Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2018

Project Title:	**Fault Tolerant Control Systems**
Student:	**Vanita K**
CID:	**00933621**
Course:	**EEE4**
Project Supervisor:	**Dr Imad Jaimoukha**
Second Marker:	**Dr David Angeli**

Acknowledgement

Firstly, I would like to extend my heartfelt gratitude to my supervisor, Dr. Imad Jaimoukha for his invaluable guidance and continuous support throughout the project. I have thoroughly enjoyed the opportunity to work on a project that is mathematically intensive and in one of my favourite fields in engineering and my work would not have been possible without my supervisor who was always forthcoming with his help and support. I would like to thank the previous students who worked on this project as well for their contribution to my work.

I would also like to thank all the staff and students, especially my friends in Imperial for being part of my university journey and making it an enjoyable one.

Finally, I would like mention my endless gratitude to my mother for providing me with all the opportunities in life to achieve a world-class education and my sister for being my constant pillar of support. Thank you both for always being my inspiration.

List of Abbreviations

ARE	Algebraic Riccati Equation
AWGN	Additive White Gaussian Noise
BHP	Block Hadamard Product
BRL	Bounded Real Lemma
BSθ	Binary Search for Theta
DCP	Disciplined Convex Programming
FDI	Fault Detection and Identification
FFTC	Fast Fault Tolerant Control
FTC	Fault Tolerant Control
FTCS	Fault Tolerant Control Systems
FTO	Fault Tolerant Observer
HFFTC	Hybrid Fast Fault Tolerant Control
HFTC	Hybrid Fault Tolerant Control
LHP	Left Half-Plane
LMI	Linear Matrix Inequality
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
LS	Lyapunov Stability
NFTC	Non Fault Tolerant Control
RPP	Regional Pole Placement
SDR	Semidefinite Relaxation

Abstract

The presence of sensor faults in control system could cause adverse effects such as performance degradation or system instability. This project focuses on designing a fault tolerant observer (FTO) using $H\infty$ and regional pole placement RPP control methods where the problems are all posed as linear matrix inequalities (LMIs). Further simplification of the problem is carried out using semidefinite relaxation techniques. The pareto-optimality of system performance and fault tolerance is also investigated for both control design methods. An extension into fault scenarios with linked sensors is also explored using Block Hadamard Product theory, where a similarity between the case of linked sensors and that of FTO design with RPP methods is found, thus providing an integrated solution. Finally, a hybrid active-passive fault tolerant control is briefly outlined with intention of encouraging future development into that area.

List of Figures

2.2.1 Active Fault Tolerant Control System [1] 7

3.2.1 Control Block Diagram of State-Space Control System 12
3.2.2 Control Block Diagram of State Feedback System 13
3.2.3 Control Block Diagram of State Observer System 14

4.3.1 Diagram of region defined by r,α and θ [2] 20

5.1.1 State Observer with Faults Simulation 24

10.1.1 State Observer for Hybrid Fault Tolerant Design 47

11.1.1 Figure showing eigenvalues of the NFTC, FTC and FFTC systems using H_∞ with Lyapunov stability design(from left to right) 52
11.1.2 Figure showing eigenvalues of NFTC, FTC and FFTC systems using H_∞ with RPP design (from left to right) 53
11.1.3 Comparison of γ for NFTC, FTC and FFTC design methods 54
11.1.4 Comparison of γ for Iterative and SDR methods 55
11.1.5 Comparison of average time elapsed for the execution of the 6 different design methods 55
11.1.6 Comparison of average values of θ for NFTC, FTC and FFTC design methods 56
11.1.7 Comparison of average values of γ for NFTC, FTC and FFTC design methods for stable, marginally stable and unstable systems 58
11.2.1 Comparison of γ for H_∞ with LS, FTC and FFTC and H_∞ with RPP, FTC and FFTC design methods against p 60
11.2.2 (a) Comparison of θ for H_∞ with RPP, FTC and FFTC design methods against p, and (b) Comparison of θ loss due to SDR against p 61
11.2.3 Comparison of % modes of instability for each number of faults for H_∞ with LS, FTC and FFTC and H_∞ with RPP, FTC and FFTC design methods against p 62
11.2.4 Comparison of % modes of instability for each number of faults for BSθ, RPP, FTC and FFTC design methods against p 63

11.3.1 Comparison of using unlinked sensors design method with linked sensors design method for FTC design against p: (a) for H_∞ with LS desig, and (b) for H_∞ with RPP design 65
11.3.2 Comparison of using unlinked sensors design method with linked sensors design method for FFTC design against p: (a) for H_∞ with LS design, and (b) for H_∞ with RPP design 66
11.3.3 Comparison of % modes of instability for each number of faults in linked and unlinked sensor FTC designs against p 67
11.3.4 Comparison of % modes of instability for each number of faults in linked and unlinked sensor FFTC designs against p 68
11.4.1 Comparison of Hybrid FTC and FFTC design methods: (a) in terms of performance, and (b) in terms of stability 69

List of Tables

11.1.1 Number of systems that produced a feasible solution for $H\infty$ with Lyapunov stability and $H\infty$ with RPP design methods 53
11.1.2 Number of systems that produced a feasible solution for $H\infty$ with RPP, Binary Search for θ design method 56
11.1.3 Comparison of γ_{diff_R} for NFTC, FTC and FFTC design using BSθ 57
11.1.4 Number of systems that produced a feasible solution for stable, marginally stable and unstable systems 57
11.1.5 % instability for each design method for stable, marginally stable and unstable systems 58
11.2.1 Number of systems that produced a feasible solution for each p for all design methods 59
11.3.1 Number of systems that produced a feasible solution for each p for linked and unlinked FTC and FFTC 64

List of Algorithms

6.1.1 H_∞ with Lyapunov Stability, Iterative Method 28
6.2.1 H_∞ with Lyapunov Stability, SDR Method 28
7.1.1 H_∞ with RPP, Iterative Method 30
7.2.1 H_∞ with RPP, SDR Method 33
8.2.1 H_∞ with Lyapunov Stability, Iterative Method, p-constrained 36
8.2.2 H_∞ with Lyapunov Stability, SDR Method, p-constrained 38
8.3.1 H_∞ with RPP, SDR Method, p-constrained 40
9.2.1 H_∞ with RPP for Linked Sensors, SDR Method, p-constrained 44
9.2.2 H_∞ with RPP, SDR Method, p-constrained using BHP 45
10.2.1 H_∞ with Lyapunov Stability, Iterative Method, Hybrid FTC 49
10.2.2 H_∞ with Lyapunov Stability, SDR Method, Hybrid FTC 50

Contents

Acknowledgement **i**

List of Abbreviations **ii**

Abstract **iii**

List of Figures **iv**

List of Tables **vi**

List of Algorithms **vii**

1 Introduction **4**
1.1 Aims & Objectives 5
1.2 Report Outline 5

2 Literature Review **6**
2.1 Overview of FTCS 6
2.2 Active vs Passive FTCS 7
2.3 LMI in Robust Control 8
2.4 Review Summary 8

3 Theoretical Background **9**
3.1 Matrix Definitions and Identities 9
3.2 Control System Theory 12
3.3 Optimal Control Theory 15
3.4 Semidefinite Programming 16
3.5 Software Packages 17

4 Nominal Control Design **18**
4.1 Pole Placement 18
4.2 H_∞ with Bounded Real Lemma 18

4.3 Regional Pole Placement . . . 19
4.4 Linear Quadratic Gaussian . . . 21
4.5 Comparison of Design Methods . . . 22

5 Fault Tolerant Implementation **23**
5.1 Fault Scenario Modeling . . . 23
5.2 Semidefinite Relaxation . . . 25

6 Fault Tolerant Design: H_∞ with Lyapunov Stability **27**
6.1 Iterative Method . . . 27
6.2 Semidefinite Relaxation Method . . . 28

7 Fault Tolerant Design: H_∞ with RPP **29**
7.1 Iterative Method . . . 29
7.2 Semidefinite Relaxation Method . . . 30
7.3 Binary Search for θ with RPP . . . 33

8 Minimum Fault Tolerance Threshold, p **35**
8.1 Introduction . . . 35
8.2 H_∞ with Lyapunov Stability . . . 36
8.3 H_∞ with RPP . . . 38

9 Fault Tolerant Design for Linked Sensors **41**
9.1 Introduction . . . 41
9.2 p-constrained SDR Method . . . 42

10 Hybrid Active-Passive Fault Tolerant Design **46**
10.1 Introduction . . . 46
10.2 H_∞ with Lyapunov Stability . . . 48

11 Simulation and Results **51**
11.1 Fault Tolerant Design Methods . . . 52
11.2 Minimum Fault Tolerance Threshold . . . 59
11.3 Linked Sensor Faults . . . 63
11.4 Hybrid Fault Tolerant Control . . . 68
11.5 Summary of Results . . . 69

12 Evaluation **71**

13 Conclusion & Future Work **73**

Bibliography **74**

A System Generation **77**

B Fault Tolerant Design **79**
B.1 H_∞ with Lyapunov Stability criterion 79
B.2 H_∞ with RPP 81
B.3 Binary Search for θ 85

C p-constrained FTC Design **93**
C.1 H_∞ with Lyapunov Stability 93
C.2 H_∞ with RPP 96
C.3 Binary Search for θ 100

D Linked Sensors FTC Design **107**
D.1 Linked Faults Generation 107
D.2 Fault Tolerant Control 107
D.3 Fast Fault Tolerant Control 116

E Hybrid FTC **133**

Chapter 1

Introduction

A traditional control feedback system is unable to achieve satisfactory performance, or even stability in the presence of faults. Faults can occur in any components of the system at any time and modern control systems need to be able to achieve sufficient performance and stability to meet increased safety and performance standards. In some industries, faults can lead to catastrophic events e.g. nuclear and aircraft industries, where system failure can lead to immense tragedy. Fault tolerant control systems (FTCS) are a cost-efficient way to ensure that these faults do not turn into failures that cause production to stop or impose a safety risk[3]. Instead of introducing hardware redundancy as a fail safe measure, FTCS use analytical redundancies to ensure robust and stable performance under faulty scenarios, while maximising the optimal fault-free performance. Improving the trade-off between fault-free performance and performance under fault conditions is the motivation for extended research into this field.

One of the challenges of FTCS design is the complexity of it as compared to traditional control systems. This complexity also increases for increasing fault tolerance required by the controller. Although the existence of high performance computers allow for finding a solution for higher order of complexities, this approach will be cost and time intensive. A possible way to alleviate this situation is through semidefinite relaxation (SDR) where it is possible to condense a multi-objective problem into a single linear matrix inequality (LMI) [4]. The use of SDR with FTC is not a novel approach [5] but will be useful in simplifying the implementation of this project.

Another challenge of FTCS design is achieving a design that is relevant to real-life systems. The project does not hone in on one specific system, but rather develops general solutions that can be chosen for implementation based on the designer's requirements. The majority of design methods available do not account for characteristics of real-life systems e.g. the presence of linked sensors and the use of partial fault detection, hence formulating a much more complex design method than necessary.

1.1 Aims & Objectives

This project aims to expand on the work done in previous years on fault tolerant observer (FTO) systems, where fault tolerance is implemented into a general observer system via various control design methods. The project focuses on building an active FTCS in the presence of sensor faults, assuming that fault detection and identification has already been achieved. This problem is seen as a multi-objective optimisation problem, constrained by the non-compatibility of the objectives of maximising optimal controller performance and preserving the stability of the system under faults. This is known as pareto optimality, where it is impossible to make one criterion better off without making the other worse off.

The main contribution of the project is the design of more realistic models that many not necessarily be fault tolerant to all degrees of fault. The primary deliverables of the project can be summarised as follows:

1. Develop **nominal control methods** for design of **fault-free scenarios**
2. Design **active FTCS using H_∞ methods and SDR**
3. Incorporate **more realistic fault scenarios** into FTCS design
4. Design **active FTCS** with a **greater degree of practical applicability**
5. **Simulate and test** design methods for the **investigation of performance-stability trade-off**
6. **Outline and compare** the different characteristics of each design method

1.2 Report Outline

The report is structured as follows. In Chapter 2, an overview of the literature in FTCS is provided. Following that, Chapter 3 outlines the theoretical knowledge required to understand the report, along with control system theory that will be referred to in the design of FTCS. Chapter 4 involves the discussion of control design methods for the nominal controller, followed by an explicit discussion of how faults are incorporated into the design in Chapter 5. Chapter 6 and 7 provide a detailed overview of the fault tolerant design methods while Chapters 8, 9 and 10 are used to examine the implementation of a more realistic approach to FTCS design. While Chapter 8 and 9 include a detailed study into its design formulations, Chapter 10 introduces a novel approach with a preliminary analysis. Chapter 11 summarises the testing outcomes, followed by an overall project evaluation in Chapter 12. Finally, the report is concluded in Chapter 13, along with a discussion of possible future work.

Chapter 2

Literature Review

2.1 Overview of FTCS

The primary developments of fault tolerant control (FTC) have occurred in the past four decades, with the introduction of model-based engineering systems. Prior to that, fault tolerance was achieved manually, through human checks, additional hardware and signal analysis[6]. The concept of FTCS allow for hardware redundancies to be replaced with less costly software redundancies. The research into observer based fault tolerant design was spearheaded by Jones [7] in 1973, following which various design methods have been developed based on the observer scheme to improve robustness and performance. Although there exists various other methods of residual generation techniques, the project is implemented for state observers due to the use of state-space control theory for system description. The earlier studies into FTCS design was largely motivated by the aircraft industry, primarily due to two aeroplane accidents in the 1970s that could have been avertible in the presence of a responsive FTC such that safe landing was possible even in the presence of system and actuator faults [6, 8, 1, 9]. In recent times, however, FTC has garnered extensive interest from industries that require higher levels of safety and reliability e.g. nuclear, power as it guarantees stability and satisfactory performance in the presence of faults [3]. The terminology surrounding FTCS is still widely debated, but in general, the aim is to build a robust FTCS controller, where both the performance and stability are adequate [9].

The FTCS design methods are usually classified by type of controller, method used to design controller, type of system and type of uncertainty. The uncertainties within a system can be characterised as either actuator, sensor or system faults. The process of dealing with system faults are considered more complex as they are more intrinsic to the system and are generally multiplicative while sensor and actuator faults are additive [10]. There is a wide range of research into sensor and actuator faults, notably from Dr.

Jaimoukha's research group [11, 12, 5]. The research group is also focused on the use of linear matrix inequalities (LMIs) with H_∞ control to formulate the design problems. An overview of various other design methods used in FTCS is given in [1]. Moreover, the majority of the research conducted is also concentrated on linear systems with an increasing number of new research on non-linear systems. However, there exists no one ideal unanimous solution for FTCS design, hence it still remains a widely researched topic.

2.2 Active vs Passive FTCS

The primary classification of FTCS is either as passive or active, describing the kind of controller used. Passive FTCS are designed based on a prior knowledge of faults while active FTCS involve fault detection and identification (FDI) and controller reconfiguration, such that the system reacts to a given fault scenario. Passive FTCS provide limited fault tolerance and require an accurate estimation of faults, but are much simpler to design and do not require time to reconfigure. In many real-life systems, passive FTCS are employed while the active FTCS reconfigure to prevent instability [8, 1, 13]. The design of passive FTCS using linear quadratic Gaussian (LQG) control is discussed in [13, 14].

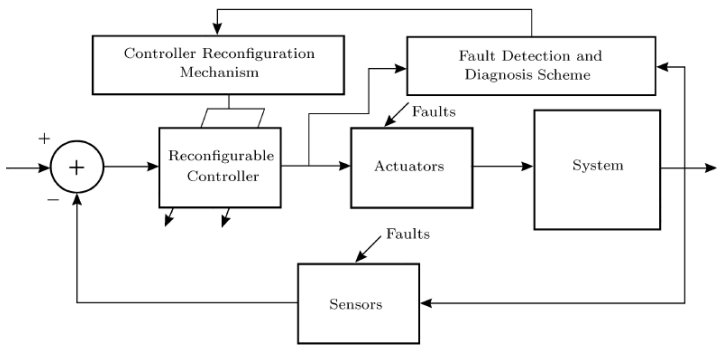

Figure 2.2.1: Active Fault Tolerant Control System [1]

The design of active FTCS involve the design of a fault tolerant controller, the development of a FDI mechanism, the formulation of a reconfiguration mechanism and the integration of all the components [1] as seen in Figure 2.2.1. The project focuses only on the design of an active fault tolerant controller, while assuming the other components exist and interact seamlessly. The active FTCS also face performance degradation in the presence of failures so as to maintain system stability without saturating the redundancies [8]. The measurement of active FTCS performance can be separated into three time periods: time before the fault occurs, time while reconfiguration occurs and time after reconfiguration is complete[1]. Although it is impossible to measure these parameters

purely based on simulations, the use of a switching observer can be essential in identifying model system behaviour before and after a fault occurs.

2.3 LMI in Robust Control

The adoption of LMI for FTCS is popular as seen in [13, 14, 15, 11, 12, 5]. LMIs allow for multi-objective and complex problems to be condensed into more tractable ones that can be solved more efficiently [15]. LMIs are also a key tool in robust control e.g. H_∞, H_2 and Lyapunov based control, hence, the project will be utilise LMI as a design tool for building the FTC.

2.4 Review Summary

The purpose of the literature review is to identify the pre-existing concepts within FTCS, homing in on active controller designs driven by robust control theory and LMIs. Additionally, the review is used to understand the terminology within FTCS and to formulate the criteria that the project should be able to satisfy. The knowledge of the methods used are also vital in drafting novel ideas for the extension of the project.

Chapter 3

Theoretical Background

This chapter outlines the relevant concepts required to understand the technical aspects of the project. The undertaking of the project requires an in-depth knowledge into matrix theory and familiarity with state-space control theory. Firstly, the matrix theory is outlined, followed by the introduction of state-space theory and Lyapunov Stablity criterion. Finally, optimal control theory, which is used to achieve both stability and performance, and the tools required to execute the design methods are discussed.

3.1 Matrix Definitions and Identities

3.1.1 Hermitian Matrix

A complex square matrix $\mathbf{A} \in \mathbb{C}^{\mathbf{m}\times\mathbf{m}}$ is Hermitian [16] if $\mathbf{A} = \mathbf{A}'$, where $\mathbf{A}'$ signifies the complex conjugate transpose of the matrix i.e.

$$\mathbf{A}' = \begin{bmatrix} a_{11} & \cdots & \overline{a_{m1}} \\ \vdots & \ddots & \vdots \\ \overline{a_{1m}} & \cdots & a_{mm} \end{bmatrix} \tag{3.1.1}$$

where $\overline{a_{ij}}$ is the conjugate of a_{ij}.

3.1.2 Positive Definite Matrix

A Hermitian matrix is said to be positive definite [16], denoted as $\mathbf{A} \succ 0$ if

$$x'\mathbf{A}x > 0, \quad \forall x \neq 0 \in \mathbb{C}^{m\times m} \tag{3.1.2}$$

A Hermitian matrix is said to be positive semidefinite [16], denoted as $\mathbf{A} \succeq 0$ if

$$x'\mathbf{A}x \geq 0, \quad \forall x \in \mathbb{C}^{m\times m} \tag{3.1.3}$$

This can be extended to define negative definite and negative semidefinite matrices.

3.1.3 Kronecker Product

For two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$, the Kronecker product [17] of the two matrices is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & \cdots & a_{11}b_{1q} & \cdots & a_{1n}b_{1q} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & \cdots & a_{11}b_{pq} & \cdots & a_{1n}b_{pq} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & \cdots & a_{m1}b_{pq} & \cdots & a_{mn}b_{pq} \end{bmatrix} \tag{3.1.4}$$

3.1.4 Hadamard Product

The Hadamard product between two matrices of the same dimensions $m \times n$ is a $m \times n$ matrix defined as [18]

$$\mathbf{A} \circ \mathbf{B} := (a_{ij}b_{ij}) = \begin{bmatrix} a_{11}b_{11} & \cdots & a_{1n}b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & \cdots & a_{mn}b_{mn} \end{bmatrix} \tag{3.1.5}$$

The multiplication of a Hadamard product by a diagonal matrix, $\mathbf{D}$ provides an useful associative property [19] where

$$\mathbf{D}(\mathbf{A} \circ \mathbf{B}) = \mathbf{A} \circ (\mathbf{D}\mathbf{B}) \tag{3.1.6}$$

3.1.5 Block Hadamard Product

The block Hadamard product (BHP) between two matrices of the same dimensions $m \times n$ is a $m \times n$ matrix defined as [20]

$$\mathbf{A} \,\square\, \mathbf{B} := (\mathbf{A}_{\alpha\beta}\mathbf{B}_{\alpha\beta})_{\alpha=1,\ldots,m}^{\beta=1,\ldots,n} \tag{3.1.7}$$

where each block is a $p \times p$ square matrix and $\mathbf{A}_{\alpha\beta}\mathbf{B}_{\alpha\beta}$ denotes the normal matrix product. For example,

$$\left[\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \\ \hline 3 & 4 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{array}\right] \square \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ \hline 3 & 3 & 3 & 3 \\ 2 & 3 & 4 & 4 \end{array}\right] = \left[\begin{array}{cc|cc} 3 & 3 & 11 & 11 \\ 9 & 9 & 20 & 20 \\ \hline 17 & 21 & 11 & 11 \\ 21 & 15 & 16 & 16 \end{array}\right] \tag{3.1.8}$$

The multiplication of a block Hadamard product by a diagonal matrix, $\mathbf{D}$ also provides an associative property [20] where

$$\mathbf{D}(\mathbf{A} \square \mathbf{B}) = \mathbf{A} \square (\mathbf{DB}) \tag{3.1.9}$$

3.1.6 Schur's Product Theorem

Schur's Theorem states that the Hadamard product of two positive definite matrices, $\mathbf{A} \succ 0$ and $\mathbf{B} \succ 0$ is also positive definite [21] i.e. $\mathbf{A} \circ \mathbf{B} \succ 0$. This theorem can also be extended for the block Hadamard product [20] such that $\mathbf{A} \square \mathbf{B} \succ 0$

3.1.7 Linear Matrix Inequality

A linear matrix inequality (LMI) is formalised as

$$\mathbf{F}(x) \triangleq \mathbf{F}_0 + \sum_{i=1}^{n} x_i \mathbf{F}_i \geq 0 \tag{3.1.10}$$

where $x \in \mathbb{R}^n$ and the symmetric matrix, $\mathbf{F}_i = \mathbf{F}_i'$ are given and $\mathbf{F}(x)$ is a positive definite matrix [22]. LMIs are prevalent in control theory as they can be used to transform complex and multi-objective equations into expressions that can be solved explicitly by convex programming methods. As many of the control problems can be posed as an LMI, the LMI framework becomes ideal to use.

3.1.8 Schur's Complement

The Schur's Complement is an important tool in matrix analysis and can be used to simplify complex matrix calculations. Given $\mathbf{X} = \mathbf{X}' = \begin{bmatrix} X_{11} & X_{12} \\ X_{12}' & X_{22} \end{bmatrix}$, then $\mathbf{X} \prec 0$ implies [16]

$$X_{22} \prec 0, \quad X_{11} - X_{12} X_{22}^{-1} X_{12}' \prec 0 \tag{3.1.11}$$

This comes from the observation that

$$\begin{bmatrix} I & -X_{12}X_{22}^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} X_{11} & X_{12} \\ X_{12}' & X_{22} \end{bmatrix} \begin{bmatrix} I & 0 \\ -X_{22}^{-1}X_{12}' & I \end{bmatrix} = \begin{bmatrix} X_{11} - X_{12}X_{22}^{-1}X_{12}' & 0 \\ 0 & X_{22} \end{bmatrix} \tag{3.1.12}$$

3.2 Control System Theory

3.2.1 State-Space Stability

A continuous-time linear time-invariant state-space model takes the form of

$$\begin{aligned} \dot{x} &= \mathbf{A}x + \mathbf{B}u \\ y &= \mathbf{C}x \end{aligned} \tag{3.2.1}$$

where $x \in \mathbb{R}^n$ represents the state vector, $u \in \mathbb{R}^{n_u}$ represents the input vector and $y \in \mathbb{R}^{n_y}$ represents the output vector. The system can also be represented diagrammatically as seen in Figure 3.2.1.

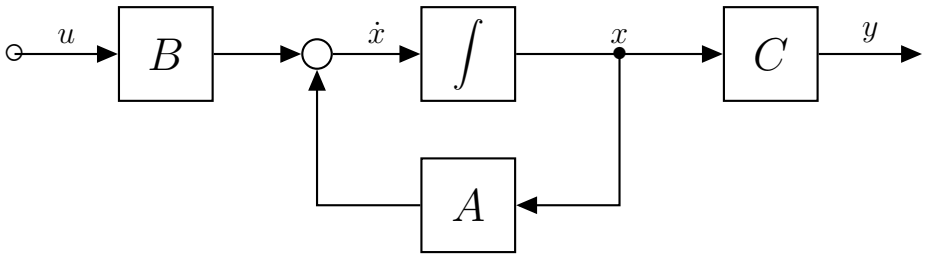

Figure 3.2.1: Control Block Diagram of State-Space Control System

stability of the system is determined by the eigenvalues, denoted by λ, of $\mathbf{A}$ and can be characterised as follows [23]:

- Stable System: $Re\{\lambda_i\} < 0 \quad \forall\, \lambda_i(\mathbf{A})$
- Marginally Stable System: $Re\{\lambda_i\} \leq 0 \quad \forall\, \lambda_i(\mathbf{A})$
- Unstable System: $\exists\, Re\{\lambda_i\} > 0$

3.2.2 State Feedback

The use of a state feedback controller $\mathbf{K}$ modifies the input-to-state interaction, so as to stabilise the closed loop system. This is achieved by introducing a control input that is proportional to the state vector x.

$$u = -\mathbf{K}x \tag{3.2.2}$$

The closed loop system is now defined as

$$\begin{aligned} \dot{x} &= (\mathbf{A} - \mathbf{BK})x + \mathbf{B}r \\ y &= \mathbf{C}x \end{aligned} \tag{3.2.3}$$

where the system stability is dependent on the eigenvalues of $\mathbf{A} - \mathbf{BK}$. The block representation of the system is seen in Figure 3.2.2.

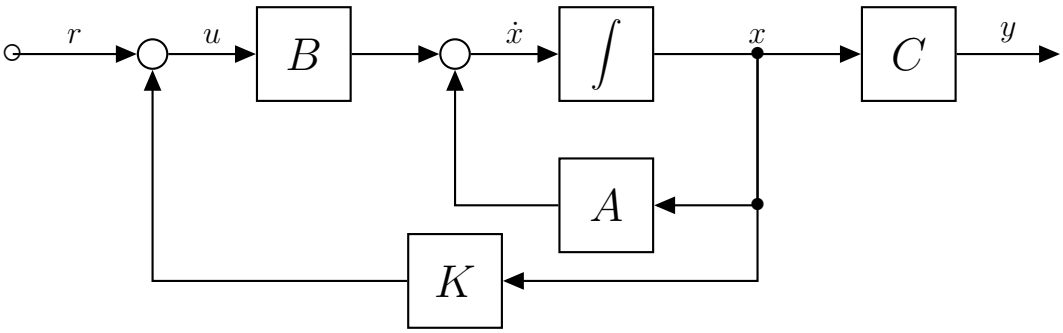

Figure 3.2.2: Control Block Diagram of State Feedback System

3.2.3 State Observer

The state observer is used to estimate the internal state of a system defined as x with only the knowledge of the system inputs, outputs and matrix parameters. The implementation of state feedback control requires knowledge of the system state and since the internal state of the system cannot always be measured, the observer provides a good approximation, such that $\hat{x} \approx x$. The system is shown in Figure 3.2.3 where $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times n_u}$, $\mathbf{C} \in \mathbb{R}^{n_y \times n}$, $\mathbf{C_z} \in \mathbb{R}^{n_z \times n}$, $\mathbf{B_d} \in \mathbb{R}^{n \times n_d}$ and $\mathbf{D_d} \in \mathbb{R}^{n_y \times n_d}$. The system is defined by the following equations:

$$\begin{aligned} \dot{x} &= \mathbf{A}x + \mathbf{B}u + \mathbf{B_d}d \\ y &= \mathbf{C}x + \mathbf{D_d}d \\ \dot{\hat{x}} &= \mathbf{A}\hat{x} + \mathbf{B}u - \mathbf{L}(y - \mathbf{C}\hat{x}) \\ \dot{x} - \dot{\hat{x}} &= \mathbf{A}(x - \hat{x}) + \mathbf{LC}(x - \hat{x}) + (\mathbf{B_d} + \mathbf{LD_d})d \\ \dot{x}_e &\approx (\mathbf{A} + \mathbf{LC})e \end{aligned} \tag{3.2.4}$$

The estimation error of the system, e converges to zero over time **if and only if** the system $\mathbf{A}+\mathbf{LC}$ is asymptotically stable, such that all its eigenvalues have a negative real part [23].

The state feedback controller, $\mathbf{K}$ and the state observer controller, $\mathbf{L}$ can be designed separately from each other due to the separation principle. However, as the state feedback controller is dependent on the accurate approximation of x, the system described in

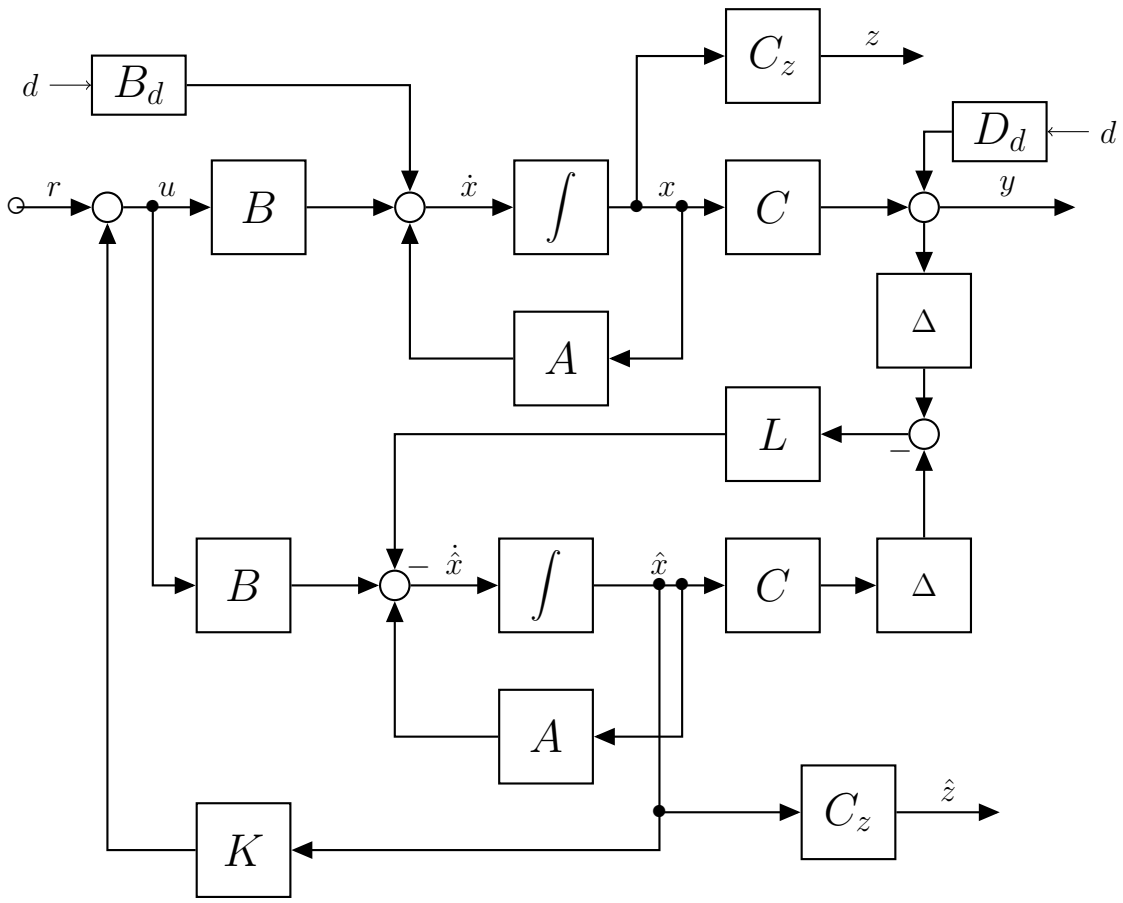

Figure 3.2.3: Control Block Diagram of State Observer System

3.2.4 will have to converge faster than the state feedback system. This requires for the eigenvalues of $\mathbf{A} + \mathbf{LC}$ to be further to the left than the eigenvalues of $\mathbf{A} - \mathbf{BK}$ on the s plane.

It is important to note that the study of the state observer problem is dual to that of the state feedback problem. Since the eigenvalues of a closed loop system are invariant to its transpose, the system characterised by $\dot{x} = (\mathbf{A} + \mathbf{LC})x$ has the same eigenvalues as the system characterised by $\dot{x} = (\mathbf{A}' + \mathbf{C}'\mathbf{L}')x$. This has the same form as the state feedback problem.

$$(\mathbf{A} - \mathbf{BK}) \Leftrightarrow (\mathbf{A}' + \mathbf{C}'\mathbf{L}') \tag{3.2.5}$$

Given the duality of the state feedback and state observer problems, the project only focuses on the design of the state observer as solving one would imply the other is also solved.

3.2.4 Lyapunov Stability Criterion

The Lyapunov stability theory reduces the problem of stability to a LMI which can be used to ensure asymptotic stability of a system, such that all of its poles have negative real parts. Suppose there exists

$$\mathbf{P} = \mathbf{P}' \succ 0 \tag{3.2.6}$$

such that the Lyapunov function is defined as

$$V = x(t)'\mathbf{P}x(t) \tag{3.2.7}$$

If $\dot{V} < 0 \ \forall t$ and $\forall x(t) \neq 0$, the system is stable such that $x \to 0$ as $t \to \infty$ [16]. For a system defined by $\dot{x} = \mathbf{A}x$,

$$\begin{aligned} \dot{V} &= \dot{x}'\mathbf{P}x + x'\mathbf{P}\dot{x} \\ &= x'(\mathbf{A}'\mathbf{P} + \mathbf{PA})x \end{aligned} \tag{3.2.8}$$

Therefore, the system is stable **if and only if**,

$$\mathbf{PA} + \mathbf{A}'\mathbf{P} < 0, \quad \mathbf{P} = \mathbf{P}' \succ 0 \tag{3.2.9}$$

3.3 Optimal Control Theory

3.3.1 $\mathcal{L}_2$ and H_∞ norm

In H_∞ control, the energy of a signal is a measure of the signal size and is represented as the $\mathcal{L}_2$ norm,

$$||x||_2^2 = \int_0^\infty ||x(t)||^2 \, dt$$

Given $u(t) \in \mathcal{L}_2$ and $y(t) \in \mathcal{L}_2$, where $y(s) = \mathbf{G}(s)u(s)$, the H_∞ norm is defined as [16]

$$||\mathbf{G}||_\infty = \max_{w \in \mathbb{R}} \bar{\sigma}[\mathbf{G}(jw)] = \max_{u \neq 0} \frac{||y||_2}{||u||_2}$$

3.3.2 Bounded Real Lemma

The bounded real lemma relates the frequency response characteristics of a system to a LMI, using which the bound on the system response can be computed. Based on the state-space representation of the state observer system defined in equation 3.2.4,

$$\mathbf{G}(s) \stackrel{s}{=} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \tag{3.3.1}$$

where $\mathbf{G}(s)$ is the transfer matrix between the input and output. Given $\gamma > 0$, $||\mathbf{G}||_\infty < \gamma$ and $\mathbf{A}$ is stable **if and only if** there exists $\mathbf{P} = \mathbf{P}' \succ 0$, such that [16]

$$\mathcal{L}(P, \gamma) := \begin{bmatrix} \mathbf{A}'\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{C}'\mathbf{C} & \mathbf{P}\mathbf{B} + \mathbf{C}'\mathbf{D} \\ \mathbf{B}'\mathbf{P} + \mathbf{D}'\mathbf{C} & \mathbf{D}'\mathbf{D} - \gamma^2\mathbf{I} \end{bmatrix} \prec 0 \tag{3.3.2}$$

Using Schur Complement defined in equation 3.1.11,

$$\mathcal{L}(P, \gamma) = \begin{bmatrix} \mathbf{A}'\mathbf{P} + \mathbf{P}\mathbf{A} & \mathbf{P}\mathbf{B} \\ \mathbf{B}'\mathbf{P} & -\gamma^2\mathbf{I} \end{bmatrix} - \begin{bmatrix} \mathbf{C}' \\ \mathbf{D}' \end{bmatrix} (-\mathbf{I}) \begin{bmatrix} \mathbf{C} & \mathbf{D} \end{bmatrix} \prec 0 \tag{3.3.3}$$

implies that

$$\begin{bmatrix} \mathbf{A}'\mathbf{P} + \mathbf{P}\mathbf{A} & \mathbf{P}\mathbf{B} & \mathbf{C}' \\ \mathbf{B}'\mathbf{P} & -\gamma\mathbf{I} & \mathbf{D}' \\ \mathbf{C} & \mathbf{D} & -\gamma\mathbf{I} \end{bmatrix} \prec 0 \tag{3.3.4}$$

3.4 Semidefinite Programming

Semidefinite programming is used as a tool to solve convex optimisation problems where a linear function is minimised subject to a constraint that an affine combination of symmetric matrices is positive semidefinite. In control theory, semidefinite programming is used to efficiently compute solutions for a minimisation problem that is subject to a LMI. This project employs semidefinite programming to solve complex LMIs used to design the controllers.

3.5 Software Packages

3.5.1 MATLAB and Simulink

The project deliverables are achieved via the use of MATLAB and Simulink. The control systems are synthesised in the form of code in MATLAB and as a block model on Simulink. The design of a controller is performed by solving relevant LMIs using the semidefinite programming toolbox, CVX [24]. Performance analysis of the controllers are then achieved by both performance parameters computed in MATLAB and measured output of control block simulations in Simulink. MATLAB and Simulink are selected for this project due to the availability of a control system toolbox within the environment that allows for the analysis, design and tuning of linear control systems.

Chapter 4

Nominal Control Design

This chapter summarises the control design methods used to design the controller for the fault-free case. The main design techniques explored are H_∞ Control and Regional Pole Placement. These design methods involve the formulation of LMIs that are used to achieve both system stability and performance. However, it is important to note that both these parameters are negatively correlated.

4.1 Pole Placement

Pole placement is a closed loop system design method that places poles of the system at predetermined locations. Since the poles of a system characterises its eigenvalues, the location of the poles affect the response of the system. For a state feedback system defined by equation 3.2.3, its poles are roots of the characteristic function [25],

$$|s\mathbf{I} - (\mathbf{A} - \mathbf{BK})| = 0 \tag{4.1.1}$$

In MATLAB, the function *place()* computes the controller gain $\mathbf{K}$ such that the state feedback allows for the closed loop poles to be placed at a location determined by the designer.

4.2 H_∞ with Bounded Real Lemma

H_∞ control is a form of robust control method where the synthesized controller can achieve both stabilisation and performance. The control problem is formulated as a mathematical optimisation problem, where the objective is to find a controller that minimises the cost function. The H_∞ control method utilises the Bounded Real Lemma (BRL) to formulate the problem as a LMI, thus making it easier to solve. By choosing an appropriate cost function, the observer gain can be chosen such that the effects of noise on the output

is minimised. In control theory, this is recognised as the maximisation of disturbance rejection.

From the state observer shown in Figure 3.2.3, the measured system output is defined as $z = \mathbf{C_z}x$ and equivalently, the measured observer output is defined as $\hat{z} = \mathbf{C_z}\hat{x}$. Hence, the transfer matrix between the measured output error, $z - \hat{z}$ and the noise, d is characterised as

$$\begin{bmatrix} \dot{\tilde{x}} \\ z - \hat{z} \end{bmatrix} = \begin{bmatrix} \mathbf{A} + \mathbf{LC} & \mathbf{B_d} + \mathbf{LD_d} \\ \mathbf{C_z} & 0 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ d \end{bmatrix} \tag{4.2.1}$$

where $\tilde{x} = x - \hat{x}$ and the transfer matrix is identified to be

$$\mathbf{T}_{z-\hat{z},d} \stackrel{s}{=} \begin{bmatrix} \mathbf{A} + \mathbf{LC} & \mathbf{B_d} + \mathbf{LD_d} \\ \mathbf{C_z} & 0 \end{bmatrix} \tag{4.2.2}$$

Using the BRL formulation and substituting $\mathbf{Y} = \mathbf{PL}$, to linearise the matrix inequality, the problem can be reformulated as,

$$\mathcal{L}(\mathbf{P}, \gamma) = \begin{bmatrix} \mathbf{PA} + \mathbf{A'P} + \mathbf{C'Y'} + \mathbf{YC} & \mathbf{PB_d} + \mathbf{YD_d} & \mathbf{C_z}' \\ \star & -\gamma\mathbf{I} & 0 \\ \star & \star & \gamma\mathbf{I} \end{bmatrix} \prec 0 \tag{4.2.3}$$

Solving th LMI provides an observer gain, $\mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$ such that

1. $\mathbf{A} + \mathbf{LC}$ is stable
2. $||\mathbf{T}_{z-\hat{z},d}||_\infty < \gamma, \quad \gamma > 0$

if and only if $\mathbf{P} = \mathbf{P'} \succ 0$. The parameter γ is used to measure the disturbance rejection and a smaller γ is indicative of better closed loop system performance.

For the simplicity of use of equation 4.2.3, it will be denoted using the shorthand

$$\mathbf{BRL}(\mathbf{P}, \gamma) < 0 \tag{4.2.4}$$

4.3 Regional Pole Placement

Regional Pole Placement (RPP) involves placing the poles a system in a predefined area dictated by a set of parameters that affect the system performance. Unlike pole placement, regional pole placement provides a greater degree of flexibility and easier implementation for higher order systems.

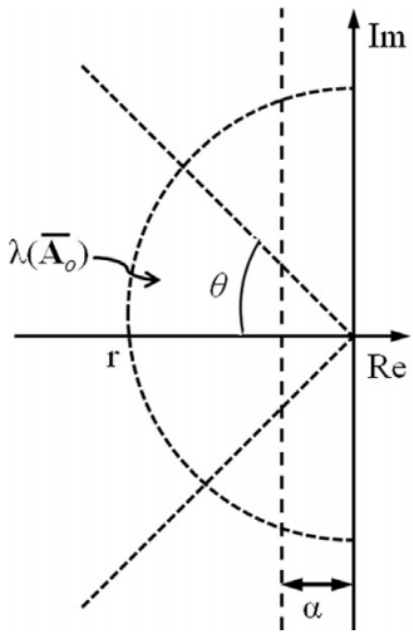

Figure 4.3.1: Diagram of region defined by r,α and θ [2]

The regions are usually defined by r, α and θ and they define the transient response of the system such that

1. $Re\{\lambda\} < -\alpha$
2. $w_d < r \sin\theta$
3. $\zeta < \cos\theta$

where λ determines the decay rate, w_d represents the undamped natural frequency and ζ represents the damping ratio. The region represents an intersection of three different regions, each specified by a LMI as a convex optimisation problem. A LMI region can be any subset $\mathcal{D}$ of the complex plane, where

$$\mathcal{D} = \{z \in \mathbb{C} : \mathbf{L} + z\mathbf{M} + \bar{z}\mathbf{M}' < 0\} \tag{4.3.1}$$

where $\mathbf{L} = \mathbf{L}'$ and $\mathbf{L}$ and $\mathbf{M}$ are real matrices. The function

$$f_{\mathcal{D}}(z) = \mathbf{L} + z\mathbf{M} + \bar{z}\mathbf{M}' \tag{4.3.2}$$

is the characteristic function of $\mathcal{D}$ [26]. A given real matrix $\mathbf{A}$ is only $\mathcal{D}$-stable, such that all its eigenvalues lie in $\mathcal{D}$ **if and only if** there exists a $\mathbf{X} = \mathbf{X}' \succ 0$ such that

$$\mathbf{L} \otimes \mathbf{X} + \mathbf{M} \otimes (\mathbf{XA}) + \mathbf{M}' \otimes (\mathbf{XA})' < 0 \tag{4.3.3}$$

Therefore, the LMIs for the 3 regions are defined as

1. **Half plane**: $\mathbf{XA} + \mathbf{A}'\mathbf{X} + 2\alpha\mathbf{X} < 0$
2. **Disk with radius, r**: $\begin{bmatrix} -r\mathbf{X} & \mathbf{XA} \\ \mathbf{A}'\mathbf{X} & -r\mathbf{X} \end{bmatrix} < 0$
3. **Conic sector with inner angle,** 2θ:

$$\begin{bmatrix} \sin\theta(\mathbf{XA} + \mathbf{A}'\mathbf{X}) & \cos\theta(\mathbf{XA} - \mathbf{A}'\mathbf{X}) \\ -\cos\theta(\mathbf{XA} - \mathbf{A}'\mathbf{X}) & \sin\theta(\mathbf{XA} + \mathbf{A}'\mathbf{X}) \end{bmatrix} < 0$$

4.4 Linear Quadratic Gaussian

The linear quadratic Gaussian (LQG) control problem is the combination of a linear quadratic regulator (LQR) with a Kalman filter, one of the most noteworthy developments from optimal control theory. The LQG control problem is characterised by additive white Gaussian noise (AWGN), linear state-space systems, lack of state information and quadratic costs. The optimal control law of the LQG problem consists of the optimal LQ feedback law applied to the Kalman filter that is used to estimate the state of the system. Hence, he design of the LQR and Kalman filter can be carried out separately. The state-space system can be described as [16]

$$\begin{aligned} \dot{x} &= \mathbf{A}x + \mathbf{B}u + \mathbf{F}w \\ y &= \mathbf{C}x + v \end{aligned} \tag{4.4.1}$$

for which w and v are independent, zero mean, stationary, white Gaussian noises and the cost function is defined as

$$J = \int_0^\infty [x'\mathbf{M}'\mathbf{Q}\mathbf{M}x + u'\mathbf{R}u]dt \tag{4.4.2}$$

for a given $\mathbf{Q} = \mathbf{Q}' \succeq 0$ and $\mathbf{M} = \mathbf{M}' \succ 0$ The minimisation of J is achieved by computing an optimal control law, $u = -\mathbf{K}_c x$, where

$$\mathbf{K}_c = \mathbf{R}^{-1}\mathbf{B}'\mathbf{P}_F \tag{4.4.3}$$

and $\mathbf{P}_F = \mathbf{P}_F' \succ 0$ is the unique solution to the algebraic riccati equation (ARE)

$$\mathbf{A}'\mathbf{P}_F + \mathbf{P}_F\mathbf{A} - \mathbf{P}_F\mathbf{B}\mathbf{R}^{-1}\mathbf{B}\mathbf{P}_F = 0 \tag{4.4.4}$$

The design of the state estimator involves computing an optimal state estimate, x_e such that $E(x - x_e) = 0$ and $E(x - x_e)(x - x_e)'$ is minimised. Hence, the state estimation gain, $\mathbf{L}_O$ is defined as

$$\mathbf{L}_O = \mathbf{P}_O\mathbf{C}'\mathbf{V}^{-1} \tag{4.4.5}$$

where $\mathbf{P}_O = \mathbf{P}_O' \succ 0$ is the unique solution to the ARE

$$\mathbf{A}\mathbf{P}_O + \mathbf{P}_O\mathbf{A}' - \mathbf{P}_O\mathbf{C}'\mathbf{V}^{-1}\mathbf{C}\mathbf{P}_O + \mathbf{F}\mathbf{W}\mathbf{F}' = 0 \tag{4.4.6}$$

4.5 Comparison of Design Methods

The choice of design method is important in achieving the best outcome in terms of fault tolerant design. The methods outlined all have different performance criteria. While pole placement focuses only on stability, H_∞ with BRL aims to minimise disturbance rejection, RPP has the flexibility of multiple performance objectives and LQR focuses on the rejection of AWGN. However, the approach taken for H_∞ methods can also be applied similarly to H_2 methods e.g. LQG, hence, the progress into developing a LQG controller might be not be valuable. Therefore, the project concentrates on H_∞ with BRL and RPP design.

Chapter 5

Fault Tolerant Implementation

This chapter introduces the incorporation of faults into the controller design. Firstly, the fault scenario is developed in the form of a matrix. Then, the concept of semidefinite relaxation is proposed to simplify the design process of a fault tolerant controller.

5.1 Fault Scenario Modeling

The sensor faults are modelled as binary outputs such that they are either completely working or completely not working. The faults are also modelled as having equal probabilities of occurring for simplicity. This can be represented using a diagonal fault matrix, $\Delta \in \mathbb{R}^{n_y \times n_y}$, where n_y defines the number of outputs.

$$\Delta = \begin{bmatrix} \delta_1 & 0 & \dots & 0 \\ 0 & \delta_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \delta_{n_y} \end{bmatrix} \tag{5.1.1}$$

where $\delta_i = 0$ if the i_{th} sensor is not working and $\delta_i = 1$ for a completely working i_{th} sensor. A completely working set of sensors will have $\Delta = \mathbf{I}$ and a set with no working sensors is specified as $\Delta = 0$. The total number of fault scenarios will be all combinations of the fault matrix, and is equal to 2^{n_y}. Thus, as the number of outputs increase, the computational complexity of the problem increases exponentially.

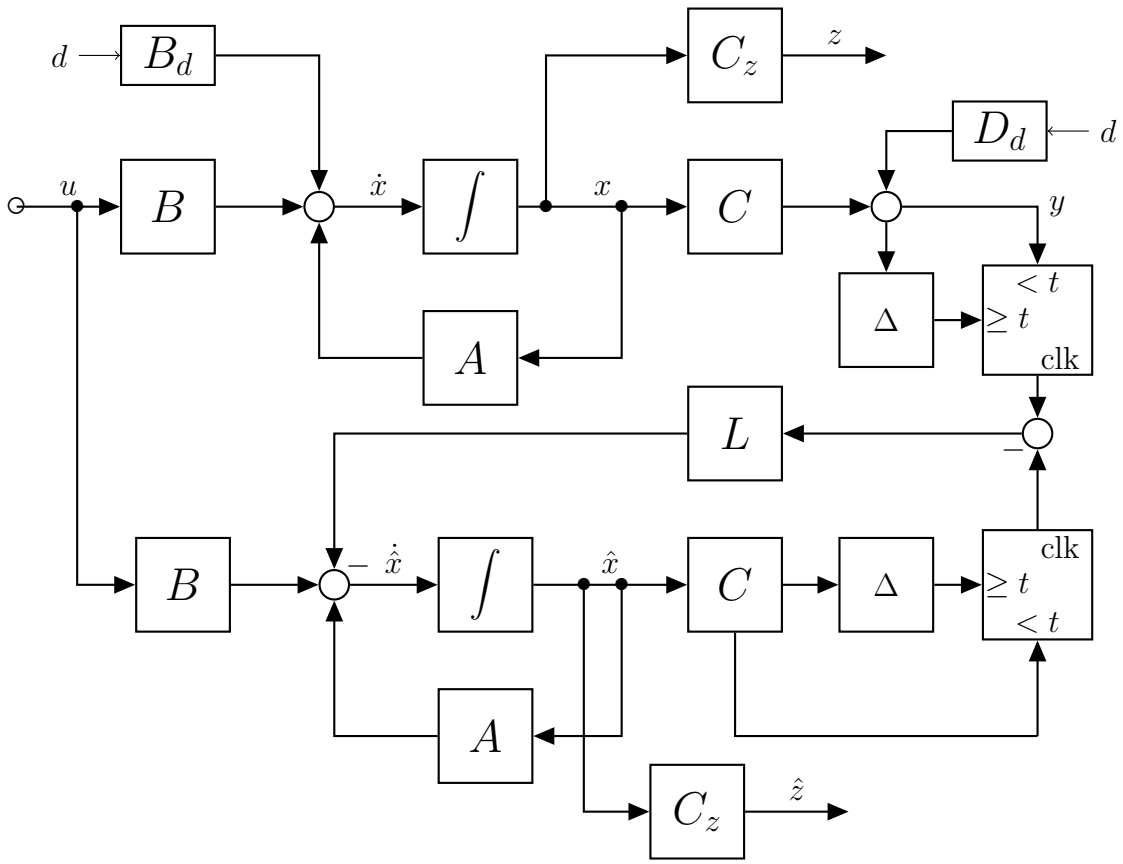

Figure 5.1.1: State Observer with Faults Simulation

The simulation of the faults in the model is carried out using a timer controlled switch, where the first half of the simulation models a fault free system while the second half of the simulation introduces a fault matrix to imitate sensor faults. The schematic of the design can be seen in Figure 5.1.1. The system is characterised by the following equations:

$$\begin{aligned}
\dot{x} &= \mathbf{A}x + \mathbf{B}u + \mathbf{B_d}d \\
y &= \mathbf{C}x + \mathbf{D_d}d \\
\dot{\hat{x}} &= \mathbf{A}\hat{x} + \mathbf{B}u - \mathbf{L}\Delta(y - \mathbf{C}\hat{x}) \\
\dot{x} - \dot{\hat{x}} &= \mathbf{A}(x - \hat{x}) + \mathbf{L}\Delta\mathbf{C}(x - \hat{x}) + (\mathbf{B_d} + \mathbf{L}\Delta\mathbf{D_d})d \\
\dot{e} &\approx (\mathbf{A} + \mathbf{L}\Delta\mathbf{C})e
\end{aligned} \tag{5.1.2}$$

Therefore, the stability of the system is dependent on the eigenvalues of $\mathbf{A} + \mathbf{L}\Delta\mathbf{C}$. The transfer matrix between the measured output error, $z - \hat{z}$ and the noise, d is now defined

as

$$\begin{bmatrix} \dot{\tilde{x}} \\ z - \hat{z} \end{bmatrix} = \begin{bmatrix} \mathbf{A} + \mathbf{L}\Delta\mathbf{C} & \mathbf{B_d} + \mathbf{L}\Delta\mathbf{D_d} \\ \mathbf{C_z} & 0 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ d \end{bmatrix} \tag{5.1.3}$$

where $\tilde{x} = x - \hat{x}$ and the transfer matrix is identified to be

$$\mathbf{T}_{z-\hat{z},d} \overset{s}{=} \begin{bmatrix} \mathbf{A} + \mathbf{LC} & \mathbf{B_d} + \mathbf{LD_d} \\ \mathbf{C_z} & 0 \end{bmatrix} \tag{5.1.4}$$

5.2 Semidefinite Relaxation

The introduction of fault scenarios within the convex optimisation problem adds additional complexity to the problem that increases exponentially with the number of sensors. The use of semidefinite relaxation (SDR) techniques mitigate this difficulty by significantly reducing the number of LMIs that need to be solved. The reformulation of the problem using SDR results in a linear observer design, thus eradicating the need for an iterative solution [5]. The relaxation approach is essential in designing a controller that can be used for real-life scenarios as they require a responsive FTCS design with small time delays. A large time delay could cause the system to become unstable, especially if the underlying system is not stable and is one of the limitations of active FTCS [3].

The general formulation of the problem, taking into account the uncertainties, can be expressed as [5]

$$\mathbf{T_1} + \mathbf{T_2}\Delta\mathbf{T_3} + \mathbf{T_3}'\Delta'\mathbf{T_2}' < 0, \quad \forall\Delta \tag{5.2.1}$$

where $\Delta = diag(\delta_1, ..., \delta_{n_y})$ and $\delta_i \in \{0, 1\}$. Due to the binary property of Δ,

$$\Delta(\mathbf{I} - \Delta) = 0 \tag{5.2.2}$$

Given that there exists a diagonal matrix $\mathbf{S}$,

$$\Delta\mathbf{S} + \mathbf{S}'\Delta' - \Delta(\mathbf{S} + \mathbf{S}')\Delta' = \Delta\mathbf{S}(\mathbf{I} - \Delta') + (\mathbf{I} - \Delta)\mathbf{S}'\Delta' = 0 \tag{5.2.3}$$

The problem in equation 5.2.1 can now be rewritten as

$$(5.2.1) = -\underbrace{\mathbf{T_2}(5.2.3)\mathbf{T_2}'}_{=0} + \begin{bmatrix} \mathbf{I} \\ \Delta'\mathbf{T_2}' \end{bmatrix} \begin{bmatrix} \mathbf{T_1} & \mathbf{T_3}' + \mathbf{T_2}\mathbf{S}' \\ * & -\mathbf{S} - \mathbf{S}' \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{T_2}\Delta \end{bmatrix} \tag{5.2.4}$$

Since the first term of the equation is known to be 0 from equation 5.2.3, equation 5.2.1 is satisfied **if and only if** there exists a diagonal matrix, $\mathbf{S}$ such that

$$\begin{bmatrix} \mathbf{T_1} & \mathbf{T_3}' + \mathbf{T_2}\mathbf{S}' \\ * & -\mathbf{S} - \mathbf{S}' \end{bmatrix} < 0 \tag{5.2.5}$$

The relaxed LMI can now be solved in the CVX environment and provides a computationally efficient approximation approach for solving the original problem [4]. A controller designed using the SDR approach will be referred to as a fast fault tolerant controller (FFTC).

Chapter 6

Fault Tolerant Design: H_∞ with Lyapunov Stability

The first approach to fault tolerance ensures stability for all fault scenarios while maximising disturbance rejection for the fault-free model. The stability for the faulty scenario is implemented using the Lyapunov stability criterion shown in equation 3.2.4 while the fault-free model employs the H_∞ control method to find a controller that minimises γ, the performance indicator. Both LMIs are solved using the convex programming toolbox, CVX.

The design methods are first implemented by iterating through each fault scenario and then executed using the semidefinite relaxation technique to reduce the complexity of the programme.

6.1 Iterative Method

For the system to be stable for every fault scenario, it is necessary to iterate through all 2^{n_y} fault scenarios and design a controller that satisfies the Lyapunov stability criterion for each of the fault scenarios. The introduction of faults modify the observer behaviour from $\mathbf{A} + \mathbf{LC} \rightarrow \mathbf{A} + \mathbf{L}\Delta\mathbf{C}$, hence the Lyapunov stability criterion states that

$$\mathbf{PA} + \mathbf{A}'\mathbf{P} + \mathbf{Y}\Delta_i\mathbf{C} + \mathbf{C}'\Delta_i'\mathbf{Y}' < 0 \tag{6.1.1}$$

$\forall\ i \in \mathbb{Z},\ 0 \leq i \leq 2^{n_y}$ and $\mathbf{Y} = \mathbf{PL}$.

This is executed using Algorithm 6.1.1.

Algorithm 6.1.1 H_∞ with Lyapunov Stability, Iterative Method

CVX begin
Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{Y}$
Constraints: *Minimise* γ
for $i = 1 : 2^{n_y}$ **do**
$\quad \mathbf{PA} + \mathbf{A}'\mathbf{P} + \mathbf{Y}\Delta_i\mathbf{C} + \mathbf{C}'\Delta_i'\mathbf{Y}' < 0$
end for

$\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

CVX end
Outputs: : $\gamma_{min}, \mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$

6.2 Semidefinite Relaxation Method

The iterative loop involving the Lyapunov stability criteria can now be condensed into a single linear matrix inequality using semidefinite relaxation. By comparing equation 6.1.1 with equation 5.2.1, the SDR formulation is achieved by equating

$$\begin{aligned} \mathbf{T}_1 &= \mathbf{PA} + \mathbf{A}'\mathbf{P} \\ \mathbf{T}_2 &= \mathbf{C}' \\ \mathbf{T}_3 &= \mathbf{Y}' \end{aligned} \tag{6.2.1}$$

The fast fault tolerant controller can be designed using Algorithm 6.2.1.

Algorithm 6.2.1 H_∞ with Lyapunov Stability, SDR Method

CVX begin
Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{S} = diag(n_y)$, $\mathbf{Y}$
Constraints: : *Minimise* γ

$$\begin{bmatrix} \mathbf{PA} + \mathbf{A}'\mathbf{P} & \mathbf{Y} + \mathbf{C}'\mathbf{S}' \\ \star & -\mathbf{S} - \mathbf{S}' \end{bmatrix} < 0$$

$\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

CVX end
Outputs: : $\gamma_{min}, \mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$

Chapter 7

Fault Tolerant Design: H_∞ with RPP

The second approach to fault tolerant control involves implementing RPP for all fault scenarios while utilising the H_∞ design method for the fault-free scenario to maximise disturbance rejection. The RPP method allows designers to achieve system stability while imposing restrictions on damping ratio and undamped natural frequency. However, this increases the constraint on the controller, thus reducing the fault-free performance.

Regional Pole Placement is actualised by formulating the region desired using 3 different LMI regions and then solving these inequalities using the CVX toolbox. It is implemented using the iterative method and the semidefinite relaxation method.

In addition to using γ as a measure of performance, it is possible to gauge system performance based on RPP parameters, r, α and θ. One such approach, namely *binary search theta* employs θ as an assessment of system performance.

7.1 Iterative Method

The iterative method is implemented using Algorithm 7.1.1.

Algorithm 7.1.1 H_∞ with RPP, Iterative Method

CVX begin
Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{Y}$, r, α, θ
Constraints: : *Minimise* γ
for $i = 1 : 2^{n_y}$ **do**
$\mathbf{X} := \mathbf{P}$
$\mathbf{A} := \mathbf{A} + \mathbf{L}\Delta_i\mathbf{C}$

$$\mathbf{XA} + \mathbf{A}'\mathbf{X} + 2\alpha\mathbf{X} < 0$$

$$\begin{bmatrix} -r\mathbf{X} & \mathbf{XA} \\ \mathbf{A}'\mathbf{X} & -r\mathbf{X} \end{bmatrix} < 0$$

$$\begin{bmatrix} \sin\theta(\mathbf{XA} + \mathbf{A}'\mathbf{X}) & \cos\theta(\mathbf{XA} - \mathbf{A}'\mathbf{X}) \\ -\cos\theta(\mathbf{XA} - \mathbf{A}'\mathbf{X}) & \sin\theta(\mathbf{XA} + \mathbf{A}'\mathbf{X}) \end{bmatrix} < 0$$

end for

$\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

CVX end
Outputs: : $\gamma_{min}, \mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$

7.2 Semidefinite Relaxation Method

The SDR method is also used with the RPP design method in order to reduce programme complexity and it requires the reformulation of each LMI representing the regions described in Section 4.3.

Half Plane:
The half plane equation is described as

$$\mathbf{PA} + \mathbf{A}'\mathbf{P} + \mathbf{Y}\Delta_i\mathbf{C} + \mathbf{C}'\Delta_i'\mathbf{Y}' + 2\alpha\mathbf{P} < 0 \tag{7.2.1}$$

$\forall\ i \in \mathbb{Z}$, $0 \leq i \leq 2^{n_y}$ and $\mathbf{Y} = \mathbf{PL}$.

Using the SDR formulation in Section 5.2, equation 7.2.1 can be rewritten as

$$\begin{bmatrix} \mathbf{PA} + \mathbf{A}'\mathbf{P} + 2\alpha\mathbf{P} & \mathbf{Y} + \mathbf{C}'\mathbf{S}' \\ * & -\mathbf{S} - \mathbf{S}' \end{bmatrix} < 0 \tag{7.2.2}$$

where

- $\mathbf{T}_1 = \mathbf{PA} + \mathbf{A}'\mathbf{P} + 2\alpha\mathbf{P}$
- $\mathbf{T}_2 = \mathbf{C}'$
- $\mathbf{T}_3 = \mathbf{Y}'$

Disk:
The equation representing the disk is known to be

$$\begin{bmatrix} -r\mathbf{P} & \mathbf{PA} + \mathbf{Y}\Delta_i\mathbf{C} \\ * & -r\mathbf{P} \end{bmatrix} < 0 \tag{7.2.3}$$

$\forall\ i \in \mathbb{Z},\ 0 \leq i \leq 2^{n_y}$ and $\mathbf{Y} = \mathbf{PL}$.

This can be reformulated into the SDR equation using the following formula:

$$\begin{bmatrix} -r\mathbf{P} & \mathbf{PA} + \mathbf{Y}\Delta_i\mathbf{C} \\ * & -r\mathbf{P} \end{bmatrix} = \begin{bmatrix} -r\mathbf{P} & \mathbf{PA} \\ * & -r\mathbf{P} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{C} \end{bmatrix}^T \Delta^T \begin{bmatrix} \mathbf{Y} & 0 \end{bmatrix}^T + \begin{bmatrix} \mathbf{Y} \\ 0 \end{bmatrix} \Delta \begin{bmatrix} 0 & \mathbf{C} \end{bmatrix} \tag{7.2.4}$$

- $\mathbf{T}_1 = \begin{bmatrix} -r\mathbf{P} & \mathbf{PA} \\ * & -r\mathbf{P} \end{bmatrix}$
- $\mathbf{T}_2 = \begin{bmatrix} 0 \\ \mathbf{C} \end{bmatrix}^T$
- $\mathbf{T}_3 = \begin{bmatrix} \mathbf{Y} & 0 \end{bmatrix}^T$

Conic Sector:
The conic sector LMI is outlined in equation 7.2.5

$$\begin{bmatrix} \sin\theta(\mathbf{PA} + \mathbf{A}'\mathbf{P} + \mathbf{Y}\Delta_i\mathbf{C} + \mathbf{C}'\Delta_i'\mathbf{Y}') & \cos\theta(\mathbf{PA} - \mathbf{A}'\mathbf{P} + \mathbf{Y}\Delta_i\mathbf{C} - \mathbf{C}'\Delta_i'\mathbf{Y}') \\ -\cos\theta(\mathbf{PA} - \mathbf{A}'\mathbf{P} + \mathbf{Y}\Delta_i\mathbf{C} - \mathbf{C}'\Delta_i'\mathbf{Y}') & \sin\theta(\mathbf{PA} + \mathbf{A}'\mathbf{P} + \mathbf{Y}\Delta_i\mathbf{C} + \mathbf{C}'\Delta_i'\mathbf{Y}') \end{bmatrix} < 0 \tag{7.2.5}$$

is reformulated using trigonometric addition formulas into

$$\mathbf{T}_1 + \mathbf{T}_2 \begin{bmatrix} \Delta & 0 \\ 0 & \Delta \end{bmatrix} \mathbf{T}_3 + \mathbf{T}_3' \begin{bmatrix} \Delta & 0 \\ 0 & \Delta \end{bmatrix} \mathbf{T}_2' \tag{7.2.6}$$

where

- $\mathbf{T}_1 = \begin{bmatrix} \sin\theta(\mathbf{PA} + \mathbf{A}'\mathbf{P}) & \cos\theta(\mathbf{PA} - \mathbf{A}'\mathbf{P}) \\ -\cos\theta(\mathbf{PA} - \mathbf{A}'\mathbf{P}) & \sin\theta(\mathbf{PA} + \mathbf{A}'\mathbf{P}) \end{bmatrix}$
- $\mathbf{T}_2 = \begin{bmatrix} \sin\frac{\theta}{2}\mathbf{C}' & -\cos\frac{\theta}{2}\mathbf{C}' \\ \cos\frac{\theta}{2}\mathbf{C}' & \sin\frac{\theta}{2}\mathbf{C}' \end{bmatrix}$
- $\mathbf{T}_3 = \begin{bmatrix} \cos\frac{\theta}{2}\mathbf{Y}' & \sin\frac{\theta}{2}\mathbf{Y}' \\ -\sin\frac{\theta}{2}\mathbf{Y}' & \cos\frac{\theta}{2}\mathbf{Y}' \end{bmatrix}$

The conic sector adopts a new uncertainty matrix composed of repeated Δs due to the asymmetry of the matrix in equation 7.2.5. Therefore, this problem requires an extension to the original SDR formulation so that the repeated uncertainty matrix is taken into account.

SDR Extension

Given there exists a matrix $\mathbf{G}$ such that

$$\mathbf{G} = \begin{bmatrix} 0 & \mathbf{D} \\ -\mathbf{D} & 0 \end{bmatrix}$$

where $\mathbf{D} \in \mathbb{R}^{n_y \times n_y}$ is a diagonal matrix and the new uncertainty matrix is defined as

$$\tilde{\Delta} = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta \end{bmatrix} \tag{7.2.7}$$

Then,

$$\tilde{\Delta}\mathbf{G} + \mathbf{G}'\tilde{\Delta}' = 0 \tag{7.2.8}$$

It is also possible to introduce a diagonal matrix, $\mathbf{S}_a \in \mathbb{R}^{2n_y \times 2n_y}$ such that

$$\Delta\mathbf{S}_a + \mathbf{S}_a'\Delta' - \Delta(\mathbf{S}_a + \mathbf{S}_a')\Delta' = 0 \tag{7.2.9}$$

Hence, the SDR formulation is redefined as

$$(7.2.6) = -\underbrace{\mathbf{T}_2(7.2.9)\mathbf{T}_2'}_{=0} - \underbrace{\mathbf{T}_2(7.2.8)\mathbf{T}_2'}_{=0} + \begin{bmatrix} \mathbf{T}_1 & \mathbf{T}_3' + \mathbf{T}_2\mathbf{S}_a' + \mathbf{T}_2\mathbf{G}' \\ * & -\mathbf{S}_a - \mathbf{S}_a' \end{bmatrix} \tag{7.2.10}$$

where $\mathbf{S}_a \in \mathbb{R}^{2n_y \times 2n_y}$ is a diagonal matrix. Therefore,

$$\mathbf{T}_1 + \mathbf{T}_2\tilde{\Delta}\mathbf{T}_3 + \mathbf{T}_3'\tilde{\Delta}'\mathbf{T}_2' < 0 \implies \begin{bmatrix} \mathbf{T}_1 & \mathbf{T}_3' + \mathbf{T}_2 S_a' + \mathbf{T}_2\mathbf{G}' \\ * & -\mathbf{S}_a - \mathbf{S}_a' \end{bmatrix} < 0 \tag{7.2.11}$$

Based on the extension, the SDR method is implemented as follows:

Algorithm 7.2.1 H_∞ with RPP, SDR Method

CVX begin
Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{S} = diag(n_y)$, $\mathbf{S}_a = diag(2n_y)$, $\mathbf{G}$, $\mathbf{Y}$, r, α, θ
Constraints: : *Minimise* γ

$$\begin{bmatrix} \mathbf{PA} + \mathbf{A}'\mathbf{P} + 2\alpha\mathbf{P} & \mathbf{Y} + \mathbf{C}'\mathbf{S}' \\ * & -\mathbf{S} - \mathbf{S}' \end{bmatrix} < 0$$

$$\begin{bmatrix} \mathbf{T}_{1_{Disk}} & \mathbf{T}_{3_{Disk}}' + \mathbf{T}_{2_{Disk}}\mathbf{S}' \\ * & -\mathbf{S} - \mathbf{S}' \end{bmatrix} < 0$$

$$\begin{bmatrix} \mathbf{T}_{1_{Conic}} & \mathbf{T}_{3_{Conic}}' + \mathbf{T}_{2_{Conic}}\mathbf{S}_a' + \mathbf{T}_{2_{Conic}}\mathbf{G}' \\ * & -\mathbf{S}_a - \mathbf{S}_a' \end{bmatrix} < 0$$

$\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

CVX end
Outputs: : $\gamma_{min}, \mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$

7.3 Binary Search for θ with RPP

The following design approach is an extension of that discussed in Section 7.1 and Section 7.2. It involves implementing H_∞ with RPP with varying θ, where the aim is to minimise θ. The purpose of this approach is to identify the minimum value θ can take for a given

r and *alpha*, while achieving stability and disturbance rejection. As discussed in Section 4.3, θ contributes to the damping ratio, ζ of the system. For $0 < \theta < \frac{\pi}{2}$, the system is underdamped, therefore causing it to be oscillatory [27]. As $\theta \to 0$, the system damping increases and approaches critical damping ($\zeta = 1$), where equilibrium state is reached in the minimum amount of time.

It is also possible to choose r or α as the varying parameter and is left to the designer's discretion. However, the optimisation problem cannot be solved using the CVX toolbox as it is no longer a convex optimisation problem and does not obey the disciplined convex programming (DCP) ruleset [28]. Consequently, a binary search is employed on θ to find the solution to the minimisation problem. Given $\theta \in [0, \frac{\pi}{2}]$ and there exists a solution within the range, the binary search proceeds as follows:

1. Initialise $\theta_L = 0$ and $\theta_U = \frac{\pi}{2}$.
2. Run CVX programme for $\theta_x = \frac{\theta_L + \theta_U}{2}$.
3. If solution exists and $\theta_L - \theta_U > \epsilon$, set $\theta_U = \theta_x$; go to Step 2.
4. If solution does not exist and $\theta_L - \theta_U > \epsilon$, set $\theta_L = \theta_x$; go to Step 2.
5. If solution exists and $\theta_L - \theta_U < \epsilon$, search is done; return $\theta_{min} = \theta_x$.
6. If solution does not exist and $\theta_L - \theta_U < \epsilon$, search is done; return $\theta_{min} = \theta_U$.
7. If solution does not exist and $\theta_x = \frac{\pi}{2}$, search is done; no feasible solution found.

The procedure is performed for the CVX programmes shown in Algorithm 7.1.1 (iterative method) and 7.2.1 (SDR method). The effects of minimising θ on the disturbance rejection can also be studied by comparing the values of γ for each θ_x.

Chapter 8

Minimum Fault Tolerance Threshold, p

8.1 Introduction

For a real-world system with high order of complexity, it becomes extremely computationally exhaustive to solve for a controller that is fault-tolerant for all degrees of faults. Moreover, designing a controller that is completely fault tolerant involves sacrificing a large portion of fault-free performance. A realistic implementation of fault tolerance involves adding a minimum fault tolerance threshold,p such that fault tolerance is only achieved for situations where the number of working sensors $\geq p$. The condition can be defined mathematically as

$$trace(\Delta) \geq p \tag{8.1.1}$$

The compromise between the extent of fault tolerance and the system performance for fault-free scenario can be measured by comparing the effects of p on the disturbance rejection parameter, γ or the damping ratio parameter, θ. Based on the relationship between the performance parameters and p, it is essential to design a fault-tolerant controller with a minimum threshold that corresponds to a level of performance that the fault-free system can live with.

The concept of minimum fault tolerance threshold is applied to the fault tolerance design methods discussed in Chapters 6 and 7. The programme is executed for all $p \in [0, n_y]$, where n_y refers to the total number of sensors.

8.2 H_∞ with Lyapunov Stability

The following algorithms extend the minimum fault tolerance threshold model for the H_∞ with Lyapunov stability methods, where the fault tolerance designed by the Lyapunov stability criterion is p-constrained.

8.2.1 Iterative Method

Algorithm 8.2.1 H_∞ with Lyapunov Stability, Iterative Method, p-constrained

```
for p = 0 : n_y do
    CVX begin
    Inputs: P = P' ≻ 0, Y
    Constraints: : Minimise γ
    for i = 1 : 2^{n_y} do
        if trace(Δ_i) ≥ p then
            PA + A'P + YΔ_iC + C'Δ_i'Y' < 0
        end if
    end for

    BRL(P, γ) < 0

    CVX end
    Outputs: : γ_min, L = P^{-1}Y
end for
```

8.2.2 Semidefinite Relaxation Method

For the p-constrained iterative solution, there exists $n_y \times 2^{n_y}$ for loops. Therefore, the complexity of the programme approaches infinity much faster than the original iterative programme. A possible solution is to employ the semidefinite relaxation method such that there exists only n_y for loops, greatly reducing complexity of the problem. The formulation of the SDR problem requires additional work to incorporate the additional parameter, p.

The condition posed in equation 8.1.1 can be rewritten as

$$\mathbf{e}'\Delta\mathbf{e} \geq p \tag{8.2.1}$$

where $\mathbf{e}' = [1, 1, \ldots, 1] \in \mathbb{R}^{1\times n_y}$. It is possible to prove that

$$\begin{aligned} \mathbf{e}'(\mathbf{I}-\Delta)'(\mathbf{I}-\Delta)\mathbf{e} &\leq \mathbf{e}'\mathbf{e} - p \\ \mathbf{e}'(\mathbf{I}-\Delta)'(\mathbf{I}-\Delta)\mathbf{e} &\leq n_y - p \\ (n_y - p) - \mathbf{e}'(\mathbf{I}-\Delta)'(\mathbf{I}-\Delta)\mathbf{e} &\geq 0 \end{aligned} \tag{8.2.2}$$

Applying Schur's Complement twice enables the transformation of the inequality into

$$\begin{aligned} \begin{bmatrix} n_y - p & \mathbf{e}'(\mathbf{I}-\Delta)' \\ (\mathbf{I}-\Delta)\mathbf{e} & \mathbf{I} \end{bmatrix} &\geq 0 \\ \mathbf{I} - (\mathbf{I}-\Delta)\mathbf{e}(ny-p)^{-1}\mathbf{e}'(\mathbf{I}-\Delta)' &\geq 0 \\ (ny-p)\mathbf{I} - (\mathbf{I}-\Delta)\mathbf{e}\mathbf{e}'(\mathbf{I}-\Delta)' &\geq 0 \end{aligned} \tag{8.2.3}$$

Pre-multiplying the equation by $(\mathbf{I}-\Delta)$, post-multiplying by $(\mathbf{I}-\Delta)'$ and realising $(\mathbf{I}-\Delta)$ is an idempotent matrix (i.e yields itself when multiplied with itself) gives

$$\begin{aligned} (\mathbf{I}-\Delta)(ny-p)\mathbf{I}(\mathbf{I}-\Delta)' - (\mathbf{I}-\Delta)\mathbf{e}\mathbf{e}'(\mathbf{I}-\Delta)' &\geq 0 \\ (\mathbf{I}-\Delta)[\underbrace{(ny-p)\mathbf{I} - \mathbf{e}\mathbf{e}'}_{\mathbf{R}}](\mathbf{I}-\Delta)' &\geq 0 \end{aligned} \tag{8.2.4}$$

where $\mathbf{R} \in \mathbb{R}^{n_y\times n_y}$.

$$\begin{aligned} (\mathbf{I}-\Delta)\mathbf{R}(\mathbf{I}-\Delta)' &\geq 0 \\ \mathbf{R} - \Delta\mathbf{R} - \mathbf{R}\Delta' + \Delta\mathbf{R}\Delta' &\geq 0 \end{aligned} \tag{8.2.5}$$

Since $\mathbf{R}$ is fixed, the incorporation of an additional matrix will add the degree of freedom required to reformulate the problem. This is achieved by introducing a new matrix, $\mathbf{M} = \mathbf{M}' \succ 0$. From Schur's Product Theorem (Section 3.1.6), $\mathbf{M} \circ \mathbf{F} \geq 0$ for any $\mathbf{F} \geq 0$. Henceforth,

$$\begin{aligned} \mathbf{M} \circ (\mathbf{R} - \Delta\mathbf{R} - \mathbf{R}\Delta' + \Delta\mathbf{R}\Delta') &\geq 0 \\ \mathbf{M}\circ\mathbf{R} - \mathbf{M}\circ\Delta\mathbf{R} - \mathbf{M}\circ\mathbf{R}\Delta' + \mathbf{M}\circ\Delta\mathbf{R}\Delta' &\geq 0 \end{aligned} \tag{8.2.6}$$

The multiplication of a Hadamard product by a diagonal matrix, Δ allows for the equation to be redefined as

$$\mathbf{M}\circ\mathbf{R} - \Delta(\mathbf{M}\circ\mathbf{R}) - (\mathbf{M}\circ\mathbf{R})\Delta' + \Delta(\mathbf{M}\circ\mathbf{R})\Delta' \geq 0 \tag{8.2.7}$$

Finally, the SDR formulation for the p-constrained controller design can be expressed as

$$\begin{aligned} (5.2.1) = &- \underbrace{\mathbf{T_2}(5.2.3)\mathbf{T_2}'}_{=0} - \underbrace{\mathbf{T_2}(8.2.7)\mathbf{T_2}'}_{\geq 0} \\ &+ \begin{bmatrix} \mathbf{I} & \mathbf{T}_2\Delta \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 + \mathbf{T}_2(\mathbf{M}\circ\mathbf{R})\mathbf{T}_2' & \mathbf{T}_3' + \mathbf{T}_2(\mathbf{S}' + \mathbf{M}\circ\mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M}\circ\mathbf{R}) \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \Delta'\mathbf{T}_2' \end{bmatrix} \end{aligned} \tag{8.2.8}$$

where the problem of solving for the controller has been condensed into a single LMI.

$$\begin{bmatrix} \mathbf{T}_1 + \mathbf{T}_2(\mathbf{M} \circ \mathbf{R})\mathbf{T}_2' & \mathbf{T}_3' + \mathbf{T}_2(\mathbf{S}' - \mathbf{M} \circ \mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M} \circ \mathbf{R}) \end{bmatrix} < 0 \tag{8.2.9}$$

The algorithm implemented is shown below.

Algorithm 8.2.2 H_∞ with Lyapunov Stability, SDR Method, p-constrained

for $p = 0 : n_y$ **do**
  CVX begin
  Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{Y}$, $\mathbf{S} = diag(n_y)$, $\mathbf{M} = \mathbf{M}' \succ 0$
  Constraints: : *Minimise* γ

  $\mathbf{R} := (ny - p)\mathbf{I} - \mathbf{e}\mathbf{e}'$

  $$\begin{bmatrix} \mathbf{T}_1 + \mathbf{T}_2(\mathbf{M} \circ \mathbf{R})\mathbf{T}_2' & \mathbf{T}_3' + \mathbf{T}_2(\mathbf{S}' - \mathbf{M} \circ \mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M} \circ \mathbf{R}) \end{bmatrix} < 0$$

  $\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

  CVX end
  Outputs: : γ_{min}, $\mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$
end for

where $\mathbf{T}_1 = \mathbf{PA} + \mathbf{A}'\mathbf{P}$, $\mathbf{T}_2 = \mathbf{C}'$ and $\mathbf{T}_3 = \mathbf{Y}'$.

8.3 H_∞ with RPP

The algorithms presented in this Section model the minimum fault tolerance for the H_∞ with RPP design methods. The model for the iterative method is repeated from Algorithm 8.2.1, where the Lyapunov criterion is replaced with the RPP LMIs outlined in Algorithm 7.1.1. The discussion for this has been omitted due to triviality. However, the SDR design method requires additional formulation to implement p-constrained fault scenarios and will therefore be explored in this Section. Additionally, since the binary search technique is unique to RPP, it will be discussed briefly.

8.3.1 Semidefinite Relaxation Method

The SDR formulation requires an extension for the conic sector due to the repeated Δs within the formula. Hence, equation 8.1.1 is revised to

$$\mathbf{e}'\tilde{\Delta}\mathbf{e} \geq 2p \tag{8.3.1}$$

where $\tilde{\Delta} \in \mathbb{R}^{2n_y \times 2n_y}$ and $\mathbf{e}' = [1, 1, \ldots, 1] \in \mathbb{R}^{1 \times 2n_y}$. Following the steps proposed in Section 8.2.2, it can be deduced that $\mathbf{M}_c \in \mathbb{R}^{2n_y \times 2n_y}$ and $\mathbf{R}_c \in \mathbb{R}^{2n_y \times 2n_y}$, where

$$\mathbf{R}_c = 2(ny - p)\mathbf{I} - \mathbf{e}\mathbf{e}' \tag{8.3.2}$$

Hence, equation 8.2.7 can be rewritten as

$$\mathbf{M}_c \circ \mathbf{R}_c - \Delta(\mathbf{M}_c \circ \mathbf{R}_c) - (\mathbf{M}_c \circ \mathbf{R}_c)\Delta' + \Delta(\mathbf{M}_c \circ \mathbf{R}_c)\Delta' \geq 0 \tag{8.3.3}$$

Taking into account the modifications applied in Section 7.2,

$$\begin{aligned}(7.2.6) = &- \underbrace{\mathbf{T}_2(7.2.9)\mathbf{T}_2'}_{=0} - \underbrace{\mathbf{T_2}(7.2.8)\mathbf{T_2}'}_{=0} - \underbrace{\mathbf{T_2}(8.3.3)\mathbf{T_2}'}_{\geq 0} \\ &+ \begin{bmatrix} \mathbf{I} & \mathbf{T}_2\Delta \end{bmatrix} \begin{bmatrix} \mathbf{T}_1 + \mathbf{T}_2(\mathbf{M}_c \circ \mathbf{R}_c)\mathbf{T}_2' & \mathbf{T}_3' + \mathbf{T}_2(\mathbf{S}_a' + \mathbf{G}' - \mathbf{M}_c \circ \mathbf{R}_c) \\ \star & -\mathbf{S}_a - \mathbf{S}_a' + (\mathbf{M}_c \circ \mathbf{R}_c) \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \Delta'\mathbf{T}_2' \end{bmatrix}\end{aligned} \tag{8.3.4}$$

where the design of the conic sector is compacted into

$$\begin{bmatrix} \mathbf{T}_1 + \mathbf{T}_2(\mathbf{M}_c \circ \mathbf{R}_c)\mathbf{T}_2' & \mathbf{T}_3' + \mathbf{T}_2(\mathbf{S}_a' + \mathbf{G}' - \mathbf{M}_c \circ \mathbf{R}_c) \\ \star & -\mathbf{S}_a - \mathbf{S}_a' + (\mathbf{M}_c \circ \mathbf{R}_c) \end{bmatrix} < 0 \tag{8.3.5}$$

The implementation of the design can be found in Algorithm 8.3.1 and the respective substitutions for $\mathbf{T}_1$, $\mathbf{T}_2$ and $\mathbf{T}_3$ have been computed in Section 7.2.

8.3.2 Binary Search for θ

The implementation of the minimum fault tolerance threshold with binary search for θ is necessary to compare the change of minimum θ achievable with varying p. This is accomplished by using the steps outlined in Section 7.3 along with the CVX programmes for the p-constrained iterative and SDR (Algorithm 8.3.1) design methods.

Algorithm 8.3.1 H_∞ with RPP, SDR Method, p-constrained

for $p = 0 : n_y$ **do**

CVX begin

Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{Y}$, $\mathbf{S} = diag(n_y)$, $\mathbf{S}_a = diag(2n_y)$, $\mathbf{M} = \mathbf{M}' \succ 0$,
$\mathbf{M}_c = \mathbf{M}_c' \succ 0$, $\mathbf{G}$

Constraints: : *Minimise* γ

$\mathbf{R} := (ny - p)\mathbf{I} - \mathbf{e}\mathbf{e}'$

$\mathbf{R}_c := 2(ny - p)\mathbf{I} - \mathbf{e}\mathbf{e}'$

$$\begin{bmatrix} \mathbf{T}_{1_{Plane}} + \mathbf{T}_{2_{Plane}}(\mathbf{M} \circ \mathbf{R})\mathbf{T}'_{2_{Plane}} & \mathbf{T}'_{3_{Plane}} + \mathbf{T}_{2_{Plane}}(\mathbf{S}' - \mathbf{M} \circ \mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M} \circ \mathbf{R}) \end{bmatrix} < 0$$

$$\begin{bmatrix} \mathbf{T}_{1_{Disk}} + \mathbf{T}_{2_{Disk}}(\mathbf{M} \circ \mathbf{R})\mathbf{T}'_{2_{Disk}} & \mathbf{T}'_{3_{Disk}} + \mathbf{T}_{2_{Disk}}(\mathbf{S}' - \mathbf{M} \circ \mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M} \circ \mathbf{R}) \end{bmatrix} < 0$$

$$\begin{bmatrix} \mathbf{T}_{1_C} + \mathbf{T}_{2_C}(\mathbf{M}_c \circ \mathbf{R}_c)\mathbf{T}'_{2_C} & \mathbf{T}'_{3_C} + \mathbf{T}_{2_C}(\mathbf{S}'_a + \mathbf{G}' - \mathbf{M}_c \circ \mathbf{R}_c) \\ \star & -\mathbf{S}_a - \mathbf{S}'_a + (\mathbf{M}_c \circ \mathbf{R}_c) \end{bmatrix} < 0$$

$\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

CVX end

Outputs: : γ_{min}, $\mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$

end for

Chapter 9

Fault Tolerant Design for Linked Sensors

9.1 Introduction

The design methods studied so far assume that the sensors operate independently from each another. However, in reality, where systems consist of multiple interconnected sensors, it is likely that there exists some dependence. This could be due to similarities in manufacturing, proximity and fluctuations in a common power supply. The presence of linked sensors allow for designing a controller that provides a greater degree of performance as it reduces the number of possible fault scenarios. This translates into having to trade-off less stability for a given performance.

The first step to designing the controller involves the determination of a fault matrix that can capture the linked sensors. Following that, it is possible to formulate design methods, focusing on the concept of Block Hadamard product. This prompts further research into the connection between the design for linked sensors and H_∞ with RPP design methods. Although the correlation between the linked sensors could vary from 0% to 100%, faults are regarded as binary in the context of this project. Hence, the linked sensors are assumed to have identical outputs.

9.1.1 Fault Scenario Modelling

Given that the linked sensors are completely dependent on each other such that their outputs are also linked, the faults can be modelled as

$$\widehat{\Delta} = \begin{bmatrix} \widehat{\delta_1}\mathbf{I}_1 & 0 & \dots & 0 \\ 0 & \widehat{\delta_2}\mathbf{I}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \widehat{\delta_L}\mathbf{I}_L \end{bmatrix} \tag{9.1.1}$$

where L refers to the total number of blocks of linked sensors and each block has the same dimension, $l \times l$. For example, a system with 4 sensor outputs $(\delta_1, \delta_2, \delta_3, \delta_4)$ where δ_1 is linked with δ_2 and δ_3 is linked with δ_4 can be represented as

$$\widehat{\Delta} = \begin{bmatrix} \delta_1 & 0 & 0 & 0 \\ 0 & \delta_2 & 0 & 0 \\ 0 & 0 & \delta_3 & 0 \\ 0 & 0 & 0 & \delta_4 \end{bmatrix} \xRightarrow[\delta_3=\delta_4]{\delta_1=\delta_2} \begin{bmatrix} \delta_1 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & 0 \\ 0 & 0 & \delta_3 & 0 \\ 0 & 0 & 0 & \delta_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \widehat{\delta_1}\mathbf{I}_1 & 0 \\ 0 & \widehat{\delta_2}\mathbf{I}_2 \end{bmatrix} \tag{9.1.2}$$

The implementation of fault tolerant design for linked sensors is identical to the implementation of fault tolerant design for independent sensors and is performed by simply changing the fault matrix from Δ to $\widehat{\Delta}$ in the previous algorithms. However, further consideration is required for the p-constrained SDR methods. It is important to remember that Block Hadamard product can only be employed when the blocks are all of same size, which restricts the type of systems that this design can be implemented for.

9.2 p-constrained SDR Method

Due to the block nature of the linked fault matrix, $\widehat{\Delta}$, Block Hadamard product (BHP) can be used instead of Hadamard product for the SDR formulation. This motivates the transformation of equation 8.2.6 into

$$\mathbf{M}_L \,\square\, (\mathbf{R} - \widehat{\Delta}\mathbf{R} - \mathbf{R}\widehat{\Delta}' + \widehat{\Delta}\mathbf{R}\widehat{\Delta}') \tag{9.2.1}$$

where $\mathbf{M}_L \in \mathbb{R}^{n_y \times n_y}$ and has the form

$$\mathbf{M}_L = \begin{bmatrix} M_{11} & \dots & M_{1L} \\ \vdots & \ddots & \vdots \\ M'_{1L} & \dots & M_{LL} \end{bmatrix} \tag{9.2.2}$$

such that its diagonal blocks, $\mathbf{M}_{ii}$ have the same dimension as the diagonal blocks of the fault matrix, $\delta_i \mathbf{I}_i$.

The extension of Schur's Product Theroem can be used to prove that

$$\mathbf{M}_L \,\square\, (\mathbf{R} - \widehat{\Delta}\mathbf{R} - \mathbf{R}\widehat{\Delta}' + \widehat{\Delta}\mathbf{R}\widehat{\Delta}') \geq 0 \tag{9.2.3}$$

since $\mathbf{M}_L = \mathbf{M}_L' \succ 0$. Following that, the multiplication of a Block Hadamard product by a diagonal matrix, $\widehat{\Delta}$ provides useful property such that the inequality can be redefined as

$$\mathbf{M}_L \,\square\, \mathbf{R} - \widehat{\Delta}(\mathbf{M}_L \,\square\, \mathbf{R}) - (\mathbf{M}_L \,\square\, \mathbf{R})\widehat{\Delta}' + \widehat{\Delta}(\mathbf{M}_L \,\square\, \mathbf{R})\widehat{\Delta}' \geq 0 \tag{9.2.4}$$

Thus, the p-constrained SDR formulation for linked sensors can be derived from the original formulation with the following changes:

1. $\mathbf{M} \rightarrow \mathbf{M}_L$
2. $\mathbf{M} \circ \mathbf{R} \rightarrow \mathbf{M}_L \,\square\, \mathbf{R}$

The algorithm required to realise this implementation calls for a Block Hadamard Product which cannot be achieved with MATLAB functions. Since $\mathbf{M}$ is a variable within the CVX environment, it is not possible to create an additional function to perform BHP. Hence, each block of linked sensors of $\mathbf{M}$ and $\mathbf{R}$ have to identified as variables so that normal matrix multiplication can be carried out between the appropriate blocks. The resultant matrix will produce the BHP between $\mathbf{M}$ and $\mathbf{R}$. The implementation of H_∞ with RPP controller design using the p-constrained SDR method for the linked sensors is summarised in Algorithm 9.2.1.

The use of Block Hadamard product is a more effective way of solving for a controller in the case of linked sensors as it eliminates repeated calculations due to the repeated elements within the fault matrix. A similar approach can also be used to deal with the repeated faults in the uncertainty matrix from equation 9.2.5. For the H_∞ with RPP design, the fault matrix for the conic sector is defined as

$$\tilde{\Delta} = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta \end{bmatrix} \tag{9.2.5}$$

Firstly, it is necessary to define a new structure for $\mathbf{M}_c$ and $\tilde{\Delta}$ that resemble the structure of $\mathbf{M}_L$ and $\widehat{\Delta}$ respectively. By recognising that there exists 2 copies of each sensor output within the uncertainty matrix, it is possible to define $\mathbf{M}_c$ as

$$\mathbf{M}_c = \begin{bmatrix} M_{11} & \dots & M_{1n_y} \\ \vdots & \ddots & \vdots \\ M'_{1n_y} & \dots & M_{n_y n_y} \end{bmatrix} \tag{9.2.6}$$

Algorithm 9.2.1 H_∞ with RPP for Linked Sensors, SDR Method, p-constrained

for $p = 0 : n_y$ **do**

CVX begin

Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{Y}$, $\mathbf{S} = diag(n_y)$, $\mathbf{S}_a = diag(2n_y)$, $\mathbf{G}$

$\mathbf{M}_{ii} = \mathbf{M}'_{ii} \succ 0, \mathbf{M}_{ii_c} = \mathbf{M}'_{ii_c} \succ 0, \mathbf{M}_{ij} = \mathbf{M}'_{ji} \succ 0, \mathbf{M}_{ij_c} = \mathbf{M}'_{ji_c} \succ 0$

Constraints: : *Minimise* γ

$\mathbf{R}_{ii} := (ny - p)\mathbf{I}_L - \mathbf{ones}(l)$

$\mathbf{R}_{ij} := -\mathbf{ones}(l)$

$\mathbf{M}_L \,\square\, \mathbf{R} := \mathbf{M}_{ij} \times \mathbf{R}_{ij} \,\forall\, i, j$

$\mathbf{M}_{L_c} \,\square\, \mathbf{R} := \mathbf{M}_{ij_c} \times \mathbf{R}_{ij} \,\forall\, i, j$

$$\begin{bmatrix} \mathbf{T}_{1_{Plane}} + \mathbf{T}_{2_{Plane}}(\mathbf{M}_L \,\square\, \mathbf{R})\mathbf{T}'_{2_{Plane}} & \mathbf{T}'_{3_{Plane}} + \mathbf{T}_{2_{Plane}}(\mathbf{S}' - \mathbf{M}_L \,\square\, \mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M}_L \,\square\, \mathbf{R}) \end{bmatrix} < 0$$

$$\begin{bmatrix} \mathbf{T}_{1_{Disk}} + \mathbf{T}_{2_{Disk}}(\mathbf{M}_L \,\square\, \mathbf{R})\mathbf{T}'_{2_{Disk}} & \mathbf{T}'_{3_{Disk}} + \mathbf{T}_{2_{Disk}}(\mathbf{S}' - \mathbf{M}_L \,\square\, \mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M}_L \,\square\, \mathbf{R}) \end{bmatrix} < 0$$

$$\begin{bmatrix} \mathbf{T}_{1_C} + \mathbf{T}_{2_C}(\mathbf{M}_{L_c} \,\square\, \mathbf{R})\mathbf{T}'_{2_C} & \mathbf{T}'_{3_C} + \mathbf{T}_{2_C}(\mathbf{S}'_a + \mathbf{G}' - \mathbf{M}_{L_c} \,\square\, \mathbf{R}) \\ \star & -\mathbf{S}_a - \mathbf{S}'_a + (\mathbf{M}_{L_c} \,\square\, \mathbf{R}) \end{bmatrix} < 0$$

$\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

CVX end

Outputs: : γ_{min}, $\mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$

end for

where $M_{ij} \in \mathbb{R}^{2\times 2}\ \forall\ i,j$. For $\tilde{\Delta}$ to be restructured, there needs to exist some permutation matrix, $\mathbf{X}$ such that

$$\mathbf{X}\tilde{\Delta}\mathbf{X}' = \widehat{\Delta} \tag{9.2.7}$$

The left multiplication by $\mathbf{X}$ manipulates the rows of $\tilde{\Delta}$ while the right multiplication by $\mathbf{X}'$ performs column manipulations. The SDR formulation from equation 7.2.6 can now be characterised as

$$\begin{aligned}\mathbf{T}_1 + \mathbf{T}_2\tilde{\Delta}\mathbf{T}_3 + \mathbf{T}_3'\tilde{\Delta}\mathbf{T}_2' &= \mathbf{T}_1 + \mathbf{T}_2\mathbf{X}^{-1}\widehat{\Delta}\mathbf{X}'^{-1}\mathbf{T}_3 + \mathbf{T}_3'\mathbf{X}^{-1}\widehat{\Delta}\mathbf{X}'^{-1}\mathbf{T}_2' \\ &= \mathbf{T}_1 + \widehat{\mathbf{T}}_2\widehat{\Delta}\widehat{\mathbf{T}}_3 + \widehat{\mathbf{T}}_3'\widehat{\Delta}\widehat{\mathbf{T}}_2'\end{aligned} \tag{9.2.8}$$

where $\widehat{\mathbf{T}}_2 = \mathbf{T}_2\mathbf{X}^{-1}$ and $\widehat{\mathbf{T}}_3 = \mathbf{X}'^{-1}\mathbf{T}_3$. The H_∞ with RPP fault tolerant design using the p-constrained SDR method can now be materialised with Block Hadamard product as seen in Algorithm 9.2.2.

Algorithm 9.2.2 H_∞ with RPP, SDR Method, p-constrained using BHP

---

for $p = 0 : n_y$ **do**

CVX begin

Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{Y}$, $\mathbf{S} = diag(n_y)$, $\mathbf{S}_a = diag(2n_y)$, $\mathbf{M} = \mathbf{M}' \succ 0$,
$\mathbf{M}_{ii_c} = \mathbf{M}'_{ii_c} \succ 0, \mathbf{M}_{ij_c} = \mathbf{M}'_{ji_c} \succ 0$, $\mathbf{G}$

Constraints: : *Minimise* γ

$\mathbf{R} := (ny - p)\mathbf{I} - \mathbf{ee}'$

$\mathbf{R}_{ii_c} := 2(ny - p)\mathbf{I}_{2L} - \mathbf{ones}(2l)$

$\mathbf{R}_{ij_c} := -\mathbf{ones}(2l)$

$\mathbf{M}_c \,\square\, \mathbf{R}_c := \mathbf{M}_{ij_c} \times \mathbf{R}_{ij_c}\ \forall\, i,j$

$$\begin{bmatrix} \mathbf{T}_{1_{Plane}} + \mathbf{T}_{2_{Plane}}(\mathbf{M}\circ\mathbf{R})\mathbf{T}'_{2_{Plane}} & \mathbf{T}'_{3_{Plane}} + \mathbf{T}_{2_{Plane}}(\mathbf{S}' - \mathbf{M}\circ\mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M}\circ\mathbf{R}) \end{bmatrix} < 0$$

$$\begin{bmatrix} \mathbf{T}_{1_{Disk}} + \mathbf{T}_{2_{Disk}}(\mathbf{M}\circ\mathbf{R})\mathbf{T}'_{2_{Disk}} & \mathbf{T}'_{3_{Disk}} + \mathbf{T}_{2_{Disk}}(\mathbf{S}' - \mathbf{M}\circ\mathbf{R}) \\ \star & -\mathbf{S} - \mathbf{S}' + (\mathbf{M}\circ\mathbf{R}) \end{bmatrix} < 0$$

$$\begin{bmatrix} \widehat{\mathbf{T}_{1_C}} + \widehat{\mathbf{T}_{2_C}}(\mathbf{M}_c\,\square\,\mathbf{R}_c)\widehat{\mathbf{T}_{2_C}}' & \widehat{\mathbf{T}_{3_C}}' + \widehat{\mathbf{T}_{2_C}}(\mathbf{S}'_a + \mathbf{G}' - \mathbf{M}_c\,\square\,\mathbf{R}_c) \\ \star & -\mathbf{S}_a - \mathbf{S}'_a + (\mathbf{M}_c\,\square\,\mathbf{R}_c) \end{bmatrix} < 0$$

$\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

CVX end

Outputs: : γ_{min}, $\mathbf{L} = \mathbf{P}^{-1}\mathbf{Y}$

end for

Chapter 10

Hybrid Active-Passive Fault Tolerant Design

10.1 Introduction

The idea of incorporating an active controller design with a passive design is known as a hybrid active-passive design. The development of a hybrid active-passive controller is beneficial for situations where the fault detection is extremely complex or fault scenarios are extremely unlikely for some outputs. The concept of partial fault detection is yet another way of making the controller design more applicable to real-life situations.

The design of a hybrid controller requires a change in the state-space description of the state observer since the uncertainty matrices are now designed in such a way that $\hat{\Delta} \neq \Delta$ due to the partial fault detection and identification. It is only after a state-space description has been achieved that a design approach can be formulated.

10.1.1 State-Space Description

The hybrid controller design is implemented for the state observer as shown in Figure 10.1.1. The state-space equations are reformulated as

$$\begin{aligned}
\dot{x} &= \mathbf{A}x + \mathbf{B}u + \mathbf{B_d}d \\
y &= \mathbf{C}x + \mathbf{D_d}d \\
\dot{\hat{x}} &= \mathbf{A}\hat{x} + \mathbf{B}u - \mathbf{L}\Delta\mathbf{C}x - \mathbf{L}\Delta\mathbf{D_d}d + \mathbf{L}\hat{\Delta}\mathbf{C}\hat{x} \\
\dot{x} - \dot{\hat{x}} &= (\mathbf{A} + \mathbf{L}\Delta\mathbf{C})x - (\mathbf{A} + \mathbf{L}\hat{\Delta}\mathbf{C})\hat{x} + (\mathbf{B_d} + \mathbf{L}\Delta\mathbf{D_d})d
\end{aligned} \tag{10.1.1}$$

Due to the intractability of the state-space equations, a new state vector has to be defined in such a way that it describes the closed loop state-space model in equation 10.1.2.

$$
\begin{aligned}
\dot{v} &= \mathbf{A}_v v + \mathbf{B}_v f \\
\dot{z} - \dot{\hat{z}} &= \mathbf{C}_v v + \mathbf{D}_v f
\end{aligned}
\tag{10.1.2}
$$

As we require both the actual system state, x and estimated state $\hat{x}$ to be stable, the new state-space vector can be characterised as

$$
v = \begin{bmatrix} x \\ \hat{x} \end{bmatrix}
\tag{10.1.3}
$$

The state-space equations are then reconstructed to be

$$
\begin{aligned}
\dot{v} &= \begin{bmatrix} \mathbf{A} & 0 \\ -\mathbf{L}\Delta\mathbf{C} & \mathbf{A} + \mathbf{L}\hat{\Delta}\mathbf{C} \end{bmatrix} v + \begin{bmatrix} \mathbf{B} & \mathbf{B_d} \\ \mathbf{B} & -\mathbf{L}\Delta\mathbf{D_d} \end{bmatrix} \begin{bmatrix} u \\ d \end{bmatrix} \\
\dot{z} - \dot{\hat{z}} &= \begin{bmatrix} \mathbf{C}_z & -\mathbf{C}_z \end{bmatrix} v
\end{aligned}
\tag{10.1.4}
$$

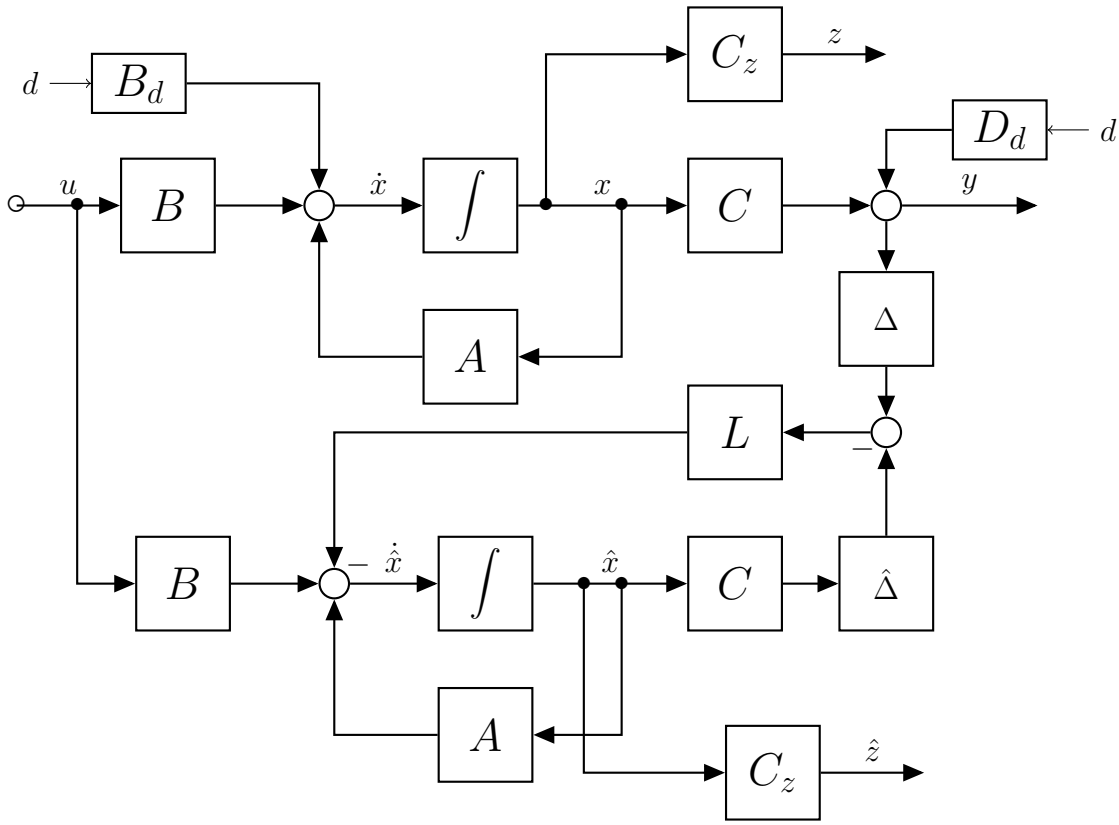

Figure 10.1.1: State Observer for Hybrid Fault Tolerant Design

From the state-space model, it is evident that there exists closed loop stability **if and only if** all eigenvalues of $\mathbf{A}_v$ have negative real parts. Since $\mathbf{A}_v$ is an upper triangular matrix, it is sufficient that the entries of its main diagonal satisfy the closed loop stability criterion. Hence, the closed loop system is stable **if and only if**

1. $\mathbf{A}$ is stable $\implies Re\{\lambda_i(\mathbf{A})\} > 0 \quad \forall i$
2. $\mathbf{A} + \mathbf{L}\hat{\Delta}\mathbf{C}$ is stable $\implies Re\{\lambda_i(\mathbf{A} + \mathbf{L}\hat{\Delta}\mathbf{C})\} > 0 \quad \forall i$

The stability of $\mathbf{A}$ is guaranteed for systems that already possess closed loop stability. However, this restricts the design flexibility as it cannot be implemented for marginally stable or unstable closed loop systems. The stability of $\mathbf{A} + \mathbf{L}\hat{\Delta}\mathbf{C}$ can be ensured by either using Lyapunov Stability criterion or RPP design methods.

Prior to addressing the design method, it is necessary to outline the uncertainty matrix, $\hat{\Delta}$. Due to partial fault detection, it is impossible to know the exact value of Δ. $\hat{\Delta}$ is an estimation of Δ and can be split into actively controlled sensors for which $\Delta = \hat{\Delta}$ and passively controlled sensors for which the uncertainty matrix is estimated as $\mathbf{I}$, the identity matrix.

$$\hat{\Delta} = \begin{bmatrix} \Delta_A & 0 \\ 0 & \mathbf{I} \end{bmatrix} \tag{10.1.5}$$

The partition of the uncertainty matrix allows for the active design controller and passive design controller to be designed separately as seen in equation 10.1.6.

$$\begin{aligned} \mathbf{A} + \mathbf{L}\hat{\Delta}\mathbf{C} :&= \mathbf{A} + \begin{bmatrix} \mathbf{L}_A & \mathbf{L}_P \end{bmatrix} \begin{bmatrix} \Delta_A & 0 \\ 0 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{C}_A \\ \mathbf{C}_P \end{bmatrix} \\ &= \mathbf{A} + \mathbf{L}_A \Delta_A \mathbf{C}_A + \mathbf{L}_P \mathbf{C}_P \end{aligned} \tag{10.1.6}$$

The more the fault scenarios that cannot be identified, the higher the risk of the system becoming unstable as the controller approaches a totally passive and non-fault tolerant design. However, it is also possible contrive the passive controller separately using passive FTCS design e.g. in [13, 14]. The project has solely focused on active FTCS techniques up till now, hence, an enhanced passive FTCS design approach will have to be considered for future work.

10.2 H_∞ with Lyapunov Stability

10.2.1 Iterative Method

Based on the Lyapunov Stability criterion, a controller can be designed to assure stability **if and only if** the following equation is true $\forall\, \Delta$.

$$\mathbf{P}(\mathbf{A}+\mathbf{L}\hat{\Delta}\mathbf{C})+(\mathbf{A}+\mathbf{L}\hat{\boldsymbol{\Delta}}\mathbf{C})'\mathbf{P}<0, \quad \mathbf{P}=\mathbf{P}'\succ 0 \tag{10.2.1}$$

The equation can then be linearised and decomposed into its constituent active and passive parts.

$$\mathbf{PA}+\mathbf{A}'\mathbf{P}+\mathbf{Y}_A\Delta_A\mathbf{C}_A+\mathbf{C}_A'\Delta'\mathbf{Y}_A'+\mathbf{Y}_P\mathbf{C}_P+\mathbf{C}_P'\mathbf{Y}_P'<0, \quad \mathbf{P}=\mathbf{P}'\succ 0 \tag{10.2.2}$$

Similar to previous design approaches, the LMI is solved using the CVX toolbox for all fault scenarios. The implementation also incorporates the BRL as seen previously. The BRL equation is unchanged as it is only applied for the fault-free scenario for which $\hat{\Delta}=\Delta=\mathbf{I}$.

Algorithm 10.2.1 H_∞ with Lyapunov Stability, Iterative Method, Hybrid FTC

CVX begin
Inputs: $\mathbf{P}=\mathbf{P}'\succ 0$, $\mathbf{Y}_A$, $\mathbf{Y}_P$, $\mathbf{C}_A$, $\mathbf{C}_P$
Constraints: Minimise γ
for $i=1:2^{n_y}$ **do**
  $\mathbf{PA}+\mathbf{A}'\mathbf{P}+\mathbf{Y}_A\Delta_A\mathbf{C}_A+\mathbf{C}_A'\Delta'\mathbf{Y}_A'+\mathbf{Y}_P\mathbf{C}_P+\mathbf{C}_P'\mathbf{Y}_P'<0$
end for

$\mathbf{BRL}(\mathbf{P},\gamma)<0$

CVX end
Outputs: : $\gamma_{min}, \mathbf{L}_A=\mathbf{P}^{-1}\mathbf{Y}_A$, $\mathbf{L}_P=\mathbf{P}^{-1}\mathbf{Y}_P$

10.2.2 Semidefinite Relaxation Method

The algorithm for the iterative method can be compacted into a single LMI using the SDR method, increasing the speed and ease of execution of the algorithm. By comparing equation 10.2.2 with equation 5.2.1, the SDR formulation is achieved by equating

$$\begin{aligned}\mathbf{T}_1&=\mathbf{PA}+\mathbf{A}'\mathbf{P}+\mathbf{Y}_P\mathbf{C}_P+\mathbf{C}_P'\mathbf{Y}_P'\\ \mathbf{T}_2&=\mathbf{C}_A'\\ \mathbf{T}_3&=\mathbf{Y}_A'\end{aligned} \tag{10.2.3}$$

The fast fault tolerant controller can be designed using Algorithm 6.2.1.

Algorithm 10.2.2 H_∞ with Lyapunov Stability, SDR Method, Hybrid FTC

CVX begin

Inputs: $\mathbf{P} = \mathbf{P}' \succ 0$, $\mathbf{S} = diag(n_y)$, $\mathbf{Y}_A$, $\mathbf{Y}_P$, $\mathbf{C}_A$, $\mathbf{C}_P$

Constraints: : *Minimise* γ

$$\begin{bmatrix} \mathbf{PA} + \mathbf{A}'\mathbf{P} + \mathbf{Y}_P\mathbf{C}_P + \mathbf{C}_P'\mathbf{Y}_P' & \mathbf{Y}_A + \mathbf{C}_A'\mathbf{S}' \\ \star & -\mathbf{S} - \mathbf{S}' \end{bmatrix} < 0$$

$\mathbf{BRL}(\mathbf{P}, \gamma) < 0$

CVX end

Outputs: : $\gamma_{min}, \mathbf{L}_A = \mathbf{P}^{-1}\mathbf{Y}_A$, $\mathbf{L}_P = \mathbf{P}^{-1}\mathbf{Y}_P$

Chapter 11

Simulation and Results

The project deliverable involves simulating the various design techniques devised for randomly generated systems. The testing is initially carried out on state observer models created in Simulink and this is referred to as independent model testing and aims to verify if the design methods achieve the desired results in a simulation. The eigenvalues of the system are also generated using the MATLAB funtion *eig()* to check for instability.

The second stage of testing, which is also the most significant, pits the design methods against each other to compare differences in stability and performance. This is referred to as loop testing as it is run 100 times such that the design methods are tested against each other for performance and stability for 100 different randomly generated systems. The measurement of performance can be done by comparing γ, θ or even time while the measurement of stability is done by measuring the % of unstable modes. This is equivalent to the number of fault scenarios with any $\lambda > 0$ relative to the total number of fault scenarios.

Another measure of system performance is the number of systems that can be solved to obtain a feasible controller. A design that can be implemented for all randomly generated systems is seen to be extremely flexible while design methods that are unable to provide a feasible solution for any of the design methods are not of much use in the real world even though it might have ideal characteristics.

The system parameters used are $n = 5, n_u = 4, n_y = 6, n_d = 1, n_z = 1$ and the system matrices are generated using the code from Appendix A. Moreover, $log_{10}(\gamma)$ is frequently used for performance analysis and is generally denoted as $log(\gamma)$. The preference for $log(\gamma)$ instead of γ is because of the high standard deviation in output, primarily due to the randomness of system generation. The use of the logarithm allows for a more comprehensive view of the results.

11.1 Fault Tolerant Design Methods

There exist 2 fault tolerant design methods, namely H_∞ with Lyapunov stability and H_∞ with RPP, as discussed previously. For each design method, three different types of controllers are programmed: non fault-tolerant controller (NFTC), FTC and FFTC.

11.1.1 Independent Model Testing

The first phase of testing of design methods investigates the integrity of the methods outlined in Chapter 6 and 7 to its theory. Figure 11.1.1 provides an example of implementation of H_∞ with Lyapunov stability (LS) for NFTC, FTC and FFTC formulations. As detailed in the theory, the use of LS criterion ensures all eigenvalues are in the left half-plane (LHP) for all fault scenarios, hence safeguarding the system from instability. The results of the NFTC design clearly shows the impact of not having FTC, that is eigenvalues with positive real parts and system instability.

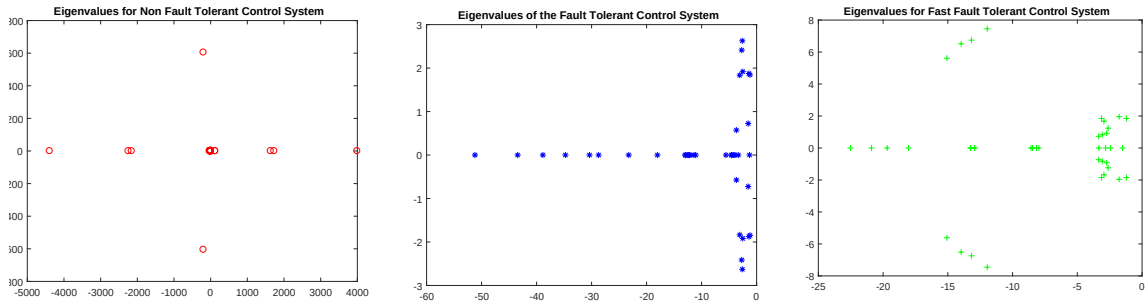

Figure 11.1.1: Figure showing eigenvalues of the NFTC, FTC and FFTC systems using H_∞ with Lyapunov stability design(from left to right)

Figure 11.1.2 also confirms the validity of the RPP algorithm as all the eigenvalues are encased within the required LMI region, except in the case of the NFTC, where instability is reached once again.

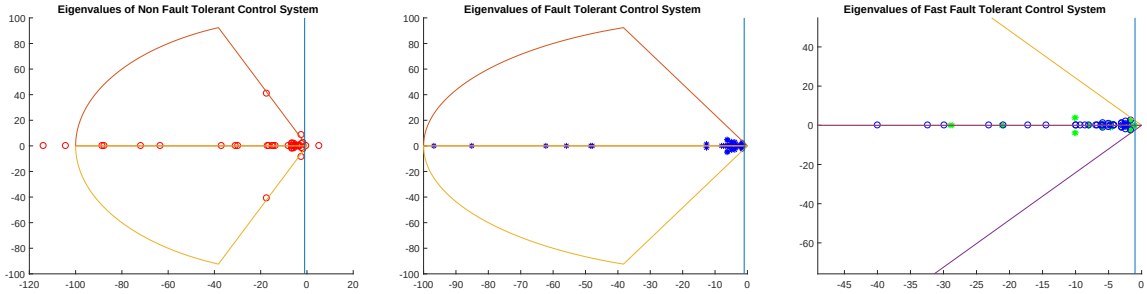

Figure 11.1.2: Figure showing eigenvalues of NFTC, FTC and FFTC systems using H_∞ with RPP design (from left to right)

11.1.2 Comparison of H_∞ Design Methods

The second phase involves comparing the performance, stability and flexibility of the 6 different controllers. The flexibility of the design methods is apparent from Table 11.1.1, where the number of systems with feasible solutions are given for each controller design. The feasibility of the RPP design drops significantly when implementing fault tolerance, while the FTC using LS design can still be used for almost all systems.

	NFTC	FTC	FFTC
H_∞ with LS	100	100	99
H_∞ with RPP	100	48	47

Table 11.1.1: Number of systems that produced a feasible solution for $H\infty$ with Lyapunov stability and $H\infty$ with RPP design methods

Secondly, the performance between the RPP design method and LS method is studied by comparing the values of $log(\gamma)$ for each randomly generated system. This method is beneficial in analysing the exact differences in γ for each trial.

From Figure 11.1.3, it is clear that the use of fault tolerance leads to significant performance degradation for RPP design method. The LS method outperforms RPP for both FTC and FFTC with negligible sacrifice of performance. Finally, with regards to stability, both design methods achieve **100**% stability with the incorporation of fault tolerance. The nominal LS and RPP controllers, on the other hand, have a **32.1**% and **43.1**% chance of reaching instability on average respectively.

Overall, the LS method surpasses the RPP method in flexibility, performance and stability. The redeeming quality of the RPP method is its marginally better performance for NFTC

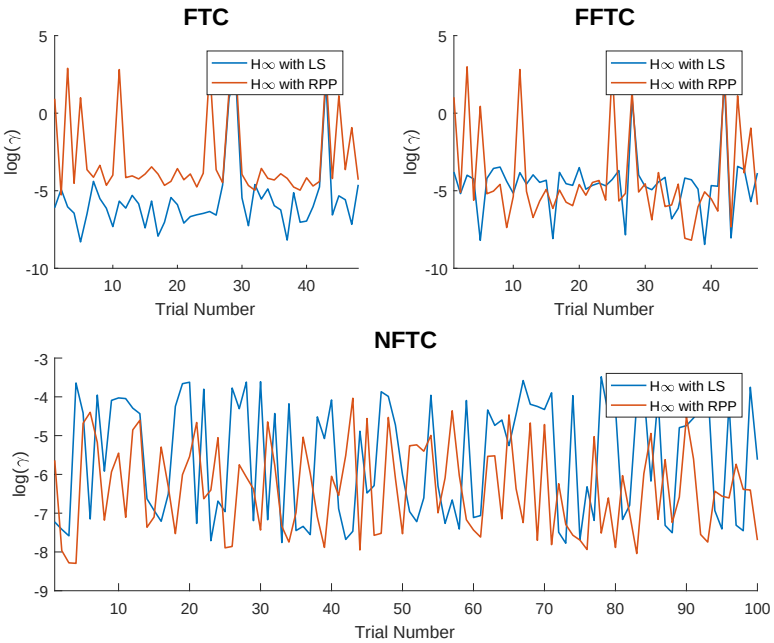

Figure 11.1.3: Comparison of γ for NFTC, FTC and FFTC design methods

design and ability to achieve more specific design requirements as compared to LS. It is important to note that the performance value varies wildly based on the system, however, a majority of the tests display a good level of disturbance rejection i.e $\gamma < 10^{-1}$.

11.1.3 Effectiveness of SDR

The final phase comprises of assessing the feasibility of SDR methods. For the SDR method to be viable, it has to provide similar performance to the iterative method, with significant reduction in complexity and time. The performance comparison is provided in Figure 11.1.4, where the similarity in performance between the iterative and SDR methods can be seen. In fact, SDR provides performance enhancement for RPP methods.

The measure of time elapsed for the execution of each algorithm is done in MATLAB using its internal timers *tic* and *toc*. The SDR has shown to significantly reduce time required in Figure 11.1.5, making the design execution speed comparable to that of NFTC. This is essential for active FTCS where a slow speed of response could push the system to instability.

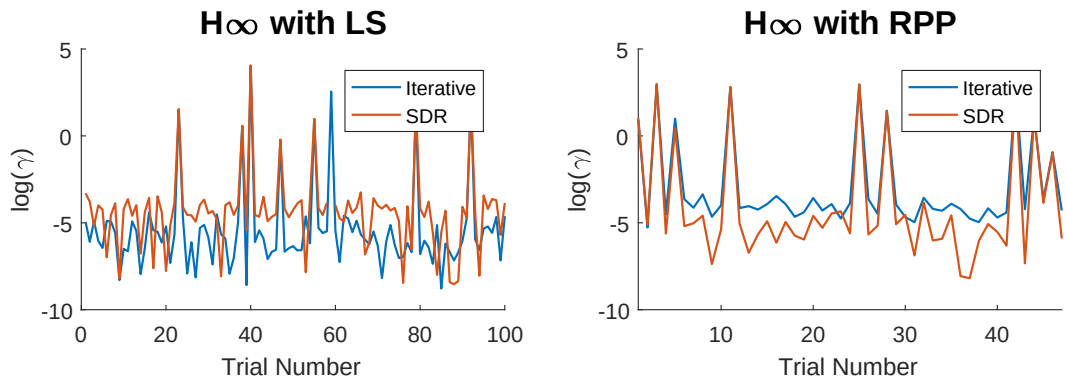

Figure 11.1.4: Comparison of γ for Iterative and SDR methods

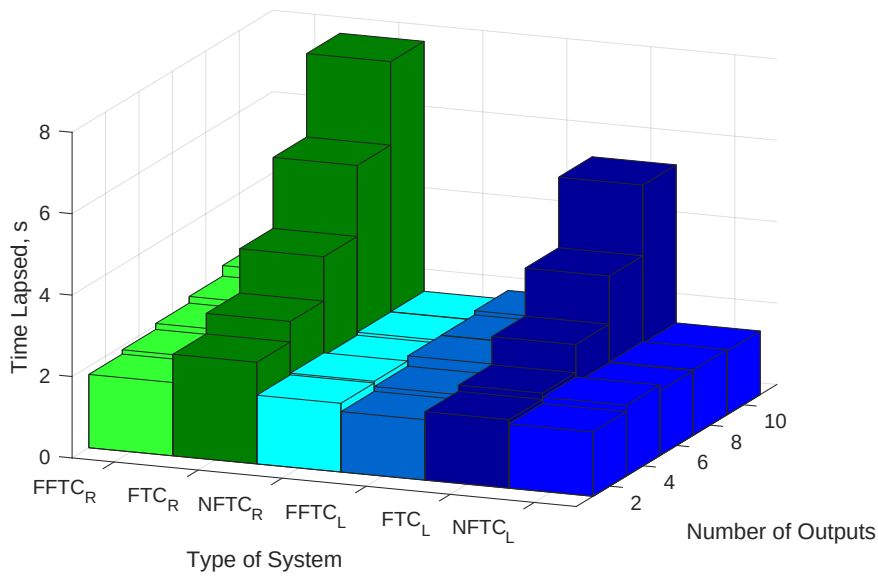

Figure 11.1.5: Comparison of average time elapsed for the execution of the 6 different design methods

11.1.4 Binary Search for θ

An extension of the RPP design method involves the use of θ as a performance indicator. This provides designers with alternate requirements for system design that is not focused

on disturbance rejection. The design of a RPP based FTC using θ is evidently more flexible as seen from Table 11.1.2.

	NFTC	FTC	FFTC
BSθ, RPP	100	100	100

Table 11.1.2: Number of systems that produced a feasible solution for $H\infty$ with RPP, Binary Search for θ design method

The measure of the performance indicator is given in Figure 11.1.6, where the value of maximum possible θ increases significantly with the implementation of fault tolerance. This is a good example of the pareto-optimality of performance parameters and robustness of design. The figure also demonstrates the effectiveness of SDR as it causes no performance sacrifice.

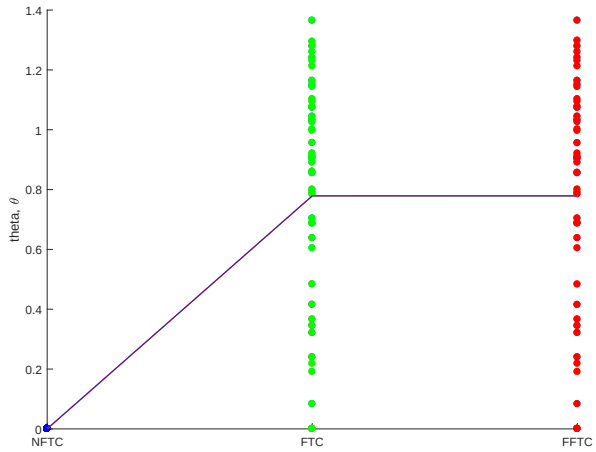

Figure 11.1.6: Comparison of average values of θ for NFTC, FTC and FFTC design methods

The downside of using BSθ is the execution time as it the CVX programme has to be run multiple times to identify a minimum theta. The iterative loop that encases the slowest part of the programme, the CVX method leads to a large sacrifice of system responsiveness. The average run-time for system with 6 outputs is approximately **66**s for FTCS and **9**s for FFTCS. Although SDR has improved speed of execution, it is still almost 4 times greater than other design methods.

γ **vs** θ

For binary search θ algorithms, there exists some effort to find the minimum possible γ for a given minimum θ. However, the ideal γ and θ values are unlikely to coincide. The analysis of the effects of employing $BS\theta$ on the disturbance rejection is an important consideration when using these algorithms. The relative difference between the minimum γ generated in the set of θ and and that of the final γ is noted as γ_{diff_R}, where

$$\gamma_{diff_R} = \frac{\gamma_\theta - \gamma_{min}}{\gamma_{min}} \tag{11.1.1}$$

	NFTC	FTC	FFTC
$\log(\gamma_{diff_R})$	2.18	8.79	7.33

Table 11.1.3: Comparison of γ_{diff_R} for NFTC, FTC and FFTC design using BSθ

The trade-off of γ for θ is significant, especially for FTC methods and hence, cannot be disregarded to limit the impact of external noise on system performance and stability.

11.1.5 Marginally Stable and Unstable Systems

The availability of a stable closed loop system is taken for granted in this project. The performance of FTCS against marginally stable and unstable systems is a further study into the flexibility of these design techniques as internal system stability may not always be true in real-world scenarios. The implementation of randomly generated marginally stable and unstable systems can be found in Appendix A.

		$\mathbf{A}_{stab}$	$\mathbf{A}_{mstab}$	$\mathbf{A}_{unstab}$
H_∞ LS	NFTC	100	100	100
	FTC	100	100	0
	FFTC	100	99	0
H_∞ RPP	NFTC	100	100	100
	FTC	56	0	0
	FFTC	55	0	0

Table 11.1.4: Number of systems that produced a feasible solution for stable, marginally stable and unstable systems

The FTCS design for unstable systems is unachievable with these design methods. The FTCS design for marginally stable systems is only achievable with the more flexible LS methods with satisfactory robustness. The results of this test show that the design of FTCS for unstable systems will be incredibly challenging and that the effectiveness of design methods formulated in this report are best suited for stable systems.

		$\mathbf{A}_{stab}$	$\mathbf{A}_{mstab}$	$\mathbf{A}_{unstab}$
H_∞ LS	NFTC	29.50%	54.25%	60.94%
	FTC	0%	4.8%	NA
	FFTC	0%	4.92	NA
H_∞ RPP	NFTC	38.31%	57.19%	64.06%
	FTC	0%	NA	NA
	FFTC	0%	NA	NA

Table 11.1.5: % instability for each design method for stable, marginally stable and unstable systems

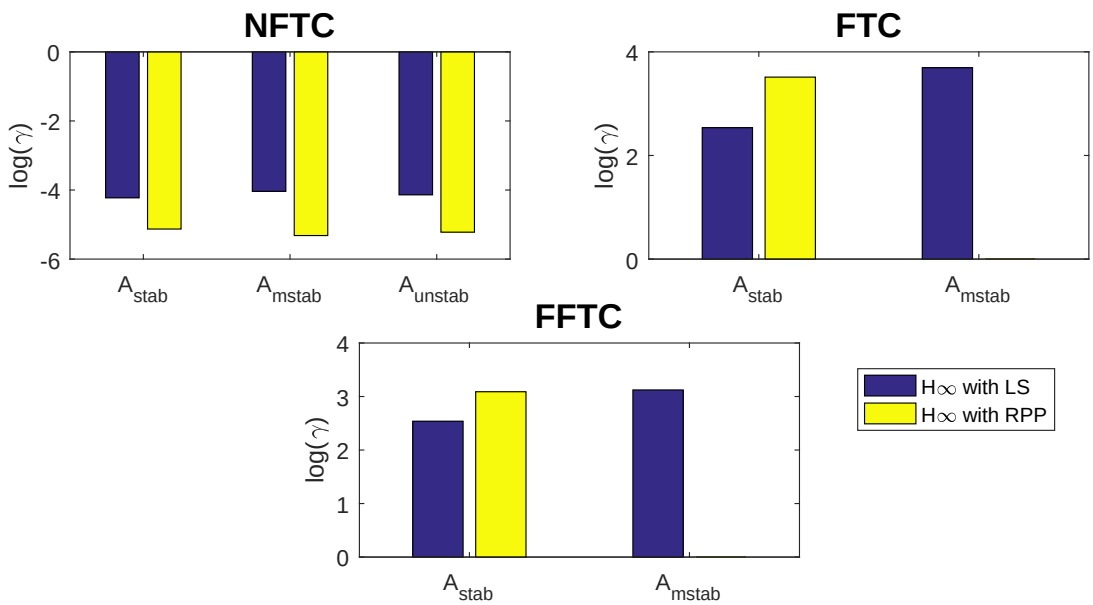

Figure 11.1.7: Comparison of average values of γ for NFTC, FTC and FFTC design methods for stable, marginally stable and unstable systems

The comparison of performance is carried out based on Figure 11.1.7. The FTCS experience some performance degradation when building on marginally stable systems. The RPP method shows superior performance for NFTC design, alas it is outperformed by the LS technique for FTCS design. The use of SDR seems to also provide some performance enhancement, albeit small.

11.2 Minimum Fault Tolerance Threshold

The tests for minimum fault tolerance threshold compare the flexibility, stability and performance of FTC and FFTC design methods against p, the minimum fault tolerance threshold. The assessment of system flexibility and performance is unchanged. However, the stability measure for p-constrained FTC design is further classified by the number of faulty sensors for a given Δ that causes instability. For the scope of this project, the FTCS design is achieved for $p = 0$ to $p = 6$. For each p-constrained controller, the faults that cause instability are recorded and then split based on the number of faulty sensors (from 1 to 5). The final values are then divided by a weighted matrix that takes into account that some fault scenarios are more recurrent than others. For example, there are 64 possible fault scenarios for a system with 6 outputs, of which 20 consist of 3 faulty sensors while only 6 consist of 1 faulty sensor. This approach helps to identify stability with regards to p and the number of faulty sensors while analysing the effect of p on the number of faulty sensors required for instability.

11.2.1 Flexibility of Design Methods

The number of feasible solutions for each design method is outlined in Table 11.2.1. The feasibility increases with p, as the controller becomes less fault tolerant. While BSθ and LS algorithms prove to designable for all degrees of fault, the RPP method requires a higher minimum fault tolerance threshold to improve applicability to any random system.

		p					
		0	1	2	3	4	5
H_∞ LS	FTC	100	100	100	100	100	100
	FFTC	100	100	100	100	100	100
H_∞ RPP	FTC	45	78	96	100	100	100
	FFTC	47	78	83	90	99	100
BSθ, RPP	FTC	100	100	100	100	100	100
	FFTC	100	100	100	100	100	100

Table 11.2.1: Number of systems that produced a feasible solution for each p for all design methods

11.2.2 Performance of Design Methods

The primary performance comparison is that of the different design methods with p as shown in Figure 11.2.1. As expected, γ increases with decreasing fault tolerance as the trade-off between performance and stability is unavoidable.

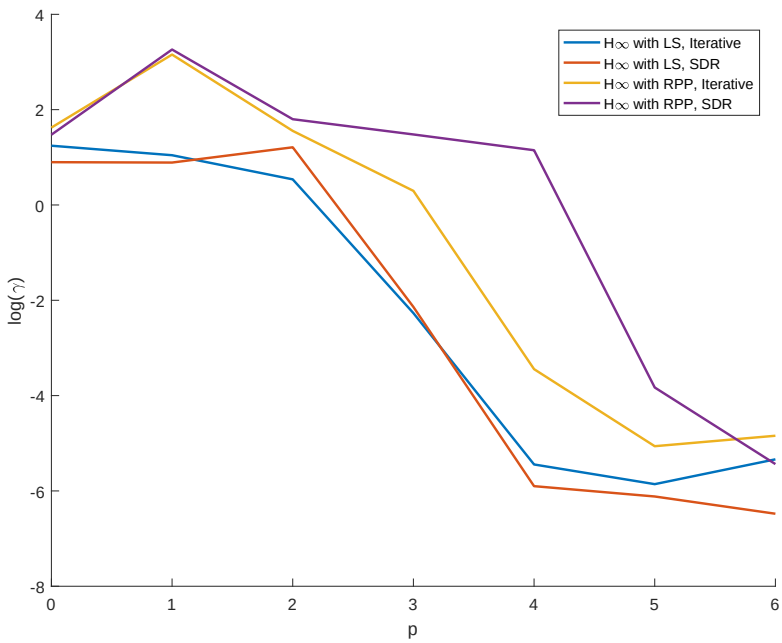

Figure 11.2.1: Comparison of γ for H_∞ with LS, FTC and FFTC and H_∞ with RPP, FTC and FFTC design methods against p

The LS method provides better performance for varying p as compared to RPP. The use of SDR seems more effective on the LS method with the RPP method experiencing some optimality loss due to SDR. The trade-off between stability and performance is also greater for RPP design methods.

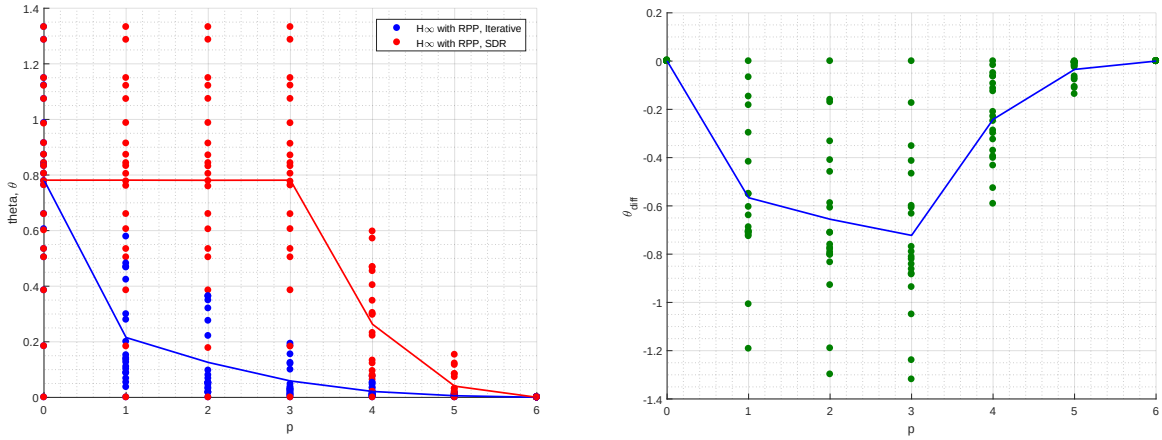

Figure 11.2.2: (a) Comparison of θ for H_∞ with RPP, FTC and FFTC design methods against p, and (b) Comparison of θ loss due to SDR against p

Similarly, the tests are carried out for θ using BSθ design algorithms. As foreseen, θ decreases with decreasing fault tolerance. The SDR technique leads to great loss of optimality for this algorithm with varying p. Since both the RPP methods face some performance loss due to SDR with varying p, the problem most likely lies with SDR RPP formulation.

11.2.3 Stability of Design Methods

The stability matrix has a three-dimensional form, where the x-axis represents the instability of the system due to the presence of 1 faulty sensor to 5 faulty sensors. The y-axis represents the p for which the controller has been designed for. The comparison of the H_∞ methods are given in Figure 11.2.3 while Figure 11.2.4 produces the comparison matrix for BSθ algorithms.

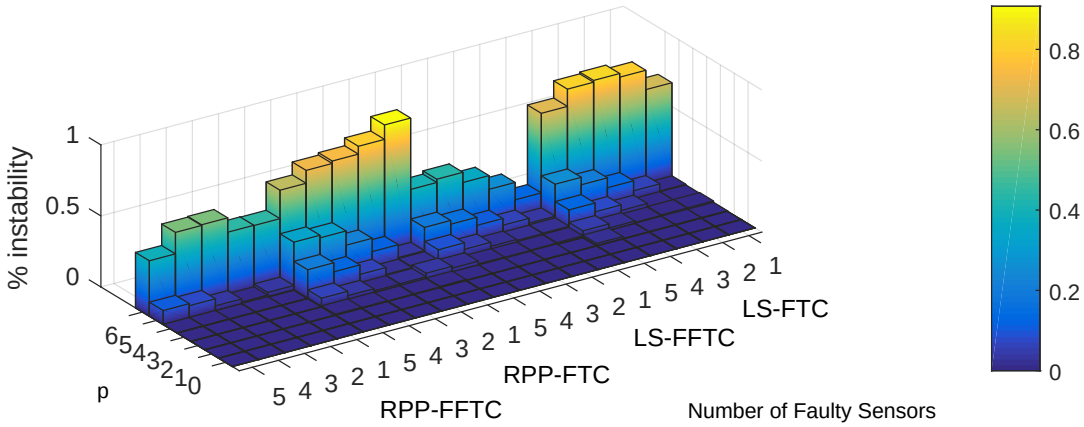

Figure 11.2.3: Comparison of % modes of instability for each number of faults for H_∞ with LS, FTC and FFTC and H_∞ with RPP, FTC and FFTC design methods against p

The system stability is well preserved from $p = 0$ to $p = 3$, with instabilities arising after due to the lack of fault tolerance. It is interesting to note that the greatest % instability is not caused by the greatest number of fault sensors. As discussed previously, the results are normalised using a weight matrix, such that the probability of occurrence is taken out of the equation. The results are thus purely a reflection of the impact the number of faulty sensors has on a system. Another remarkable result is the enhancement of stability caused by the application of SDR.

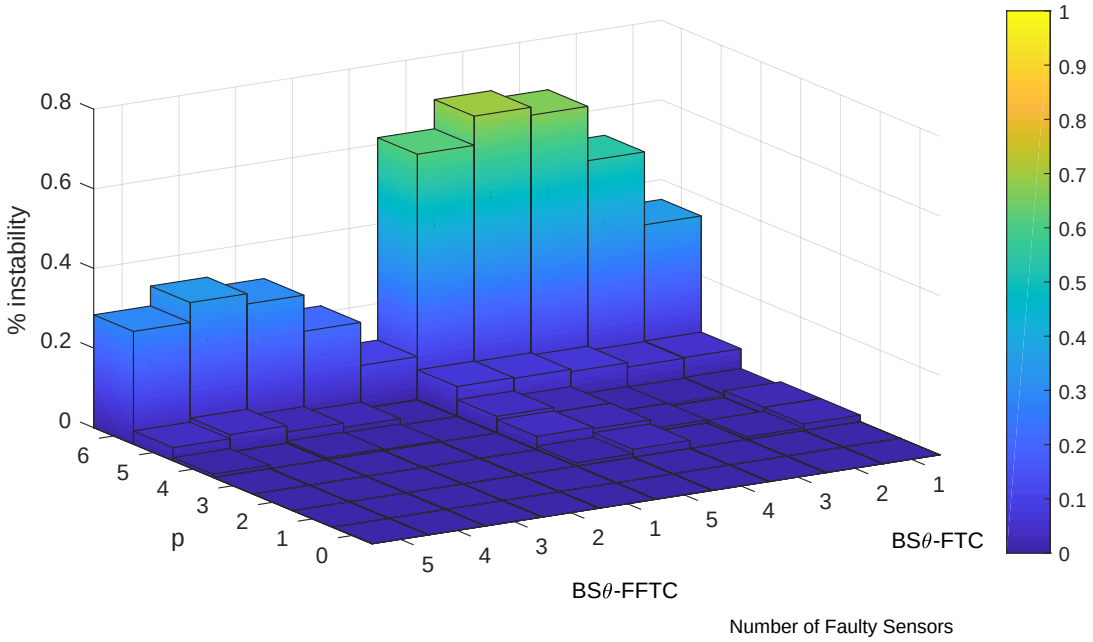

Figure 11.2.4: Comparison of % modes of instability for each number of faults for BSθ, RPP, FTC and FFTC design methods against p

The BSθ exhibits the same suppression of instability using SDR techniques as seen in H_∞ design methods.

11.3 Linked Sensor Faults

The testing of linked sensors involves the design of FTC system without linked sensors and the design of 2 FTC systems with linked sensors. The first system has 3 pairs of linked sensors, hence the diagonal of uncertainty matrix can be partitioned ito 32×2 matrices. This controller design is identified as $\text{FTC}_{L_{2\times 2}}$. The second system has 2 blocks of 3 linked sensors and as is referred to as $\text{FTC}_{L_{3\times 3}}$. The H_∞ with RPP FFTC controller is also tested using the new formulation based on block Hadamard product (BHP).

11.3.1 Flexibility of Design Methods

		p					
		0	1	2	3	4	5
H_∞ LS	FTC	100	100	100	100	100	100
	FFTC	100	100	100	100	100	100
	$\text{FTC}_{L_{2\times2}}$	48	100	100	100	100	100
	$\text{FTC}_{L_{3\times3}}$	48	100	100	100	100	100
	$\text{FFTC}_{L_{2\times2}}$	100	100	100	100	100	100
	$\text{FFTC}_{L_{3\times3}}$	100	100	100	100	100	100
H_∞ RPP	FTC	48	86	100	100	100	100
	FFTC	60	76	80	80	99	100
	FFTC_{BHP}	98	100	98	100	99	100
	$\text{FTC}_{L_{2\times2}}$	100	100	100	100	100	100
	$\text{FTC}_{L_{3\times3}}$	100	100	100	100	100	100
	$\text{FFTC}_{L_{2\times2}}$	100	100	100	100	100	100
	$\text{FFTC}_{L_{3\times3}}$	100	100	100	100	100	100

Table 11.3.1: Number of systems that produced a feasible solution for each p for linked and unlinked FTC and FFTC

The design of linked sensors significantly improves feasibility as the number of possible fault scenarios are reduced. More importantly, the use BHP has boosted the feasibility of H_∞ with RPP FFTC design method.

11.3.2 Performance of Design Methods

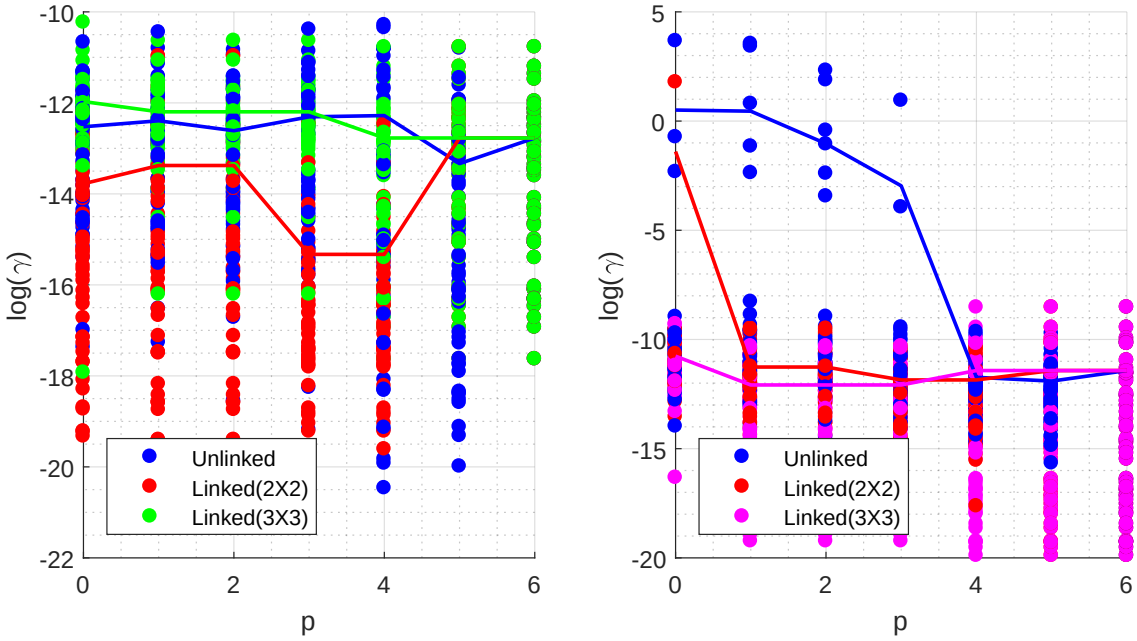

Figure 11.3.1: Comparison of using unlinked sensors design method with linked sensors design method for FTC design against p: (a) for H_∞ with LS desig, and (b) for H_∞ with RPP design

The testing of FTCS, seen in Figure 11.3.2, shows how the linked sensor design greatly enhances the RPP design performance. The performance differences are minute for the LS design approach. The increase of the number of links (from 2 to 3) also improves RPP FTC performance even further, especially for the beginning values of p. Performance enhancement is expected due to the reduction of fault scenarios. A similar approach is undertaken for FFTC design displayed in Figure 11.3.2. The results are very different to FTC implementation. There is extensive performance degradation as compared to the FTC. This could possibly be due to the use of BHP and the way it has been formulated into the CVX programme. The improvement of FFTCS design for the case of linked sensors could possibly be an area of interest for future students due to the favourable characteristics seen from the feasibility discussion and FTC analysis.

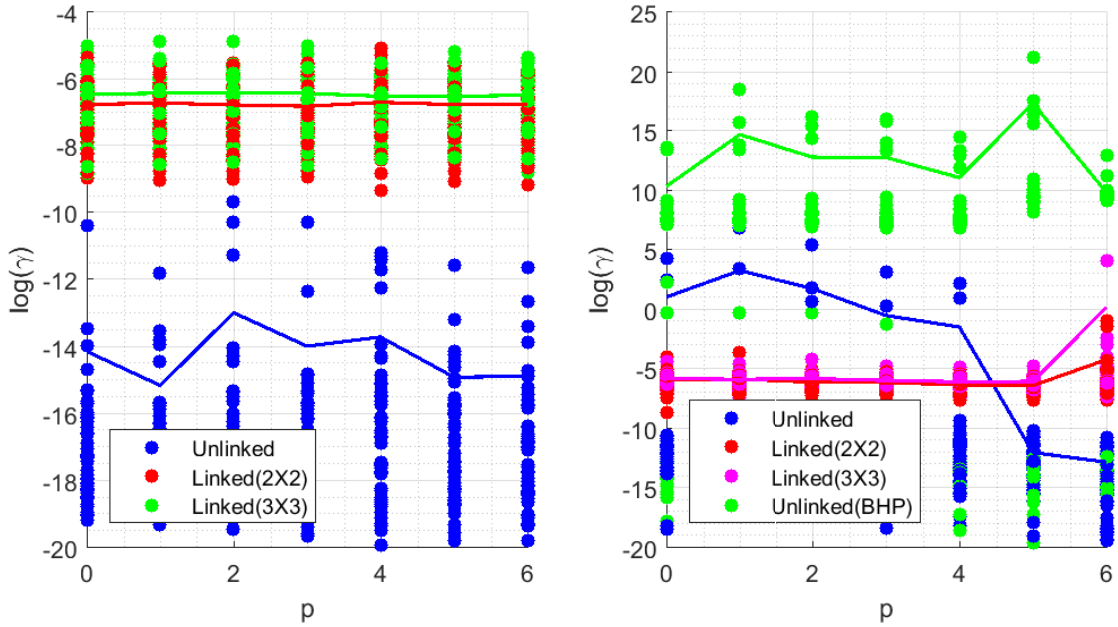

Figure 11.3.2: Comparison of using unlinked sensors design method with linked sensors design method for FFTC design against p: (a) for H_∞ with LS design, and (b) for H_∞ with RPP design

11.3.3 Stability of Design Methods

The stability analysis will be performed with respect to all possible fault scenarios. Since the sensors are linked and are not designed for all the possible uncertainties, some loss in stability is expected. The study of stability is primarily to investigate what would happen if the sensors are not as linked as previously thought and to analyse the differences brought about by SDR.

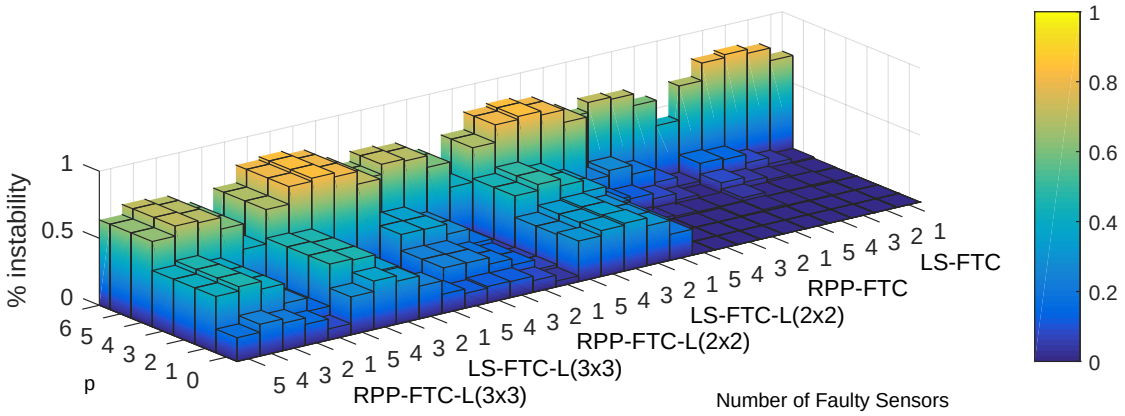

Figure 11.3.3: Comparison of % modes of instability for each number of faults in linked and unlinked sensor FTC designs against p

Figure 11.3.3 depicts the linked FTC design as possessing good robustness in general. The incorporation of linked faults has produced greater instability across the plane. The instability increases with p. The RPP deign method provides a greater degree of stability as compared to the LS design technique. Any possible unlinking of sensors in the system could be catastrophic for the system.

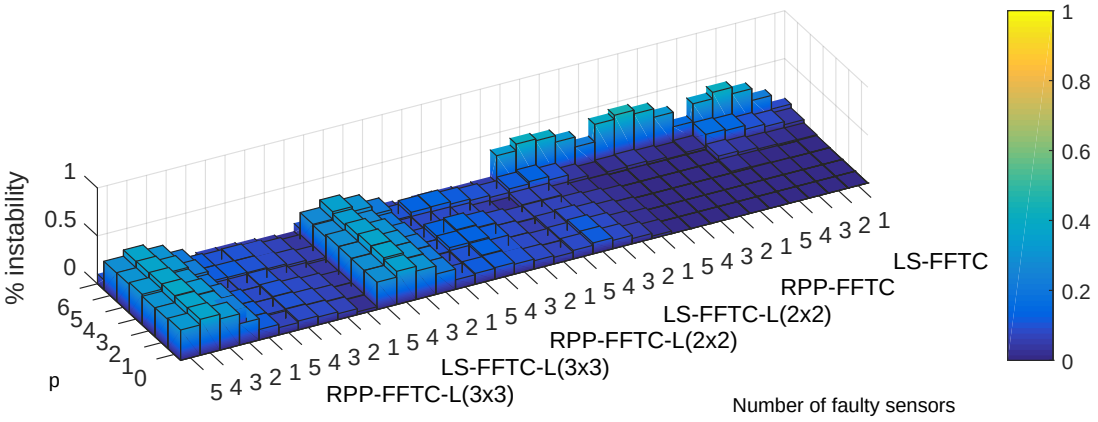

Figure 11.3.4: Comparison of % modes of instability for each number of faults in linked and unlinked sensor FFTC designs against p

The FFTC system stability is much more improved than that of the FTC.There is very little variation with p and the faults are spread evenly across the plane. The FFTC design makes up for its poor performance with good robustness. In the event of sensor unlinking, the system will still have a high probability of being stable.

11.4 Hybrid Fault Tolerant Control

The testing Hybrid FTC (HFTC) implementation is brief and involves only the implementation the the LS criterion, iterative and relaxed. Figure 11.4.1 summarises the test data. The performance deteriorates as the number of active sensors increase as it requires a greater degree of active FTC. The iterative and SDR methods are well matched to each other, which is promising for the development of design methods with low complexity. The stability is preserved for 4 and 5 active sensors, with instability increasing as the number of active sensors decrease. The stability of the system is measured by running it though a simulation for every possible Δ and checking to see if any of the simulations become unstable (reach a very large number).

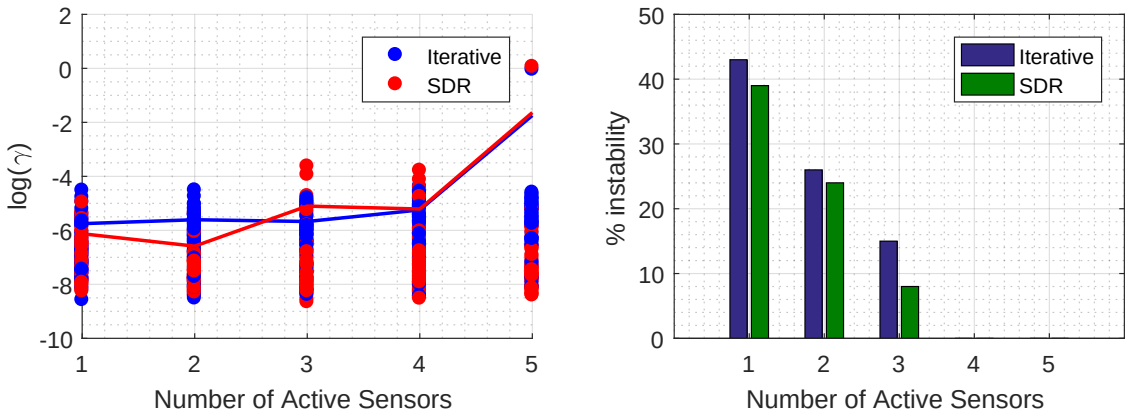

Figure 11.4.1: Comparison of Hybrid FTC and FFTC design methods: (a) in terms of performance, and (b) in terms of stability

The design of HFTC is in its early stages and more work can be done to understand the reason behind the system instability given the robust stability theory from Chapter 10. Further discussion with regards to future work in this area will be carried out in Chapter 13.

11.5 Summary of Results

Ins summary, H_∞ with LS design methods are superior to H_∞ with RPP methods in terms of flexibility, stability and performance. The fundamental reason for this is the additional restrictions posed on RPP design. However, RPP design methods are still invaluable for controllers with performance criteria of r, α or θ. Moreover, the use of SDR has proven to provide satisfactory performance, stability with immense improvement in speed. The pareto-optimality of performance and fault tolerance has also been noted throughout this project. The testing has also proved that the formulated design methods are only ideal for systems that are already stable.

The project has proposed two new concepts, namely linked sensor FTC and hybrid FTC. The use of BHP improves the stability and feasibility of H_∞ with RPP methods and the incorporation of linked sensors enhances the performance of H_∞ with RPP methods. The results of the FFTC design with linked sensors, however do not correspond with the FTC results, indicating a need for better relaxed formulation of the problem. The introduction

of hybrid FTC proves that it is possible to achieve 100% stability even with fewer active sensors, although it calls for some performance sacrifice.

Chapter 12

Evaluation

The project is evaluated based on the achievement of the deliverables outlined in Chapter 1. The first project aim was to develop the theory for nominal control methods. The project requires the knowledge of various matrix mathematics tools and an in-depth understanding of state-space control theory. The extensive research undertaken to understand the relevant concepts are summarised in Chapter 4 of the report, where nominal control design methods have been formulated for the state observer.

Secondly, the design of active FTCS is outlined in Chapter 6 and 7. The formulation of these problems as LMIs are run through meticulously, with the algorithms designed stated explicitly. Following the design of the algorithms, the implementation is carried out in MATLAB using the CVX environment. The use of CVX toolbox simplifies the implementation, thus allowing for more time to work on FTCS concepts.

Thirdly, the development of more realistic fault scenarios is achieved in Chapter 9 with the design of linked sensors. The incorporation of linked sensors and the approach of using block Hadamard product is a novel idea, with great potential for future work and application. The ability of BHP to improve the feasibility and stability of RPP design methods, along with ability of linked sensor designs to outperform other design methods calls for a more in-depth analysis of this topic.

The design of FTC with linked sensors is also a way of increasing the practical ability of FTCS. The project also outlines other ways of increasing the practical applicability of FTCS in Chapter 8 and 10. The idea of minimum fault tolerance threshold is a very effective way of studying performance-stability trade-off and allows designers to choose the trade-off they desire. The brief discussion about hybrid FTC followed by the promising results could pave the way for a controller with equivalent performance as other active FTCS designs without the need of full fault detection. The concept of partial fault detection makes the design method more applicable to wider range of systems where fault

detection is too costly.

Finally, the simulation and tests of the various design methods along with the results are presented in Chapter 11 along with the analysis of the results. The project has focused on the learning of existing concepts and developing of novel ideas that can be used to further the research into FTCS.

Chapter 13

Conclusion & Future Work

The primary objectives of the project have been achieved along with the formulation of two novel concepts, with promising results. The project has formulated and tested a design approach for linked sensors which has shown improved performance, stability and flexibility for some of the design methods. Moreover, it has begun to outline the process of building a hybrid FTC, which could have been delved deeper into with additional time.

The main challenge of the project is finding new and viable paths of research as it is a hotly researched topic. FTCS design also does not have uniform terminology or an integrated one size fits all solution which make the possibilities of research endless and not always rewarding. The project is also implemented purely on computer simulations where response times and switching delays are not taken into account.

Based on the work done on this project two different directions of future work can be proposed. The first would be the enhancement of the FFTC method for linked sensors and the second would be the continued formulation of the hybrid FTC system.

Bibliography

[1] Zhang Y, Jiang J. Bibliographical review on reconfigurable fault-tolerant control systems. Annual Reviews in Control. 2008 Dec;32(2):229–252. Available from: https://ac.els-cdn.com/S1367578808000345/1-s2.0-S1367578808000345-main.pdf?_tid=81cee438-028d-11e8-a4eb-00000aab0f26&acdnat=1516966922_4694f3d64572312447e4cb08ea650577.

[2] Lee KS, Park TG. Robust fault detection observer design under fault sensitivity constraints. Journal of the Franklin Institute. 2015;352:1791–1810.

[3] Blanke M, Staroswiecki M, Wu NE. Concepts and methods in fault-tolerant control. In: Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148). IEEE; 2001. p. 2606–2620 vol.4. Available from: http://dx.doi.org/10.1109/acc.2001.946264.

[4] Luo Zq, Ma Wk, So A, Ye Y, Zhang S. Semidefinite Relaxation of Quadratic Optimization Problems. IEEE Signal Processing Magazine. 2010 May;27(3):20–34. Available from: http://dx.doi.org/10.1109/msp.2010.936019.

[5] Segundo Sevilla FR, Jaimoukha IM, Chaudhuri B, Korba P. A Semidefinite Relaxation Procedure for Fault-Tolerant Observer Design. IEEE Transactions on Automatic Control. 2015 Dec;60(12):3332–3337. Available from: http://dx.doi.org/10.1109/tac.2015.2418681.

[6] Frank PM. Trends in Fault-Tolerant Control of Engineering Systems. IFAC Proceedings Volumes. 2004;37(15):377 – 384. 11th IFAC Symposium on Automation in Mining, Mineral and Metal Processing (MMM'04), Nancy, France, September 8-10, 2004. Available from: http://www.sciencedirect.com/science/article/pii/S1474667017310534.

[7] Harold Lee J. Failure Detection in Linear Systems. Dept. Aero. and Asfro., M.I. T Cambridge, Mass. USA; 1973.

[8] Jiang J. Fault tolerant Control Systems An Introductory Overview. Zidonghua Xuebao/Acta Automatica Sinica. 2005 Jan;31:161–174. Available from: http://www.aas.net.cn/EN/abstract/abstract15502.shtml.

[9] Patton RJ. Fault-tolerant control systems: The 1997 situation, in IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes. In: IFAC SAFEPROCESS'1997; 1997. p. 1033–1054.

[10] Zhao Z, Yang Y, Ding SX, Li L. Fault-Tolerant Control for Systems With Model Uncertainty and Multiplicative Faults. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2017;p. 1–11.

[11] Zhang C, Jaimoukha I. Fault-Tolerant Controller Design with A Tolerance Measure for Systems with Actuator and Sensor Faults. IEEE; 2017. p. 4129–4134. Available from: http://gateway.webofknowledge.com/gateway/Gateway.cgi?GWVersion=2&SrcApp=PARTNER_APP&SrcAuth=LinksAMR&KeyUT=WOS:000427033304032&DestLinkType=FullRecord&DestApp=ALL_WOS&UsrCustomerID=1ba7043ffcc86c417c072aa74d649202.

[12] Zhang C, Jaimoukha IM, Sevilla FRS. Fault-tolerant observer design with a tolerance measure for systems with sensor failures. In: 2016 American Control Conference (ACC); 2016. p. 7523–7528.

[13] Benosman M, Lum KY. Application of absolute stability theory to robust control against loss of actuator effectiveness. IET Control Theory Applications. 2009 June;3(6):772–788.

[14] Yang GH, Wang JL, Soh YC. Reliable LQG control with sensor failures. IEE Proceedings - Control Theory and Applications. 2000 Jul;147(4):433–439.

[15] Scherer C, Gahinet P, Chilali M. Multiobjective output-feedback control via LMI optimization. IEEE Transactions on Automatic Control. 1997 Jul;42(7):896–911. Available from: http://dx.doi.org/10.1109/9.599969.

[16] Jaimoukha IM. Design of Linear Multivariable Control Systems. 2018 Jan;.

[17] Horn RA, Johnson CR. Topics in matrix analysis. Cambridge University Press; 2010. Available from: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0521467136.

[18] Caro-Lopera FJ, Leiva V, Balakrishnan N. Connection between the Hadamard and matrix products with an application to matrix-variate BirnbaumSaunders distributions. Journal of Multivariate Analysis. 2012;104(1):126 – 139. Available from: http://www.sciencedirect.com/science/article/pii/S0047259X11001461.

[19] P H Styan G. Hadamard products and multivariate statistical analysis. 1973 12;6:217–240.

[20] Günther M, Klotz L. Schur's theorem for a block Hadamard product. Linear Algebra and its Applications. 2012 Apr;437:948–956. Available from: https://ac.els-cdn.com/S0024379512002650/1-s2.0-S0024379512002650-main.pdf?_tid=4650b529-95f9-46d1-9387-d5bcbb54bb29&acdnat=1528127782_b1121a00909649897545305f4b1583c3.

[21] Zhang F, editor. The Schur Complement and Its Applications. vol. 4 of Numerical Methods and Algorithms. New York: Springer-Verlag; 2005. Available from: http://dx.doi.org/10.1007/b105056.

[22] Boyd SP. Linear matrix inequalities in system and control theory. SIAM; 1994. Available from: http://www.worldcat.org/isbn/089871334.

[23] Astolfi A. Systems and Control Theory - An Introduction; 2006.

[24] Vandenberghe L, Boyd S. SEMIDEFINITE PROGRAMMING. Society for Industrial and Applied Mathematics Review. 1996 Mar;38(1):49–95. Available from: https://www.google.co.uk/_/chrome/newtab?rlz=1C1GCEA_enGB764GB764&ie=UTF-8.

[25] Kautsky J, Nichols NK, Van Dooren P. Robust Pole Assignment in Linear State Feedback. International Journal of Control. 1985;41:1129–1155.

[26] Chilali M, Gahinet P, Apkarian P. Robust pole placement in LMI regions. IEEE Transactions on Automatic Control. 1999 Dec;44(12):2257–2270. Available from: http://dx.doi.org/10.1109/9.811208.

[27] Shahruz SM, Langari G, Tomizuka M. Optimal damping ratio for linear second-order systems. In: [1991] Proceedings of the 30th IEEE Conference on Decision and Control; 1991. p. 2324–2325 vol.3.

[28] CVX Research I. CVX: Matlab Software for Disciplined Convex Programming, version 2.0; 2012. http://cvxr.com/cvx.

Appendix A

System Generation

Listing A.1: sysgen.m

```
n=5; nu=4; ny=6; nd=1; nz=1; bny=2^ny; %generate system
    parameters

% generate stable A matrix
M1=randn(n);
M2=randn(n);
M3=randn(n);
A=M3*(-M1'*M1+(M2-M2'))/M3;
% generate marginally stable A matrix
M1_marg=randn(n-1); M2_marg=randn(n-1); M3_marg=randn(n-1);
T=randn(n);
A_marg=M3_marg*(-M1_marg'*M1_marg+(M2_marg-M2_marg'))/M3_marg;
A_marg = [A_marg, zeros(n-1,1); zeros(1, n)];
A_marg = T\A_marg*T;
%generate unstable A matrix
M1_uns=randn(n-1); M2_uns=randn(n-1); M3_uns=randn(n-1);
T_uns=randn(n);
A_uns=M3_uns*(-M1_uns'*M1_uns+(M2_uns-M2_uns'))/M3_uns;
A_uns = [A_uns, zeros(n-1,1);abs(randn(1,n)) ];
A_uns = T_uns\A_uns*T_uns;

A_mat=[A,A_marg,A_uns];

%generate other system matrices
B=randn(n,nu);
C=randn(ny,n);
```

```
D=zeros(nz,nd);

Cz=randn(nz,n);
Bd=rand(n,nd);
Dd=rand(ny,nd);
%generate faults
Delta=faultsgen(ny);
%generate eigenvalue placeholders
EigenValues=zeros(n,bny);
EigenValuesNFTC=zeros(n,bny); %no fault tolerant control
EigenValuesFTC=zeros(n,bny); %with fault tolerant control
EigenValuesFFTC=zeros(n,bny); %with fast fault tolerant control
```

Listing A.2: faultsgen.m

```
%generates all possible fault matrix delta
function D = faultsgen(m)

D=zeros(m,m,2^m);
for i=1:2^m
    D(:,:,i)=diag(de2bi(i-1,m,'left-msb')-eps);
end

end
```

Appendix B

Fault Tolerant Design

B.1 H_∞ with Lyapunov Stability criterion

Listing B.1: FaultTolerantLS.m

```
%% Fault tolerant with Lyapunov Stability

while(true)
clear; clc;
% clearvars ('-except' ,keepVars{:}) %used for loop testing

%generate system param
sysgen;

%% Non fault tolerant control
cvx_begin sdp
cvx_precision high
variable Y(n,ny);
variable P(n,n) symmetric;
variable Gamma;
minimize (Gamma);
P>=eye(n);


BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
     (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
     Cz,D,-Gamma*eye(nz)];

BRL<=0;
```

```
cvx_end

L_NFTC=inv(P)*Y;
gamma_NFTC=Gamma;
cvx_NFTC=cvx_optval;

%% Fault tolerant control
cvx_begin sdp
cvx_precision high
variable Y(n,ny);
variable P(n,n) symmetric;
variable Gamma;
minimize (Gamma);
P>=eye(n);

for i=1:bny
    %lypanouv stability
    P*A+Y*Delta(:,:,i)*C+(P*A+Y*Delta(:,:,i)*C)'<=0;
end

BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
     (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
     Cz,D,-Gamma*eye(nz)];

BRL<=0;

cvx_end

L_FTC=inv(P)*Y; %generate controller gain
gamma_FTC=Gamma;
cvx_FTC=cvx_optval;

%% Fast fault tolerant control
cvx_begin sdp
cvx_precision high
variable Y(n,ny);
variable P(n,n) symmetric;
variable S(ny,ny) diagonal
variable Gamma;
```

```
minimize (Gamma);
P>=eye(n);


%lypanouv stability with relaxation
[P*A+(P*A)', Y+C'*S';
    Y'+S*C, -S-S']<=0


BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
      (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
      Cz,D,-Gamma*eye(nz)];

BRL<=0;

cvx_end

L_FFTC=inv(P)*Y; %generate controller gain
gamma_FFTC=Gamma;
cvx_FFTC=cvx_optval;

% check if all cvx has been solved-only move on until solved
cvx_check=[cvx_NFTC cvx_FTC cvx_FFTC];
if(isempty(cvx_check(isinf(cvx_check)))==1)
    break;
end

end

% testing controllers using simulation model
testScript;
```

B.2 H_∞ with RPP

Listing B.2: FaultTolerantRPP.m

```
%% Fault tolerant with RPP

 while(true)
clear;clc;
```

```
%testing with loop
% clearvars ('-except' ,keepVars{:})

sysgen;

%% Define variables for rpp
r=100;
alpha=1;
theta=(3/4)*pi/2;

%% Non fault tolerant control
cvx_begin sdp
cvx_precision high
variable Y(n,ny);
variable P(n,n) symmetric;
variable Gamma;
P>=eye(n);
minimise (Gamma);

  XA=P*A+Y*C;

  %half plane
   XA+XA'+ 2*alpha*P <=0;
   %disk
   [-r*P,XA;
        XA', -r*P]<=0;
   %conic sector
   [sin(theta)*(XA+XA'),cos(theta)*(XA-XA');
        -cos(theta)*(XA-XA'),sin(theta)*(XA+XA')]<=0

   BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
     (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
     Cz,D,-Gamma*eye(nz)];

   BRL<=0;

cvx_end

L_NFTC=inv(P)*Y;
```

```
Gamma_NFTC=Gamma;
cvx_NFTC_RPP=cvx_optval;

%% Fault tolerant control
cvx_begin sdp
cvx_precision high
variable Y(n,ny);
variable P(n,n) symmetric;
variable Gamma;
P>=eye(n);
minimise(Gamma);

for i=1:bny

    XA=P*A+Y*Delta(:,:,i)*C;

    %half plane
    XA+XA'+ 2*alpha*P <=0;
    %disk
    [-r*P,XA;
         XA', -r*P]<=0;
    %conic sector
    [sin(theta)*(XA+XA'),cos(theta)*(XA-XA');
         -cos(theta)*(XA-XA'),sin(theta)*(XA+XA')]<=0

     BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
              (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
                   Cz,D,-Gamma*eye(nz)];

     BRL<=0;

end


cvx_end

L_FTC=inv(P)*Y; %generate controller gain
Gamma_FTC=Gamma;
cvx_FTC_RPP=cvx_optval;

```

```
%% Fast Fault Tolerant Controller with G
cvx_begin sdp
cvx_precision high
variable Y(n,ny);
variable P(n,n) symmetric;
variable S_plane(ny,ny) diagonal;
variable S_disk(ny,ny) diagonal;
variable S_conic(2*ny,2*ny) diagonal;
variable Diag(ny,ny) diagonal;
G=[zeros(ny), Diag;
    -Diag, zeros(ny)];
variable Gamma;
P>=eye(n);
minimise(Gamma);

   %half plane
    [P*A+A'*P+2*alpha*P, Y+C'*S_plane';
        Y'+S_plane*C, -S_plane-S_plane']<=0;
   %disk
     T1_disk=[-r*P, P*A;
         A'*P, -r*P];
     T2_disk=[zeros(n,ny); C'];
     T3_disk=[Y', zeros(ny,n)];

     [T1_disk, T3_disk'+T2_disk*S_disk';
         T3_disk+S_disk*T2_disk', -S_disk-S_disk']<=0;
   %conic sector
     T1_conic=[sin(theta)*(P*A+A'*P), cos(theta)*(P*A-A'*P);
             -cos(theta)*(P*A-A'*P), sin(theta)*(P*A+A'*P)];
     T2_conic=[sin(theta/2)*C', -cos(theta/2)*C';
             cos(theta/2)*C', sin(theta/2)*C'];
     T3_conic=[cos(theta/2)*Y', sin(theta/2)*Y';
             -sin(theta/2)*Y', cos(theta/2)*Y'];

     [T1_conic, T3_conic'+T2_conic*S_conic'+ T2_conic*G';
         T3_conic+S_conic*T2_conic'+ G*T2_conic', -S_conic-
            S_conic']<=0;

     BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
     (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
```

```
    Cz,D,-Gamma*eye(nz)];

    BRL<=0;


cvx_end

L_FFTC=inv(P)*Y; %generate controller gain
Gamma_FFTC=Gamma;
cvx_FFTC_RPP=cvx_optval;

% check if all cvx has been solved-only move on until solved
cvx_check=[cvx_NFTC cvx_FTC cvx_FFTC];
if(isempty(cvx_check(isinf(cvx_check)))==1)
    break;
end

end
```

B.3 Binary Search for θ

Listing B.3: FaultTolerantTheta.m

```
%% Fault tolerant with RPP with binary search theta

% clearvars ('-except' ,keepVars{:});clc;
clear;clc
sysgen;

%define variables for rpp
r=100;
alpha=0.1;


%% Non fault tolerant control

% define theta param
theta_U=pi;%upper theta bound
theta_L=0;%lower theta bound
theta_tol=1e-3;
```

```
while(true)
    %set theta at midpoint
    theta_x=(theta_L+theta_U)/2;

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable Gamma;
    P>=eye(n);
    minimise (Gamma);

    XA=P*A+Y*C;

    %half plane
    XA+XA'+ 2*alpha*P <=0;
    %disk
    %
    [-r*P,XA;
        XA', -r*P]<=0;
    %conic sector
    [sin(theta_x)*(XA+XA'),cos(theta_x)*(XA-XA');
        -cos(theta_x)*(XA-XA'),sin(theta_x)*(XA+XA')]<=0;

    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
    (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
    Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    %if no solution found
    if(isinf(cvx_optval))
        %if tolerance reached
        if(theta_U-theta_L<theta_tol/1000)
            %set min theta as upper bound
            theta_final=theta_U;
            %exit loop
```

```
            break;
        end
        %if infeasible for pi/2, no feasible solution exists
        if(theta_x>=pi/2)
            break;
        end
        %if tolerance not reached set current theta as lower bound
        theta_L=theta_x;
    else
        L_NFTC=inv(P)*Y;
        if(theta_U-theta_L<theta_tol)
            theta_final=theta_x;
            break;
        end
        %if tolerance not reached set current theta a upper bound
        theta_U=theta_x;
    end
end

try
theta_min_NFTC=max(abs(abs(angle(eig(A+L_NFTC*C)))-pi));
theta_final_NFTC=theta_final;
catch
    disp('No feasibility found')
    theta_final_NFTC=nan;
end



%% Fault tolerant control

% define theta param
theta_U=pi;%upper theta bound
theta_L=0;%lower theta bound
theta_tol=1e-3;

while(true)
    %set theta at midpoint
```

```
    theta_x=(theta_L+theta_U)/2;

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable Gamma;
    P>=eye(n);
    minimise (Gamma);

    for i=1:bny

        XA=P*A+Y*Delta(:,:,i)*C;

        %half plane
        XA+XA'+ 2*alpha*P <=0;
        %disk
        [-r*P,XA;
            XA', -r*P]<=0;
        %conic sector
        [sin(theta_x)*(XA+XA'),cos(theta_x)*(XA-XA');
            -cos(theta_x)*(XA-XA'),sin(theta_x)*(XA+XA')]<=0

    end


    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
    (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
    Cz,D,-Gamma*eye(nz)];

    BRL<=0;


    cvx_end

    %if no solution found
    if(isinf(cvx_optval))
        %if tolerance reached
        if(theta_U-theta_L<theta_tol/1000)
            %set min theta as upper bound
```

```
            theta_final=theta_U;
            %exit loop
            break;
        end
        %if infeasible for pi/2, no feasible solution exists
        if(theta_x>=pi/2)
            break;
        end
        %if tolerance not reached set current theta as lower bound
        theta_L=theta_x;
    else
        L_FTC=inv(P)*Y;%set controller gain for eigenvalues
        if(theta_U-theta_L<theta_tol)
            theta_final=theta_x;
            break;
        end
        %if tolerance not reached set current theta a upper bound
        theta_U=theta_x;
    end
end

try
    theta_min_FTC=max(abs(abs(angle(eig(A+L_FTC*C)))-pi));
    theta_final_FTC=theta_final;
catch
    disp('No feasibility found')
    theta_final_FTC=nan;
end

%% Fast Fault tolerant control

% define theta param
theta_U=pi;%upper theta bound
theta_L=0;%lower theta bound
theta_tol=1e-3;

while(true)
    %set theta at midpoint
```

```
    theta_x=(theta_L+theta_U)/2;

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable S_plane(ny,ny) diagonal;
    variable S_disk(ny,ny) diagonal;
    variable S_conic(2*ny,2*ny) diagonal;
    variable Diag(ny,ny) diagonal;
    G=[zeros(ny), Diag;
        -Diag, zeros(ny)];
    variable Gamma;
    P>=eye(n);
    minimise (Gamma);

        %half plane
        [P*A+A'*P+2*alpha*P, Y+C'*S_plane';
            Y'+S_plane*C, -S_plane-S_plane']<=0;
        %disk
        T1_disk=[-r*P, P*A;
            A'*P, -r*P];
        T2_disk=[zeros(n,ny); C'];
        T3_disk=[Y', zeros(ny,n)];

        [T1_disk, T3_disk'+T2_disk*S_disk';
            T3_disk+S_disk*T2_disk', -S_disk-S_disk']<=0;
        %conic sector
        T1_conic=[sin(theta_x)*(P*A+A'*P), cos(theta_x)*(P*A-A'*P);
            -cos(theta_x)*(P*A-A'*P), sin(theta_x)*(P*A+A'*P)];
        T2_conic=[sin(theta_x/2)*C', -cos(theta_x/2)*C';
            cos(theta_x/2)*C', sin(theta_x/2)*C'];
        T3_conic=[cos(theta_x/2)*Y', sin(theta_x/2)*Y';
            -sin(theta_x/2)*Y', cos(theta_x/2)*Y'];

        [T1_conic, T3_conic'+T2_conic*S_conic'+ T2_conic*G';
            T3_conic+S_conic*T2_conic'+ G*T2_conic', -S_conic-S_conic']<=0;

```

```
        BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

        BRL<=0;

    cvx_end

    %if no solution found
    if(isinf(cvx_optval))
        %if tolerance reached
        if(theta_U-theta_L<theta_tol/1000)
            %set min theta as upper bound
            theta_final=theta_U;
            %exit loop
            break;
        end
        %if infeasible for pi/2, no feasible solution exists
        if(theta_x>=pi/2)
            break;
        end
        %if tolerance not reached set current theta as lower bound
        theta_L=theta_x;
    else
        L_FFTC=inv(P)*Y;%set controller gain for eigenvalues
        if(theta_U-theta_L<theta_tol)
            theta_final=theta_x;
            break;
        end
        %if tolerance not reached set current theta a upper bound
        theta_U=theta_x;
    end
end

try
    theta_min_FFTC=max(abs(abs(angle(eig(A+L_FFTC*C)))-pi));
    theta_final_FFTC=theta_final;

```

```
catch
    disp('No feasibility found')
    theta_final_FFTC=nan;
end
```

Appendix C

p-constrained FTC Design

C.1 H_∞ with Lyapunov Stability

Listing C.1: pConstrainedLS.m

```
%% p-constrained with Hinfinity method with Lyapunov Stability

clearvars ('-except' ,keepVars{:}) ;clc;

%generate system param
sysgen;

gamma_BRL=zeros(1,ny+1);
gamma_BRL_relaxed=zeros(1,ny+1);
F_BRL=zeros(ny+1,ny);
F_BRL_relaxed=zeros(ny+1,ny);

%% Fault Tolerant with H infinity, p-constrained

%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable S(ny,ny) diagonal
    variable Gamma;
    minimize (Gamma);
```

```
    P>=eye(n);

    %iterate through all fault matrices
    for k=1:bny

        % check number of working sensors in each fault matrix>=
            p
        if int16(trace(Delta(:,:,k)))>=p
            P*A+Y*Delta(:,:,k)*C+(P*A+Y*Delta(:,:,k)*C)'<=0;
        end
    end

    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_BRL(p+1)=Gamma;
    cvx_BRL(p+1)=string(cvx_status);
    L=inv(P)*Y;
    %computes all eigenvalues
    for i=1:bny
        if(~isinf(cvx_optval))
            EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
        end
    end
    %finds the number of faulty sensors in fault scenarios that
        lead to instability
    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_BRL,2)
            F_BRL(p+1,i)=sum(x1==i);
        end
    end
end

```

```
%% Fault Tolerant with relaxed H infinity, p-constrained

%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable M(ny,ny) symmetric;
    variable S(ny,ny) diagonal
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);
    M>=0;
    R=(ny-p)*eye(ny)-ones(ny);

    %relaxed formulation of lyapunov stabiltiy for all p
    T_1=P*A+(P*A)';
    T_2=C';
    T_3=Y';

    [T_1 + T_2*(M.*R)*T_2', T_3' + T_2*(S'-M.*R);
        T_3 + (S-M.*R)*T_2', -S-S'+(M.*R) ]<=0


    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_BRL_relaxed(p+1)=Gamma;
    cvx_BRL_relaxed(p+1)=string(cvx_status);
    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
```

```
                EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
            end
        end

        if(~isinf(cvx_optval))
            x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
            x1=sum(x'==0);
            for i=1:size(F_BRL_relaxed,2)
                F_BRL_relaxed(p+1,i)=sum(x1==i);
            end
        end
end
```

C.2 H_∞ with RPP

Listing C.2: pConstrainedRPP.m

```
%% p-constrained with Hinfinity method with RPP

clearvars ('-except' ,keepVars{:}) ;clc;
%
% %generate system param
% sysgen;

gamma_RPP=zeros(1,ny+1);
gamma_RPP_relaxed=zeros(1,ny+1);
F_RPP=zeros(ny+1,ny);
F_RPP_relaxed=zeros(ny+1,ny);

r=100;
alpha=1;
theta=(3/4)*pi/2;

%% Fault Tolerant with RPP, p-constrained
%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
```

```
    variable P(n,n) symmetric;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);

    %iterate through all fault matrices
    for k=1:bny

        % check number of working sensors in each fault matrix>=
            p
        if int16(trace(Delta(:,:,k)))>=p

            AX=P*A+Y*Delta(:,:,k)*C;

            %half plane
            AX+AX'+ 2*alpha*P <=0;
            %disk
            [-r*P,AX;
                AX', -r*P]<=0;
            %conic sector
            [sin(theta)*(AX+AX'),cos(theta)*(AX-AX');
                -cos(theta)*(AX-AX'),sin(theta)*(AX+AX')]<=0
        end
    end

    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_RPP(p+1)=Gamma;
    cvx_RPP(p+1)=string(cvx_status);
    L=inv(P)*Y;

        for i=1:bny
        if(~isinf(cvx_optval))
            EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
```

```
        end
       end

    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_RPP,2)
            F_RPP(p+1,i)=sum(x1==i);
        end
    end

end


%% Fault Tolerant with relaxed RPP, p-constrained

%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable M_disk(ny,ny) symmetric;
    variable M_plane(ny,ny) symmetric;
    variable M_conic(2*ny,2*ny) symmetric;
    variable S_disk(ny,ny) diagonal
    variable S_plane(ny,ny) diagonal
    variable S_conic(2*ny,2*ny) diagonal;
    variable Diag(ny,ny) diagonal;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);
    M_disk>=0;
    M_plane>=0;
    M_conic>=0;
    G=[zeros(ny), Diag;
    -Diag, zeros(ny)];
    R=(ny-p)*eye(ny)-ones(ny);
    R_conic=2*(ny-p)*eye(2*ny)-ones(2*ny);
```

```
%relaxed formulation of iterative method with p

%half plane
T1_plane=P*A+A'*P+2*alpha*P;
T2_plane=C';
T3_plane=Y';

plane_relaxed=[T1_plane+T2_plane*(M_plane.*R)*T2_plane', T3_plane'+T2_plane*(S_plane'-M_plane.*R);
    T3_plane+(S_plane-M_plane.*R)*T2_plane', -S_plane-S_plane'+(M_plane.*R)];
plane_relaxed<=0;
%disk
 T1_disk=[-r*P, P*A;
     A'*P, -r*P];
 T2_disk=[zeros(n,ny); C'];
 T3_disk=[Y', zeros(ny,n)];

 disk_relaxed=[T1_disk + T2_disk*(M_disk.*R)*T2_disk', T3_disk'+T2_disk*(S_disk'-M_disk.*R);
     T3_disk+(S_disk-M_disk.*R)*T2_disk', -S_disk-S_disk'+(M_disk.*R)];
 disk_relaxed<=0;

%conic sector
 T1_conic=[sin(theta)*(P*A+A'*P), cos(theta)*(P*A-A'*P);
         -cos(theta)*(P*A-A'*P), sin(theta)*(P*A+A'*P)];
 T2_conic=[sin(theta/2)*C', -cos(theta/2)*C';
         cos(theta/2)*C', sin(theta/2)*C'];
 T3_conic=[cos(theta/2)*Y', sin(theta/2)*Y';
         -sin(theta/2)*Y', cos(theta/2)*Y'];

 conic_relaxed=[T1_conic+T2_conic*(M_conic.*R_conic)*T2_conic', T3_conic'+T2_conic*(S_conic'+G'-M_conic.*R_conic);
     T3_conic+(S_conic+G-M_conic.*R_conic)*T2_conic', -S_conic-S_conic'+(M_conic.*R_conic)];
 conic_relaxed<=0;
```

```
    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
         (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
         Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_RPP_relaxed(p+1)=Gamma;
    cvx_RPP_relaxed(p+1)=string(cvx_status);
    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
             EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
         end
        end

    if(~isinf(cvx_optval))
         x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
         x1=sum(x'==0);
         for i=1:size(F_RPP_relaxed,2)
             F_RPP_relaxed(p+1,i)=sum(x1==i);
         end
    end
end
```

C.3 Binary Search for θ

Listing C.3: pConstrainedBS.m

```
% clear;clc;

clearvars ('-except' ,keepVars{:}) ;clc;
sysgen;

% define variables for rpp
r=100;
alpha=0.1;

```

```
theta_final_FTCp=nan(1,ny+1);
theta_final_FFTCp=nan(1,ny+1);
F_FTCp=zeros(ny+1,ny);
F_FFTCp=zeros(ny+1,ny);

%% Fault tolerant control with p


for p=0:ny
    % define theta param
    theta_U=pi;%upper theta bound
    theta_L=0;%lower theta bound
    theta_tol=1e-3;

    while(true)
        %set theta at midpoint
        theta_x=(theta_L+theta_U)/2;

        cvx_begin sdp
        cvx_precision high
        variable Y(n,ny);
        variable P(n,n) symmetric;
        variable Gamma;
        P>=eye(n);
        minimise (Gamma);

        %iterate through all fault matrices
        for k=1:bny

            % check number of working sensors in each fault matrix>=p
            if int16(trace(Delta(:,:,k)))>=p

                AX=P*A+Y*Delta(:,:,k)*C;

                %half plane
                AX+AX'+ 2*alpha*P <=0;
                %disk
                [-r*P,AX;
                    AX', -r*P]<=0;
```

```
            %conic sector
            [sin(theta_x)*(AX+AX'),cos(theta_x)*(AX-AX');
                -cos(theta_x)*(AX-AX'),sin(theta_x)*(AX+AX')
                ]<=0
        end
    end

    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    %if no solution found
    if(isinf(cvx_optval))
        %if tolerance reached
        if(theta_U-theta_L<theta_tol/1000)
            %set min theta as upper bound
            theta_final=theta_U;
            %exit loop
            break;
        end
        %if infeasible for pi/2, no feasible solution exists
        if(theta_x>=pi/2)
            break;
        end
        %if tolerance not reached set current theta as lower
            bound
        theta_L=theta_x;
    else
        L_FTCp=inv(P)*Y;%set controller gain for eigenvalues
        if(theta_U-theta_L<theta_tol)
            theta_final=theta_x;
            break;
        end
        %if tolerance not reached set current theta a upper
            bound
        theta_U=theta_x;
```

```
        end
    end
    try
        theta_final_FTCp(p+1)=theta_final;
        for i=1:bny
            if(~isinf(cvx_optval))
                EigenValues(:,i)=eig(A+L_FTCp*Delta(:,:,i)*C);
            end
        end

        if(~isinf(cvx_optval))
            x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
            x1=sum(x'==0);
            for i=1:size(F_FTCp,2)
                F_FTCp(p+1,i)=sum(x1==i);
            end
        end
    catch
        disp('No feasibility found')
    end

end

%% Fast Fault tolerant control with p

for p=0:ny
    % define theta param
    theta_U=pi;%upper theta bound
    theta_L=0;%lower theta bound
    theta_tol=1e-3;

    while(true)
        %set theta at midpoint
        theta_x=(theta_L+theta_U)/2;

        cvx_begin sdp
        cvx_precision high
        variable Y(n,ny);
        variable P(n,n) symmetric;
        variable M_disk(ny,ny) symmetric;
```

```
variable M_plane(ny,ny) symmetric;
variable M_conic(2*ny,2*ny) symmetric;
variable S_disk(ny,ny) diagonal
variable S_plane(ny,ny) diagonal
variable S_conic(2*ny,2*ny) diagonal;
variable Diag(ny,ny) diagonal;
 variable Gamma;
P>=eye(n);
M_disk>=0;
M_plane>=0;
M_conic>=0;
G=[zeros(ny), Diag;
    -Diag, zeros(ny)];
R=(ny-p)*eye(ny)-ones(ny);
R_conic=2*(ny-p)*eye(2*ny)-ones(2*ny);
minimise (Gamma);

%relaxed formulation of iterative method with p

%half plane
T1_plane=P*A+A'*P+2*alpha*P;
T2_plane=C';
T3_plane=Y';

plane_relaxed=[T1_plane+T2_plane*(M_plane.*R)*T2_plane',
    T3_plane'+T2_plane*(S_plane'-M_plane.*R);
    T3_plane+(S_plane-M_plane.*R)*T2_plane', -S_plane-
        S_plane'+(M_plane.*R)];
plane_relaxed<=0;

%disk
T1_disk=[-r*P, P*A;
    A'*P, -r*P];
T2_disk=[zeros(n,ny); C'];
T3_disk=[Y', zeros(ny,n)];

disk_relaxed=[T1_disk + T2_disk*(M_disk.*R)*T2_disk',
   T3_disk'+T2_disk*(S_disk'-M_disk.*R);
    T3_disk+(S_disk-M_disk.*R)*T2_disk', -S_disk-S_disk
        '+(M_disk.*R)];
```

```
        disk_relaxed <=0;

        %conic sector
        T1_conic=[sin(theta_x)*(P*A+A'*P), cos(theta_x)*(P*A-A'*P);
            -cos(theta_x)*(P*A-A'*P), sin(theta_x)*(P*A+A'*P)];
        T2_conic=[sin(theta_x/2)*C', -cos(theta_x/2)*C';
            cos(theta_x/2)*C', sin(theta_x/2)*C'];
        T3_conic=[cos(theta_x/2)*Y', sin(theta_x/2)*Y';
            -sin(theta_x/2)*Y', cos(theta_x/2)*Y'];

        conic_relaxed=[T1_conic+T2_conic*(M_conic.*R_conic)*T2_conic', T3_conic'+T2_conic*(S_conic'+G'-M_conic.*R_conic);
            T3_conic+(S_conic+G-M_conic.*R_conic)*T2_conic', -S_conic-S_conic'+(M_conic.*R_conic)];
        conic_relaxed <=0;


        BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
            (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
            Cz,D,-Gamma*eye(nz)];

        BRL<=0;

        cvx_end

        %if no solution found
        if(isinf(cvx_optval))
            %if tolerance reached
            if(theta_U-theta_L<theta_tol/1000)
                %set min theta as upper bound
                theta_final=theta_U;
                %exit loop
                break;
            end
            %if infeasible for pi/2, no feasible solution exists
            if(theta_x>=pi/2)
                break;
            end
```

```
            %if tolerance not reached set current theta as lower bound
            theta_L=theta_x;
        else
            L_FFTCp=inv(P)*Y;%set controller gain for eigenvalues
            if(theta_U-theta_L<theta_tol)
                theta_final=theta_x;
                break;
            end
            %if tolerance not reached set current theta a upper bound
            theta_U=theta_x;
        end
    end
    try
        theta_final_FFTCp(p+1)=theta_final;
        for i=1:bny
            if(~isinf(cvx_optval))
                EigenValues(:,i)=eig(A+L_FFTCp*Delta(:,:,i)*C);
            end
        end

        if(~isinf(cvx_optval))
            x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
            x1=sum(x'==0);
            for i=1:size(F_FFTCp,2)
                F_FFTCp(p+1,i)=sum(x1==i);
            end
        end
    catch
        disp('No feasibility found')
    end

end
```

Appendix D

Linked Sensors FTC Design

D.1 Linked Faults Generation

Listing D.1: faultsgenLinked.m

```
%generates all possible fault matrix delta for linked sensors
function D_linked = faultsgenlinked(m,l,l_array)
%m=total number of sensors
%l=number of blocks %l_array contains size of each block

D_linked=zeros(m,m,2^l);

for i=1:2^l
        tmp=de2bi(i-1,l,'left-msb')-eps;
        D_linked(:,:,i)=diag(repelem(tmp,l_array));
end

end
```

D.2 Fault Tolerant Control

Listing D.2: linkedSensorFTC.m

```
%% Fault tolerant Control (Linked and Unlinked Sensors
   Comparison)
clearvars ('-except' ,keepVars{:})

%generate system param
```

```
sysgen;
%linked param
l=3;%number of linked blocks
l_array=[2,2,2];% indicates how sensors linked
Delta_L=faultsgenlinked(ny,l,l_array); %linked faults
[~,~,szL]=size(Delta_L);

%linked 2 param
l2=2;%number of linked blocks
l_array2=[3,3];% indicates how sensors linked
Delta_L2=faultsgenlinked(ny,l2,l_array2); %linked faults
[~,~,szL2]=size(Delta_L2);

% generate RPP param
r=100;
alpha=1;
theta=(3/4)*pi/2;

%generate storage vecs
gamma_RPP=zeros(1,ny+1);
gamma_RPP_l=zeros(1,ny+1);
gamma_RPP_l2=zeros(1,ny+1);
gamma_BRL=zeros(1,ny+1);
gamma_BRL_l=zeros(1,ny+1);
gamma_BRL_l2=zeros(1,ny+1);

F_RPP=zeros(ny+1,ny);
F_RPP_l=zeros(ny+1,ny);
F_RPP_l2=zeros(ny+1,ny);
F_BRL=zeros(ny+1,ny);
F_BRL_l=zeros(ny+1,ny);
F_BRL_l2=zeros(ny+1,ny);

%% H infity with Lyapunov Stability, p constrained
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
```

```
    variable S(ny,ny) diagonal
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);

    %iterate through all fault matrices
    for k=1:bny

        % check number of working sensors in each fault matrix>=
            p
        if int16(trace(Delta(:,:,k)))>=p
            P*A+Y*Delta(:,:,k)*C+(P*A+Y*Delta(:,:,k)*C)'<=0;
        end
    end

    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_BRL(p+1)=Gamma;
    L=inv(P)*Y;

    for i=1:bny
        if(~isinf(cvx_optval))
            EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
        end
    end

    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_BRL,2)
            F_BRL(p+1,i)=sum(x1==i);
        end
    end
end
```

```
%% Fault Tolerant with RPP, p-constrained
%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);

    %iterate through all fault matrices
    for k=1:bny

        % check number of working sensors in each fault matrix>=
            p
        if int16(trace(Delta(:,:,k)))>=p

            AX=P*A+Y*Delta(:,:,k)*C;

            %half plane
            AX+AX'+ 2*alpha*P <=0;
            %disk
            [-r*P,AX;
                AX', -r*P]<=0;
            %conic sector
            [sin(theta)*(AX+AX'),cos(theta)*(AX-AX');
                -cos(theta)*(AX-AX'),sin(theta)*(AX+AX')]<=0
        end
    end

    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

```

```
    gamma_RPP(p+1)=Gamma;
    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
             EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
         end
        end

    if(~isinf(cvx_optval))
         x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
         x1=sum(x'==0);
         for i=1:size(F_RPP,2)
             F_RPP(p+1,i)=sum(x1==i);
         end
    end

end
%% H infity with Lyapunov Stability, p constrained, linked sensors(3 blocks)
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable S(ny,ny) diagonal
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);

    %iterate through all fault matrices
    for k=1:szL

        % check number of working sensors in each fault matrix>=p
         if int16(trace(Delta_L(:,:,k)))>=p
             P*A+Y*Delta_L(:,:,k)*C+(P*A+Y*Delta_L(:,:,k)*C)'<=0;
         end
    end
```

```
    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
         (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
         Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_BRL_l(p+1)=Gamma;
    L=inv(P)*Y;

    for i=1:bny
        if(~isinf(cvx_optval))
            EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
        end
    end

    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_BRL_l,2)
            F_BRL_l(p+1,i)=sum(x1==i);
        end
    end
end
%% Fault Tolerant with RPP, p-constrained, linked sensors(3
   blocks)
%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);

    %iterate through all fault matrices
```

```
    for k=1:szL

        % check number of working sensors in each fault matrix>=
            p
        if int16(trace(Delta_L(:,:,k)))>=p

            AX=P*A+Y*Delta_L(:,:,k)*C;

            %half plane
            AX+AX'+ 2*alpha*P <=0;
            %disk
            [-r*P,AX;
                AX', -r*P]<=0;
            %conic sector
            [sin(theta)*(AX+AX'),cos(theta)*(AX-AX');
                -cos(theta)*(AX-AX'),sin(theta)*(AX+AX')]<=0
        end
    end

    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_RPP_l(p+1)=Gamma;
    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
             EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
         end
        end

    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_RPP_l,2)
```

```
            F_RPP_l(p+1,i)=sum(x1==i);
        end
    end

end
%% H infity with Lyapunov Stability, p constrained, linked sensors (2 bocks)
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable S(ny,ny) diagonal
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);

    %iterate through all fault matrices
    for k=1:szL2

        % check number of working sensors in each fault matrix>=p
        if int16(trace(Delta_L2(:,:,k)))>=p
            P*A+Y*Delta_L2(:,:,k)*C+(P*A+Y*Delta_L2(:,:,k)*C)'<=0;
        end
    end

    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_BRL_l2(p+1)=Gamma;
    L=inv(P)*Y;

```

```
    for i=1:bny
        if(~isinf(cvx_optval))
            EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
        end
    end

    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_BRL_l2,2)
            F_BRL_l2(p+1,i)=sum(x1==i);
        end
    end
end
%% Fault Tolerant with RPP, p-constrained, linked sensors (2 blocks)
%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);

    %iterate through all fault matrices
    for k=1:szL2

        % check number of working sensors in each fault matrix>=p
        if int16(trace(Delta_L2(:,:,k)))>=p

            AX=P*A+Y*Delta_L2(:,:,k)*C;

            %half plane
            AX+AX'+ 2*alpha*P <=0;
            %disk
            [-r*P,AX;
```

```
                    AX', -r*P]<=0;
                %conic sector
                [sin(theta)*(AX+AX'),cos(theta)*(AX-AX');
                    -cos(theta)*(AX-AX'),sin(theta)*(AX+AX')]<=0
            end
        end

        BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
            (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
            Cz,D,-Gamma*eye(nz)];

        BRL<=0;

        cvx_end

        gamma_RPP_l2(p+1)=Gamma;
        L=inv(P)*Y;

            for i=1:bny
             if(~isinf(cvx_optval))
                EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
             end
            end

        if(~isinf(cvx_optval))
            x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
            x1=sum(x'==0);
            for i=1:size(F_RPP_l2,2)
                F_RPP_l2(p+1,i)=sum(x1==i);
            end
        end

end
```

D.3 Fast Fault Tolerant Control

Listing D.3: linkedSensorFFTC.m

```
clearvars ('-except',keepVars{:}) ;clc;

```

```
r=100;
alpha=1;
theta=(3/4)*pi/2;

sysgen;

gamma_BRL_relaxed=zeros(1,ny+1);
gamma_RPP_relaxed=zeros(1,ny+1);
gamma_BRL_linked=zeros(1,ny+1);
gamma_RPP_linked=zeros(1,ny+1);
gamma_BRL_linked_2=zeros(1,ny+1);
gamma_RPP_linked_2=zeros(1,ny+1);
gamma_RPP_BHP=zeros(1,ny+1);

F_BRL_relaxed=zeros(ny+1,ny);
F_RPP_relaxed=zeros(ny+1,ny);
F_BRL_linked=zeros(ny+1,ny);
F_RPP_linked=zeros(ny+1,ny);
F_BRL_linked_2=zeros(ny+1,ny);
F_RPP_linked_2=zeros(ny+1,ny);
F_RPP_BHP=zeros(ny+1,ny);


%% Fault Tolerant with relaxed H infinity, p-constrained

%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable M(ny,ny) symmetric;
    variable S(ny,ny) diagonal
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);
    M>=0;
    R=(ny-p)*eye(ny)-ones(ny);

```

```
    %relaxed formulation of lyapunov stabiltiy for all p
    T_1=P*A+(P*A)';
    T_2=C';
    T_3=Y';

    [T_1 + T_2*(M.*R)*T_2', T_3' + T_2*(S'-M.*R);
        T_3 + (S-M.*R)*T_2', -S-S'+(M.*R) ]<=0


    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_BRL_relaxed(p+1)=Gamma;
    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
             EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
         end
    end

    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_BRL_relaxed,2)
            F_BRL_relaxed(p+1,i)=sum(x1==i);
        end
    end
end
%% Fault Tolerant with relaxed RPP, p-constrained
%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
```

```
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable M_disk(ny,ny) symmetric;
    variable M_plane(ny,ny) symmetric;
    variable M_conic(2*ny,2*ny) symmetric;
    variable S_disk(ny,ny) diagonal
    variable S_plane(ny,ny) diagonal
    variable S_conic(2*ny,2*ny) diagonal;
    variable Diag(ny,ny) diagonal;
    variable Gamma;
    minimize (Gamma);
   P>=eye(n);
   M_disk>=0;
   M_plane>=0;
   M_conic>=0;
   G=[zeros(ny), Diag;
   -Diag, zeros(ny)];
   R=(ny-p)*eye(ny)-ones(ny);
   R_conic=2*(ny-p)*eye(2*ny)-ones(2*ny);

  %relaxed formulation of iterative method with p

  %half plane
  T1_plane=P*A+A'*P+2*alpha*P;
  T2_plane=C';
  T3_plane=Y';

  plane_relaxed=[T1_plane+T2_plane*(M_plane.*R)*T2_plane',
      T3_plane'+T2_plane*(S_plane'-M_plane.*R);
        T3_plane+(S_plane-M_plane.*R)*T2_plane', -S_plane-
            S_plane'+(M_plane.*R)];
  plane_relaxed<=0;
  %disk
   T1_disk=[-r*P, P*A;
        A'*P, -r*P];
   T2_disk=[zeros(n,ny); C'];
   T3_disk=[Y', zeros(ny,n)];

   disk_relaxed=[T1_disk + T2_disk*(M_disk.*R)*T2_disk',
      T3_disk'+T2_disk*(S_disk'-M_disk.*R);
```

```
        T3_disk+(S_disk-M_disk.*R)*T2_disk', -S_disk-S_disk'+(M_disk.*R)];
    disk_relaxed<=0;

   %conic sector
    T1_conic=[sin(theta)*(P*A+A'*P), cos(theta)*(P*A-A'*P);
              -cos(theta)*(P*A-A'*P), sin(theta)*(P*A+A'*P)];
    T2_conic=[sin(theta/2)*C', -cos(theta/2)*C';
              cos(theta/2)*C', sin(theta/2)*C'];
    T3_conic=[cos(theta/2)*Y', sin(theta/2)*Y';
              -sin(theta/2)*Y', cos(theta/2)*Y'];

    conic_relaxed=[T1_conic+T2_conic*(M_conic.*R_conic)*T2_conic', T3_conic'+T2_conic*(S_conic'+G'-M_conic.*R_conic);
        T3_conic+(S_conic+G-M_conic.*R_conic)*T2_conic', -S_conic-S_conic'+(M_conic.*R_conic)];
    conic_relaxed<=0;


    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
         (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
         Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_RPP_relaxed(p+1)=Gamma;

    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
              EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
         end
        end

    if(~isinf(cvx_optval))
         x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
         x1=sum(x'==0);
```

```
        for i=1:size(F_RPP_relaxed,2)
            F_RPP_relaxed(p+1,i)=sum(x1==i);
        end
    end
end

%% Fault Tolerant with relaxed H infinity, p-constrained, linked sensors

%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable S(ny,ny) diagonal
    variable M11(2,2) symmetric;
    variable M12(2,2) symmetric;
    variable M13(2,2) symmetric;
    variable M22(2,2) symmetric;
    variable M23(2,2) symmetric;
    variable M33(2,2) symmetric;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);
    M>=0;
    R11=(ny-p)*eye(2)-ones(2);
    R44=-ones(2);

    %relaxed formulation of lyapunov stabiltiy for all p
    T_1=P*A+(P*A)';
    T_2=C';
    T_3=Y';
    M_BHP=[M11*R11,M12*R44,M13*R44;
           M12'*R44,M22*R11,M23*R44;
           M13'*R44,M23'*R44,M33*R11];

    [T_1 + T_2*(M_BHP)*T_2', T_3' + T_2*(S'-M_BHP);
        T_3 + (S-M_BHP)*T_2', -S-S'+(M_BHP) ]<=0
```

```
    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
         (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
         Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_BRL_linked(p+1)=Gamma;
    L=inv(P)*Y;

    for i=1:bny
        if(~isinf(cvx_optval))
            EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
        end
    end

    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_BRL_linked,2)
            F_BRL_linked(p+1,i)=sum(x1==i);
        end
    end
end

%% Fault Tolerant with relaxed RPP, p-constrained, linked
   sensors

%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable M11_plane(2,2) symmetric;
    variable M12_plane(2,2) symmetric;
```

```
    variable M13_plane(2,2) symmetric;
    variable M22_plane(2,2) symmetric;
    variable M23_plane(2,2) symmetric;
    variable M33_plane(2,2) symmetric;
    variable M11_disk(2,2) symmetric;
    variable M12_disk(2,2) symmetric;
    variable M13_disk(2,2) symmetric;
    variable M22_disk(2,2) symmetric;
    variable M23_disk(2,2) symmetric;
    variable M33_disk(2,2) symmetric;
    variable M11_conic(4,4) symmetric;
    variable M12_conic(4,4) symmetric;
    variable M13_conic(4,4) symmetric;
    variable M22_conic(4,4) symmetric;
    variable M23_conic(4,4) symmetric;
    variable M33_conic(4,4) symmetric;
    variable S_disk(ny,ny) diagonal
    variable S_plane(ny,ny) diagonal
    variable S_conic(2*ny,2*ny) diagonal;
    variable Diag(ny,ny) diagonal;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);
    M_disk>=0;
    M_plane>=0;
    M_conic>=0;
    G=[zeros(ny), Diag;
    -Diag, zeros(ny)];
    R11=(ny-p)*eye(2)-ones(2);
    R44=-ones(2);
    R11_conic=2*(ny-p)*eye(4)-ones(4);
    R44_conic=-ones(4);

    M_BHP_plane=[M11_plane*R11,M12_plane*R44,M13_plane*R44;
        M12_plane'*R44,M22_plane*R11,M23_plane*R44;
        M13_plane'*R44,M23_plane'*R44,M33_plane*R11];
    M_BHP_disk=[M11_disk*R11,M12_disk*R44,M13_disk*R44;
        M12_disk'*R44,M22_disk*R11,M23_disk*R44;
        M13_disk'*R44,M23_disk'*R44,M33_disk*R11];
```

```
M_BHP_conic=[M11_conic*R11_conic,M12_conic*R44_conic,M13_conic*R44_conic;
    M12_conic'*R44_conic,M22_conic*R11_conic,M23_conic*R44_conic;
    M13_conic'*R44_conic,M23_conic'*R44_conic,M33_conic*R11_conic];
%relaxed formulation of iterative method with p

%half plane
T1_plane=P*A+A'*P+2*alpha*P;
T2_plane=C';
T3_plane=Y';


plane_relaxed=[T1_plane+T2_plane*(M_BHP_plane)*T2_plane',T3_plane'+T2_plane*(S_plane'-M_BHP_plane);
    T3_plane+(S_plane-M_BHP_plane)*T2_plane', -S_plane-S_plane'+(M_BHP_plane)];
plane_relaxed<=0;
%disk
T1_disk=[-r*P, P*A;
    A'*P, -r*P];
T2_disk=[zeros(n,ny); C'];
T3_disk=[Y', zeros(ny,n)];

disk_relaxed=[T1_disk + T2_disk*(M_BHP_disk)*T2_disk',T3_disk'+T2_disk*(S_disk'-M_BHP_disk);
    T3_disk+(S_disk-M_BHP_disk)*T2_disk', -S_disk-S_disk'+(M_BHP_disk)];
disk_relaxed<=0;

%conic sector
T1_conic=[sin(theta)*(P*A+A'*P), cos(theta)*(P*A-A'*P);
        -cos(theta)*(P*A-A'*P), sin(theta)*(P*A+A'*P)];
T2_conic=[sin(theta/2)*C', -cos(theta/2)*C';
        cos(theta/2)*C', sin(theta/2)*C'];
T3_conic=[cos(theta/2)*Y', sin(theta/2)*Y';
        -sin(theta/2)*Y', cos(theta/2)*Y'];

```

```
    conic_relaxed=[T1_conic+T2_conic*(M_BHP_conic)*T2_conic',
        T3_conic'+T2_conic*(S_conic'+G'-M_BHP_conic);
         T3_conic+(S_conic+G-M_BHP_conic)*T2_conic', -S_conic-
            S_conic'+(M_BHP_conic)];
    conic_relaxed<=0;


    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
         (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
         Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_RPP_linked(p+1)=Gamma;
    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
             EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
         end
        end

    if(~isinf(cvx_optval))
         x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
         x1=sum(x'==0);
         for i=1:size(F_RPP_linked,2)
             F_RPP_linked(p+1,i)=sum(x1==i);
         end
    end
end


%% Fault Tolerant with relaxed H infinity, p-constrained, linked
    sensors

%iterate through all p
for p=0:ny

```

```
    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable S(ny,ny) diagonal
    variable M11(3,3) symmetric;
    variable M12(3,3) symmetric;
    variable M22(3,3) symmetric;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);
    M>=0;
    R11=(ny-p)*eye(3)-ones(3);
    R44=-ones(3);

    %relaxed formulation of lyapunov stabiltiy for all p
    T_1=P*A+(P*A)';
    T_2=C';
    T_3=Y';
    M_BHP=[M11*R11,M12*R44;
           M12'*R44,M22*R11];

    [T_1 + T_2*(M_BHP)*T_2', T_3' + T_2*(S'-M_BHP);
        T_3 + (S-M_BHP)*T_2', -S-S'+(M_BHP) ]<=0


    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
        (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
        Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_BRL_linked_2(p+1)=Gamma;
    L=inv(P)*Y;

    for i=1:bny
        if(~isinf(cvx_optval))
            EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
```

```
        end
    end

    if(~isinf(cvx_optval))
        x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
        x1=sum(x'==0);
        for i=1:size(F_BRL_linked_2,2)
            F_BRL_linked_2(p+1,i)=sum(x1==i);
        end
    end
end

%% Fault Tolerant with relaxed RPP, p-constrained, linked
   sensors

%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable M11_plane(3,3) symmetric;
    variable M12_plane(3,3) symmetric;
    variable M22_plane(3,3) symmetric;
    variable M11_disk(3,3) symmetric;
    variable M12_disk(3,3) symmetric;
    variable M22_disk(3,3) symmetric;
    variable M11_conic(6,6) symmetric;
    variable M12_conic(6,6) symmetric;
    variable M22_conic(6,6) symmetric;
    variable S_disk(ny,ny) diagonal
    variable S_plane(ny,ny) diagonal
    variable S_conic(2*ny,2*ny) diagonal;
    variable Diag(ny,ny) diagonal;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);
    M_disk>=0;
    M_plane>=0;
```

```
    M_conic>=0;
    G=[zeros(ny), Diag;
    -Diag, zeros(ny)];
    R11=(ny-p)*eye(3)-ones(3);
    R44=-ones(3);
    R11_conic=2*(ny-p)*eye(6)-ones(6);
    R44_conic=-ones(6);

    M_BHP_plane=[M11_plane*R11,M12_plane*R44;
        M12_plane'*R44,M22_plane*R11];
    M_BHP_disk=[M11_disk*R11,M12_disk*R44;
        M12_disk'*R44,M22_disk*R11];
    M_BHP_conic=[M11_conic*R11_conic,M12_conic*R44_conic,
        M12_conic'*R44_conic,M22_conic*R11_conic];
  %relaxed formulation of iterative method with p

  %half plane
  T1_plane=P*A+A'*P+2*alpha*P;
  T2_plane=C';
  T3_plane=Y';


  plane_relaxed=[T1_plane+T2_plane*(M_BHP_plane)*T2_plane',
      T3_plane'+T2_plane*(S_plane'-M_BHP_plane);
        T3_plane+(S_plane-M_BHP_plane)*T2_plane', -S_plane-
            S_plane'+(M_BHP_plane)];
  plane_relaxed<=0;
  %disk
    T1_disk=[-r*P, P*A;
        A'*P, -r*P];
    T2_disk=[zeros(n,ny); C'];
    T3_disk=[Y', zeros(ny,n)];

    disk_relaxed=[T1_disk + T2_disk*(M_BHP_disk)*T2_disk',
        T3_disk'+T2_disk*(S_disk'-M_BHP_disk);
        T3_disk+(S_disk-M_BHP_disk)*T2_disk', -S_disk-S_disk'+(
            M_BHP_disk)];
    disk_relaxed<=0;

  %conic sector
```

```
    T1_conic=[sin(theta)*(P*A+A'*P), cos(theta)*(P*A-A'*P);
              -cos(theta)*(P*A-A'*P), sin(theta)*(P*A+A'*P)];
    T2_conic=[sin(theta/2)*C', -cos(theta/2)*C';
              cos(theta/2)*C', sin(theta/2)*C'];
    T3_conic=[cos(theta/2)*Y', sin(theta/2)*Y';
              -sin(theta/2)*Y', cos(theta/2)*Y'];

    conic_relaxed=[T1_conic+T2_conic*(M_BHP_conic)*T2_conic',
        T3_conic'+T2_conic*(S_conic'+G'-M_BHP_conic);
         T3_conic+(S_conic+G-M_BHP_conic)*T2_conic', -S_conic-
            S_conic'+(M_BHP_conic)];
    conic_relaxed<=0;


    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
         (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
         Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_RPP_linked_2(p+1)=Gamma;
    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
              EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
         end
        end

    if(~isinf(cvx_optval))
         x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
         x1=sum(x'==0);
         for i=1:size(F_RPP_linked_2,2)
              F_RPP_linked_2(p+1,i)=sum(x1==i);
         end
    end
end

```

```
%% Fault Tolerant with relaxed RPP, p-constrained, BHP

%generate permutation matrix
idx=[1,7,2,8,3,9,4,10,5,11,6,12];
I=eye(2*ny);
Per=I(idx,:);
%iterate through all p
for p=0:ny

    cvx_begin sdp
    cvx_precision high
    variable Y(n,ny);
    variable P(n,n) symmetric;
    variable M_disk(ny,ny) symmetric;
    variable M_plane(ny,ny) symmetric;
    variable M11_conic(4,4) symmetric;
    variable M12_conic(4,4) symmetric;
    variable M13_conic(4,4) symmetric;
    variable M22_conic(4,4) symmetric;
    variable M23_conic(4,4) symmetric;
    variable M33_conic(4,4) symmetric;
    variable S_disk(ny,ny) diagonal
    variable S_plane(ny,ny) diagonal
    variable S_conic(2*ny,2*ny) diagonal;
    variable Diag(ny,ny) diagonal;
    variable Gamma;
    minimize (Gamma);
    P>=eye(n);
    M_disk>=0;
    M_plane>=0;
    M_conic>=0;
    G=[zeros(ny), Diag;
    -Diag, zeros(ny)];
    R=(ny-p)*eye(ny)-ones(ny);
    R11_conic=2*(ny-p)*eye(4)-ones(4);
    R44_conic=-ones(4);
    M_BHP_conic=[M11_conic*R11_conic,M12_conic*R44_conic,
        M13_conic*R44_conic;
         M12_conic'*R44_conic,M22_conic*R11_conic,M23_conic*
            R44_conic;
```

```
        M13_conic'*R44_conic,M23_conic'*R44_conic,M33_conic*R11_conic];

%relaxed formulation of iterative method with p

%half plane
T1_plane=P*A+A'*P+2*alpha*P;
T2_plane=C';
T3_plane=Y';

plane_relaxed=[T1_plane+T2_plane*(M_plane.*R)*T2_plane', T3_plane'+T2_plane*(S_plane'-M_plane.*R);
      T3_plane+(S_plane-M_plane.*R)*T2_plane', -S_plane-S_plane'+(M_plane.*R)];
plane_relaxed<=0;
%disk
 T1_disk=[-r*P, P*A;
      A'*P, -r*P];
 T2_disk=[zeros(n,ny); C'];
 T3_disk=[Y', zeros(ny,n)];

 disk_relaxed=[T1_disk + T2_disk*(M_disk.*R)*T2_disk', T3_disk'+T2_disk*(S_disk'-M_disk.*R);
      T3_disk+(S_disk-M_disk.*R)*T2_disk', -S_disk-S_disk'+(M_disk.*R)];
 disk_relaxed<=0;

%conic sector
 T1_conic=[sin(theta)*(P*A+A'*P), cos(theta)*(P*A-A'*P);
          -cos(theta)*(P*A-A'*P), sin(theta)*(P*A+A'*P)];
 T2_conic=[sin(theta/2)*C', -cos(theta/2)*C';
          cos(theta/2)*C', sin(theta/2)*C']*Per';
 T3_conic=Per*[cos(theta/2)*Y', sin(theta/2)*Y';
          -sin(theta/2)*Y', cos(theta/2)*Y'];

 conic_relaxed=[T1_conic+T2_conic*(M_conic.*R_conic)*T2_conic', T3_conic'+T2_conic*(S_conic'+G'-M_conic.*R_conic);
      T3_conic+(S_conic+G-M_conic.*R_conic)*T2_conic', -S_conic-S_conic'+(M_conic.*R_conic)];
 conic_relaxed<=0;
```

```
    BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
         (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
         Cz,D,-Gamma*eye(nz)];

    BRL<=0;

    cvx_end

    gamma_RPP_relaxed_BHP(p+1)=Gamma;
    L=inv(P)*Y;

        for i=1:bny
         if(~isinf(cvx_optval))
              EigenValues(:,i)=eig(A+L*Delta(:,:,i)*C);
         end
        end

    if(~isinf(cvx_optval))
         x=dec2bin(find(max(real(EigenValues))>=0)-1)-'0';
         x1=sum(x'==0);
         for i=1:size(F_RPP_relaxed_BHP,2)
              F_RPP_relaxed_BHP(p+1,i)=sum(x1==i);
         end
    end
end
```

Appendix E

Hybrid FTC

Listing E.1: hybridG=FTC.m

```
%% Fault tolerant with loop with BRL
clearvars ('-except' ,keepVars{:});clc;
% clear;clc;

%generate system param
sysgen;

gamma_HFTC=zeros(ny-1,1);
gamma_HFFTC=zeros(ny-1,1);
err=zeros(ny-1,2);

for a=1:5 %number of active sensors
%decompose active and passive
any=2^a;
DeltaH=Delta(:,:,end-any+1:end);
Delta_p=eye(ny-a); %passive fault matrix
Delta_a=faultsgen(a); %active fault matrix

C_p=C(1:ny-a,:);
C_a=C(end-a+1:end,:);


%% Hybrid fault tolerant control, Iterative Loop with Lyapunov
    Stability
cvx_begin sdp
cvx_precision high
```

```
variable Y_a(n,a);
variable Y_p(n,ny-a)
variable P(n,n) symmetric;
variable Gamma;
minimize (Gamma)
P>=eye(n);

Y=[Y_a,Y_p];

for i=1:any
      %lyapunov stability
     P*A+Y_a*Delta_a(:,:,i)*C_a+Y_p*Delta_p*C_p+(P*A+Y_a*Delta_a
        (:,:,i)*C_a+Y_p*Delta_p*C_p)'<=0
end

BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
     (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
     Cz,D,-Gamma*eye(nz)];

BRL<=0;


cvx_end

 L_p=inv(P)*Y_p;
 L_a=inv(P)*Y_a;
 L_FTC=[L_p,L_a];

gamma_HFTC(a)=Gamma;



%% Hybid FFTC
cvx_begin sdp
cvx_precision high
variable Y_a(n,a);
variable Y_p(n,ny-a)
variable P(n,n) symmetric;
variable S(a,a) diagonal
variable Gamma;
```

```
minimize (Gamma);
P>=eye(n);

Y=[Y_a,Y_p];

[P*A+(P*A)'+Y_p*C_p+(Y_p*C_p)', Y_a+C_a'*S';
    Y_a'+S*C_a, -S-S']<=0


BRL=[P*A+(P*A)'+Y*C+(Y*C)',P*Bd+Y*Dd,Cz';
      (P*Bd+Y*Dd)',-Gamma*eye(nd),D';
      Cz,D,-Gamma*eye(nz)];

BRL<=0;

cvx_end

gamma_HFFTC(a)=Gamma;

L_p=inv(P)*Y_p;
L_a=inv(P)*Y_a;
L_FFTC=[L_p,L_a];


for i=1:any
      EigenValuesHFFTC(:,i)=eig(A+L_FFTC*DeltaH(:,:,i)*C);
      EigenValuesHFTC(:,i)=eig(A+L_FTC*DeltaH(:,:,i)*C);
end

%% run simulation on model and check stable
err_true=zeros(2^ny,1);
L=L_FTC;
for i=1:2^ny
D1=Delta(:,:,i);
DH1=blkdiag(eye(2),Delta(3:end,3:end,i));
try
simOut=sim('modelext');
if (sum(Err(Err>1E5))>0)
      err_true(i)=1;
end
```

```
catch
    err_true(i)=1;
end
end

err_true_1=zeros(2^ny,1);
L=L_FFTC;
for i=1:2^ny
D1=Delta(:,:,i);
DH1=blkdiag(eye(2),Delta(3:end,3:end,i));
try
simOut=sim('modelext');
if (sum(Err(Err>1E5))>0)
    err_true_1(i)=1;
end
catch
    err_true_1(i)=1;
end
end


%A+L(Delta_Hat)C stable
EigenValuesHFTC(EigenValuesHFTC>0)
EigenValuesHFFTC(EigenValuesHFFTC>0)
%check sims stable
err(a,:)=[sum(err_true),sum(err_true_1)];

end
```