{\eta}(t)^T \mathbf{P}(\delta_t) \boldsymbol{\eta}(t) \quad \mathbf{P}(\delta_{t+1}) = \mathbf{P}(\delta_t) + \mathbf{P}(\Delta \delta_t) \end{aligned}$$

Such that
$$V(t+1) - V(t) < 0$$

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{e}(t) \end{bmatrix} &= \begin{bmatrix} \mathbf{A}(\delta_t) & \mathbf{B}(\delta_t) \\ \mathbf{C}(\delta_t) & \mathbf{D}(\delta_t) \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{v}(t) \end{bmatrix} \end{aligned}$$

$$\mathbf{A}(\delta_t)^T \mathbf{P}(\delta_{t+1}) \mathbf{A}(\delta_t) - \mathbf{P}(\delta_t) < \mathbf{0} \quad [1]$$

APS ↓

LMI: Theorem 4.3

Satisfy for **finite** vertices of **WxS**



21

RP (parameter-dependent)

➤ Affine quadratic Lyapunov performance (AQLP)

Find
$$\begin{aligned} \mathbf{P}(\delta_t) > \mathbf{0} \quad \mathbf{P}(\delta_t) &= \mathbf{P}_0 + \mathbf{P}_1 \delta_{1,t} + \mathbf{P}_2 \delta_{2,t} + \dots + \mathbf{P}_n \delta_{n,t} \\ V(t) &= \boldsymbol{\eta}(t)^T \mathbf{P}(\delta_t) \boldsymbol{\eta}(t) \quad \mathbf{P}(\delta_{t+1}) = \mathbf{P}(\delta_t) + \mathbf{P}(\Delta \delta_t) \end{aligned}$$

Such that

$$\begin{aligned} \begin{bmatrix} \boldsymbol{\eta}(t+1) \\ \mathbf{e}(t) \end{bmatrix} &= \begin{bmatrix} \mathbf{A}(\delta_t) & \mathbf{B}(\delta_t) \\ \mathbf{C}(\delta_t) & \mathbf{D}(\delta_t) \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}(t) \\ \mathbf{v}(t) \end{bmatrix} \end{aligned}$$

$$\begin{bmatrix} \mathbf{A}(\delta_t)^T \mathbf{P}(\delta_{t+1}) \mathbf{A}(\delta_t) - \mathbf{P}(\delta_{t+1}) & \mathbf{A}(\delta_t)^T \mathbf{P}(\delta_{t+1}) \mathbf{B} & \mathbf{C}^T \\ \mathbf{B}^T \mathbf{P}(\delta_{t+1}) \mathbf{A}(\delta_t) & \mathbf{B}^T \mathbf{P}(\delta_{t+1}) \mathbf{B} - \gamma^2 \mathbf{I} & \mathbf{D}^T \\ \mathbf{C} & \mathbf{D} & -\mathbf{I} \end{bmatrix} < \mathbf{0} \quad [1]$$

APS ↓

LMI: Theorem 4.4

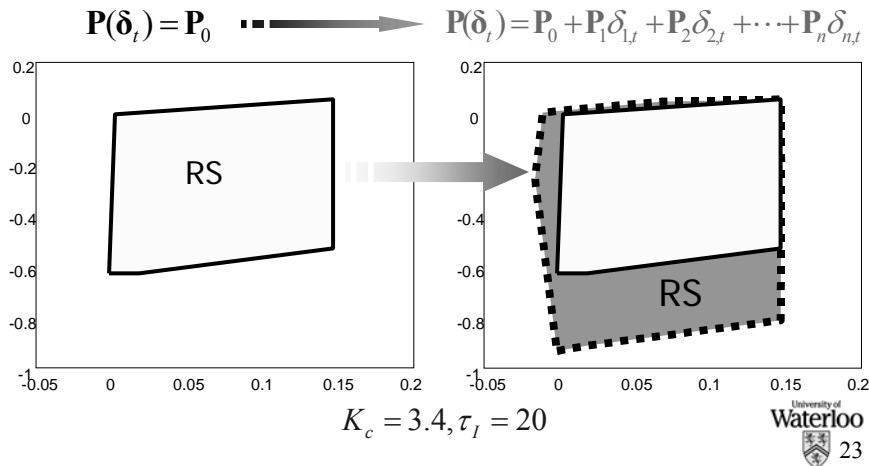
Satisfy for **finite** vertices of **WxS**



22

Results 2 (G-S PI RS)

➤ Figure 5.8:



Input Saturation Factor

➤ Input-saturation (I-S) $\hat{u}(t) = -1, \hat{u}(t) = 1$

➤ Saturation factor: additional uncertain parameter

$$\psi = \frac{1}{|\hat{u}|} \quad \text{I-S: } |\hat{u}| > 1 \quad \rightarrow \quad 0 \leq \psi \leq 1$$

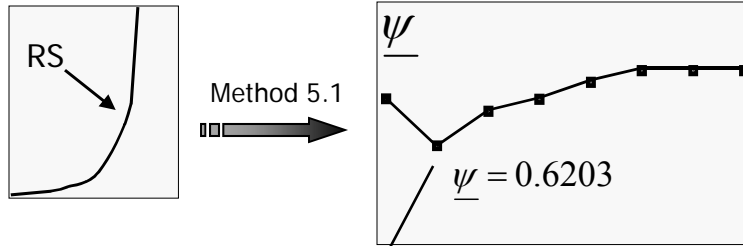
➤ Problem: $\underline{\psi} = 0 \xrightarrow{|\hat{u}| \rightarrow \infty} \text{No RP} (\gamma = \infty)$

➤ Relaxation of input-saturation factor $\underline{\psi}$

$$\begin{matrix} C \in [0 \ 1] \\ e(t) \in [e \ \bar{e}] \end{matrix} \rightarrow \begin{matrix} |\hat{u}| < \infty \\ \underline{\psi} > 0 \end{matrix} \xrightarrow{\text{Method 5.1}} \psi \in [\underline{\psi} \ \bar{\psi}]$$

University of Waterloo
24

Results 2 (Linear PI $\underline{\psi}$)

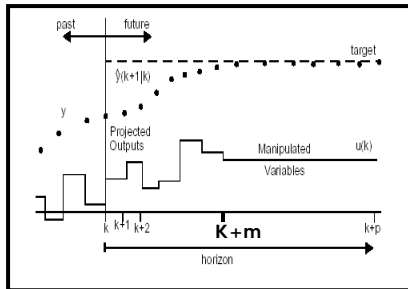


➤ Table 5.3

K_c	τ_I	ψ	$\gamma_{optimal}$
1.3	2.5	$\psi \in [0 \ 1]$	∞
		$\psi \in [0.6203 \ 1]$	0.4702

➤ Conservatism is reduced!

Unconstrained MPC



➤ State-space MPC
(Zanovello and Budman, 1999)

$$\begin{bmatrix} \mathbf{U}(k) \\ \mathbf{u}(k) \end{bmatrix} = \begin{bmatrix} \mathbf{E}_2 & -\mathbf{T}_1 \mathbf{K}_{mpc} \mathbf{N}_2 \\ \mathbf{C}_{u1} & -\mathbf{K}_{mpc} \mathbf{N}_2 \end{bmatrix} \begin{bmatrix} \mathbf{U}(k-1) \\ \mathbf{y}(k) \end{bmatrix}$$

$$\Delta \mathbf{u}(k) = \mathbf{K}_{MPC} \boldsymbol{\varepsilon}(k+1/k)$$

$$\mathbf{K}_{MPC} = [1 \ 0 \ \dots \ 0]_{1 \times m} (\mathbf{S}_p^u \mathbf{T}^T \mathbf{T} \mathbf{S}_p^u + \mathbf{\Lambda}^T \mathbf{\Lambda})^{-1} \mathbf{S}_p^u \mathbf{T}^T \mathbf{T}$$

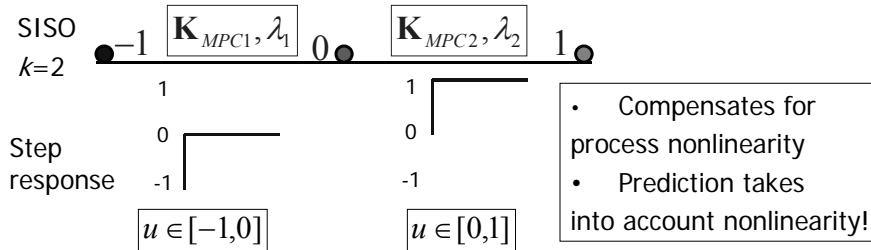
MPC
Performance

Gain-
schedule

Gain-schedule
 $\mathbf{\Lambda} = \lambda \mathbf{I}$

Gain-scheduled MPC (SISO)

- Gain-scheduled MPC: non-traditional, no linearization



- Closed-loop system: APS

$$\begin{bmatrix} \eta(t+1) \\ e(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}(\delta_t) & \mathbf{B}(\delta_t) \\ \mathbf{C}(\delta_t) & \mathbf{D}(\delta_t) \end{bmatrix} \begin{bmatrix} \eta(t) \\ v(t) \end{bmatrix}, \eta(0) = \eta_0$$

Global RS and RP ←

$$\mathbf{A}(\delta_t) = \mathbf{A}_0 + \mathbf{A}_1 \delta_{1,t} + \mathbf{A}_2 \delta_{2,t} \dots + \mathbf{A}_n \delta_{n,t}$$

Results 3 (G-S MPC SISO)

- Optimization Design Results: Table 6.1

GSMPCk	$[\lambda_1, \lambda_2, \dots, \lambda_k]$	$\gamma_{optimal}$
1	[0.7827]	0.5928
2	[0.2732, 0.9499]	0.4926
3 ☆	[0.32797, 0.8219, 1.1513]	0.4907
4	[0.8303, 0.8743, 0.8448, 1.0287]	0.6068
5	[0.9369, 1.0456, 1.0461, 1.0038, 1.0821]	0.6152

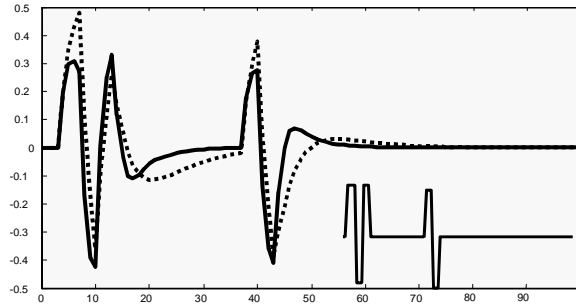
➤ Conclusions

- $\gamma_{optimal}$ close to each other
 - Best RP: k=3
 - RP depends on the discretization number k !
-

Results 3 (G-S MPC SISO)

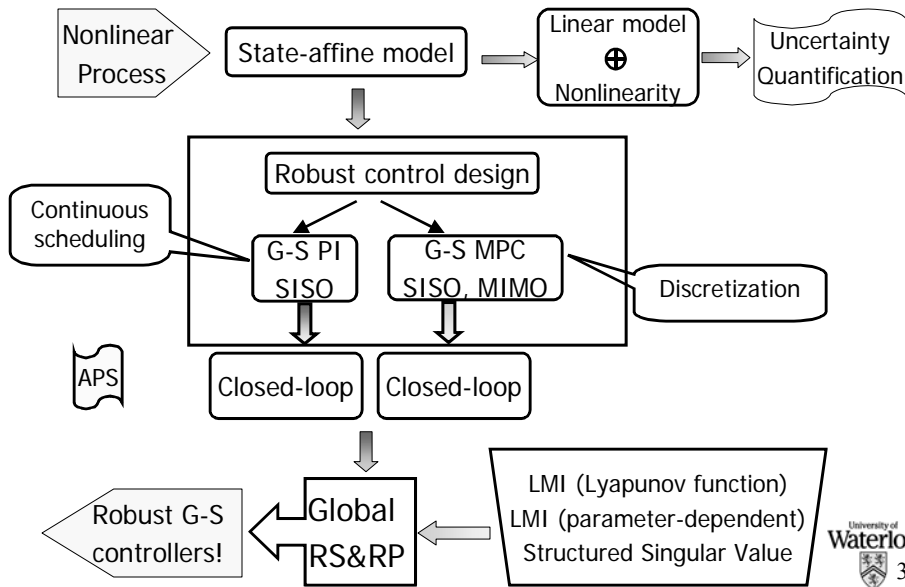
➤ Table 6.2

	$[\lambda_1, \lambda_2, \dots, \lambda_k]$	$\gamma_{optimal}$	$\gamma_{simulation}$
GSMPC5-1	[0.9369, 1.0456, 1.0464, 1.0038, 1.0821]	0.6152	0.3108
GSMPC5-2	[5, 5, 5, 5, 5]	0.7230	0.3295



29

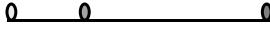
Conclusions



30

Future Directions: conservatism

➤ Gain-scheduled MPC design

- # of discretization and discretization point 
- Apply the proposed approaches to reduce conservatism

➤ Robust Performance Condition

- Integrate $\sum_{t=1}^{100} \mathbf{u}^T(t) \mathbf{u}(t)$ into the performance index
- Formulate RP along *a priori* known disturbance trajectories
- Eliminate higher-order terms from conditions based on parameter-dependent Lyapunov functions

$$\delta_{i,t}^3 = u(t)^i u(t)^i u(t)^i = u(t)^{i \times 3} = \delta_{i \times 3, t}$$

- Construct less conservative parameter box by vertex selection



Acknowledgements

➤ Professor Hector M. Budman

➤ Committee members:

- Professors Thomas Duever, Daniel Davison, James McLellan, Peter Douglas, and Ali Elkamel

➤ Professors in the Department:

- Jenő M. Scharer, Alexander Penlidis, and Park Reilly

➤ My friends and officemates!

Thank You!

