A Path-Following based Design Framework for Guaranteed Cost Control of Polynomial Fuzzy Systems

Kai-Yi Wong

電気通信大学
機械知能システム学専攻

A thesis submitted to
Department of Mechanical and Intelligent Systems Engineering,
Graduate School of Informatics and Engineering,
The University of Electro-Communications
for the degree of
Doctor of Philosophy in Engineering

Tokyo, Japan
March 2021
A Path-Following based Design Framework for Guaranteed Cost Control of Polynomial Fuzzy Systems

Examining Committee:

Chairman: Prof. Kazuo Tanaka (田中一男先生)
Members: Prof. Osamu Kaneko (金子修先生)
Prof. Aiguo Ming (明愛國先生)
Prof. Motoyasu Tanaka (田中基康先生)
Prof. Kiminao Kosigo (小木曽公尚先生)
概要

多項式ファジシステムに対するコスト保証制御のための
Path-Followingに基づく設計手法

ウォン カイイ

多項式ファジシステムの安定化に関する研究は数多く行われている。しかし、実システム制御で重要な指標である最適性に関係するコスト保証制御に関する研究はほとんど行われていないのが現状である。また、これらの研究では、設計手法の提案を主としており、扱われる設計例題は非線形システムとはいえ、かなりシンプルなものが多く、実システムへの適用可能性に関しては別の議論を行う必要がある。本論文では、多項式ファジシステムに対するコスト保証制御を実現するため、主に2つのことを試みる。一つは、nonconvex設計条件を導出し、これを解くための新しいpath-followingアルゴリズムを提案する。もう一つは、多項式ファジシステムのメンバーシップ関数（非多項式関数）の多項式表現化およびリアプノフ関数の特性を考慮したS-procedureによるsum-of-squares(SOS)設計条件の緩和を実現する。ベンチマーク設計問題を通して、従来手法との比較検討を行い、本設計手法の有効性を明らかにするが、これにとどまらず、電動パラグライダー型無人航空機の制御制御を実システム適用例として取り上げ、提案設計法の実システム適用可能性を検討する。

本論文は6章で構成され、概要は以下の通りである。

第1章では緒論を述べる。本研究の背景や目的を述べ、他の関連手法に対する本研究の位置付けを説明する。

第2章では、本研究の対象システムであるファジシステム/多項式ファジシステムおよび、コスト保証制御やpath-followingアルゴリズムの基礎的事項について述べるとともに、本論文で扱う設計条件導出や解法において重要な役割を担うsum-of-squares(SOS), S-procedure, co-positive relaxationについて述べる。

第3章では、新たにnonconvex SOS設計条件を導出し、その条件を解くためのpath-followingアルゴリズムを改良した新しいアルゴリズムを提案する。また、linear matrix inequality(LMI)設計条件導出には見られないconvex SOS条件を導出時の特異性および欠点について言及し、nonconvex SOS設計条件を用いることでこれらの欠点が回避できることを述べる。従来のpath-followingアルゴリズムは決定変数の数を考慮した局所的convex条件下で反復的に解いていく手法であるが、コスト保証制御では与えられた評価（コスト

VII
ト）関数の上限値を最小化する思想でフィードバックゲイン（とリアプノフ関数）を決定することから，従来のpath-followingアルゴリズムをそのまま適用できない．このため，決定変数の補足を考慮した局所的convex条件下で反復的に解くだけでなく，与えられた評価（コスト）関数の上限値を最小化するアルゴリズムが提案されている．設計例題を通じて，従来から試みられているコスト保証制御のconvex設計条件に比べ，本論文で提案するnonconvex設計条件を用いた新しいpath-followingアルゴリズムの有効性を明らかにする．

第4章では，Hamilton-Jacobi-Bellman (HJB)方程式に基づく制御器構造を反映した新しい制御器を提案し，これに基づくnonconvexSOS設計条件を導出する．さらに，多項式ファジイシステムのメンバーシップ関数（非多項式関数）の多項式表現化およびリアプノフ関数の特性を考慮したS-procedureによるSOS設計条件の緩和を実現する．本論文で提案する緩和手法は，LMIに基づく設計法では実現できないSOSに基づく設計法特有のものであることにも言及する．ベンチマーク設計問題を通じて，提案手法の有効性を検証する．非線形システムを対象にしたコスト保証制御であることから，評価関数値と最小化した評価関数の上限値は必ずしも一致せずにギャップが存在する．このギャップをいかに小さくし，評価関数値を設計段階で見積ることも重要である．本論文では，SOSの特徴を生かし，多項式リアプノフ関数の次数を上げることで，このギャップを小さくすることが，評価関数値を設計段階で精度よく見積ることができることを示す．

第5章では，電動パラグライダー型無人航空機の高度制御に第4章で提案したコスト保証制御を適用した結果を述べる．無人航空機では，通常の剛体ロボットダイナミクスだけでなく，翼が発生する揚力や抗力も考慮しなければならないため，制御が困難な対象である．本論文では，風洞実験データより近似して作成した揚力/抗力特性を表現する多項式モデルを用い，電動パラグライダー型無人航空機の縦方向ダイナミクスを構築する．得られた多項式モデルに対して，第4章で提案した設計法を適用し，高度制御シミュレーションを用いた従来手法との比較を通じて，提案設計法の実システム適用可能性を検討する．

第6章では，結論を述べる．本研究のまとめと問題点，および，今後の展望について述べる．
This thesis presents a guaranteed cost control of nonlinear systems based on polynomial fuzzy control and path-Following algorithm. A nonconvex sum-of-squares (SOS) conditions realizes guaranteed cost control of polynomial fuzzy systems has been achieved in this research. Although the SOS-based design method is regarded as the LMI-based design method, there are still some unsolved problems in system analysis and design. In order to solve the remaining problems of the SOS-based design method, this research proposes new ideas, that is, using the so-called path tracking algorithm to directly solve the non-convex SOS design conditions by guaranteed cost control.

This present thesis comprises six chapters, which are:

Chapter 1 is the introduction which includes research background, motivations, and the position of this research.

Chapter 2 is preliminaries which includes definitions, mathematical tools, and relaxation tools.

Chapter 3 presents a new nonconvex design algorithm for guaranteed cost control of polynomial fuzzy systems, that includes the Takagi-Sugeno (T-S) fuzzy systems as a special case. A new scheme that minimizes the upper-bound of a given performance function while minimizing the parameters which check non-negativity for SOS design conditions is one of the main contribution of this research. The two parameters in the path-following algorithm are minimized by introducing a double-loop structure. In addition, co-positive relaxation is applied to bring relaxation to sum-of-squares conditions. Two complex nonlinear system design examples (a polynomial chaotic system and a complicated nonlinear system) are employed to illustrate the validity and applicability of the proposed nonconvex design algorithm.

In chapter 4, a new type of polynomial fuzzy controller based on an approximate solution for the Hamilton-Jacobi-Bellman (HJB) inequality is introduced. Also, two relaxations are provided by bringing a peculiar benefit of the SOS framework. One is an $S$-procedure...
relaxation for the considered Lyapunov function level set that is contractively invariant set. The other is an $S$-procedure relaxation for design conditions obtained for polynomial membership functions redefined by variable replacements in considered ranges. A benchmark example is applied to illustrate the validity and applicability of the proposed nonconvex design algorithm. And the result is compared with chapter 3 algorithm. Another focus of this chapter is to provide a particular method, that is, lower upper-bound estimation, to estimate the cost value of the design cost function by increasing the order of the polynomial function under consideration. The same benchmark example is applied to present the accuracy of the estimation.

The Chapter 5 present the result of applying the proposed algorithm to a parafoil wing-type unmanned aerial vehicle (UAV) practical system, also the lower upper-bound estimation.

Finally, Chapter 6 summarizes the results and discussions of the previous chapters, as well as the future directions of current research.
I had never imagined that one day I would study for a Ph.D. abroad. And the knowledge I have learned and the help and company that I have received over these years is beyond my imagination.

I am never an excellent person when talking about study. However, my parents never value me with my score. They cultivate my talents and always show their warm support to my decisions. Also, I would like to express my greatest love to my sister. Often a phone call from her saved me from a bad mood. I felt we have never been this close before; even we are far away from each other. Without the support and accompany from my family, I would not be able to accomplish this degree.

Next, I would like to present my great thanks to Prof. Kazuo Tanaka, who guides me and encourages me since I came to UEC as an exchange student. Without his encouragement and warm welcome, I would not have started my Ph.D. course at UEC. Thanks for all the kindly supports and advice that I had received from him to complete this challenge.

Then, I would like to express my gratitude to Tamano san and Junko san for taking good care of me and be there for me like my Japanese parents. They always invite me to their house for good meals, go on trips, share their stories, and give suggestions when I face difficulties in life. This wonderful relationship is the greatest gift that I have received during my stay in Japan.

Also, I would like to take this chance to thanks my senpai, Alissa, Jairo, Chou san, and other labmates. My lovely kouhai, Yang, Chien, Ray, Lee, James, Danial, Nei, Liao, Hanpo, and so on. I had an enjoyable time hanging out with you all. And I would like to thanks Choo sensei, Suzuki sensei, Takahashi san, Okuyama san, Sato san, and Shimizu san for giving excellent chance to learn more about Aikido and Japanese culture. Especially Shimizu san being such a kind and supportive friend. Thanks to all my friends from over the world, Marzei, Jeans, Jens, Clara, Jameesh, AJ, Julio, Edgar, JUSST students, etc. And I would
like to thank Prof. Shi for hiring me as her TA. It was an excellent experience working with her.

Finally, I would like to give my special thanks to Taiwan, where I was born. Thanks to Taiwan for being a safe and warm place where I can rest and refresh myself whenever I need a break. Looking at Taiwan from an overseas perspective stronger my love for Taiwan. And realizing how wonderful and excellent we are. I am proud of being a Taiwanese.

Bless Taiwan and the world.
## Contents

Japanese Abstract .......................................................... VII
Abstract ......................................................................... IX
Acknowledgements .......................................................... XI
List of Figures ................................................................. XVI
List of Tables ................................................................. XVII
List of Notations ............................................................. XX

1 Introduction ................................................................. 1

2 Preliminaries ................................................................. 7
   2.1 Definitions ............................................................. 7
      2.1.1 Positive Definiteness ....................................... 7
      2.1.2 Sum of Squares Decomposition ..................... 8
   2.2 Mathematical Tools ................................................ 8
      2.2.1 Linear Quadratic Regulator Control .............. 8
      2.2.2 Guaranteed Cost Control ............................... 8
      2.2.3 Path-Following Algorithm ............................ 9
      2.2.4 Bisection Searching Technique .................... 10
   2.3 Relaxation Tools .................................................... 11
      2.3.1 Co-positive Relaxation ................................. 11
      2.3.2 S-Procedure ............................................... 12

3 Guaranteed Cost Control System Design Using Nonconvex Conditions 13
   3.1 Polynomial Fuzzy Systems .................................. 14
   3.2 Guaranteed Cost Control System Design ............. 16
   3.3 Guaranteed Cost Control Based on Path-Following Algorithm .... 18
3.3.1 Minimizing Objects Relationship in Iterations 23
3.4 Nonlinear System Examples 23
3.4.1 Design Example I: 3-D Polynomial Chaotic System 24
3.4.2 Design Example II: A Complex Nonlinear System 29
3.4.3 Comparing Feasible Area: Using Design Example II 32

4 The New Polynomial Fuzzy Controller and Lower Upper-Bound Estimation 35
4.1 HJB Inequality and Stabilization Conditions of Polynomial Fuzzy Controller 36
4.2 A New Path-Following Based Design 40
   4.2.1 Benchmark Example 45
4.3 Lower Upper-Bound Estimation: Benchmark Example 49
   4.3.1 Complex Nonlinear System Design Example 53

5 Practical System Design: Unmanned Aerial Vehicles 57
5.1 System Configuration of the Parafoil Wing-Type UAV 57
5.2 Mathematical Model of UAV 59
5.3 Path-Following Based Design for Guaranteed Cost Control for the UAV system 62
5.4 Simulation Results of UAV Model and Comparison 67
5.5 Lower Upper-Bound Estimation: The UAV Model 73
5.6 Results of UAV Lower Upper-Bound Estimation 73

6 Conclusions and Future Work 75
6.1 Conclusions 75
6.2 Future Work 76

References 84
# List of Figures

3.1 The novel path-following algorithm structure. ........................................... 19
3.2 The relationship among $\alpha_2$, $\alpha_3$, and $\lambda$ according to each iteration $N$ .... 23
3.3 Design Example I: Control results of Convex Algorithm (Algorithm 1), Stabilization Algorithm (Algorithm 2), and the Nonconvex Algorithm (Proposed). ................. 27
3.4 Design Example I: The controlled trajectory of Convex Algorithm (Algorithm 1), Stabilization Algorithm (Algorithm 2), and the Nonconvex Algorithm (Proposed). ............. 28
3.5 Design Example II: The behavior of complex nonlinear system with $u=0$. ......... 30
3.6 Design Example II: The control results of proposed Nonconvex Algorithm. .......... 30
3.7 Feasible areas and cost function values $J$ (1)Convex Algorithm, (2) Nonconvex Algorithm. .......................................................... 33

4.1 The new path-following algorithm structure. ............................................... 41
4.2 Benchmark Example: the behavior of complex nonlinear system with $u=0$. ....... 48
4.3 Benchmark Example: the control results of proposed Nonconvex Algorithm. ....... 49
4.4 Control result for the initial state $a = 3$ and $b = 1$. ................................. 50

5.1 UAV platform (paragofoil wing type UAV). ............................................... 57
5.2 System configuration of UAV platform [50]. ............................................. 58
5.3 Variables and parameters in a rigid body dynamics of UAV model. ..................... 59
5.4 The flowchart of the iteration. ................................................................. 63
5.5 Control results with initial state: $q_0 = [-5\ 0\ 0\ 0]^T$. ............................ 69
5.6 Control output with initial state: $q_0 = [-5\ 0\ 0\ 0]^T$. ............................ 70
5.7 Control results with initial state: $q_0 = [5\ 0\ 0\ 0]^T$. ............................ 71
5.8 Control output with initial state: $q_0 = [5\ 0\ 0\ 0]^T$. ............................ 72

6.1 The fly-wing-type UAV. ................................................................. 77
6.2 The vertical take-off and landing-type UAV
# List of Tables

3.1 Cost function values $J$ and the designed upper-bound parameters $\lambda$ values of Convex Algorithm, Stabilization Algorithm, and the Nonconvex Algorithm. 29

3.2 Minimizing upper-bound $\lambda$ and Cost function values $J$ for three different algorithm ($a = 1, b = 0$). 31

4.1 Comparison results of $\lambda$ and $J$ for benchmark example $(a, b) = (3, 1)$ 48

4.2 The minimum $\lambda$ in different order of $V$. 53

5.1 Dimensions and specifications of UAV platform. 58

5.2 Lists of the parafoil wing-type UAV variables. 60

5.3 List of the parafoil wing-type UAV parameters. 60

5.4 Minimum $\lambda$ value of different degrees ($q_0 = [-5 0 0 0]^T, J = 79.93$) 74

5.5 Minimum $\lambda$ value of different degrees ($q_0 = [5 0 0 0]^T, J = 99.12$) 74
**List of Notations**

**List of Acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOS</td>
<td>Sum-of-squares</td>
</tr>
<tr>
<td>T-S fuzzy</td>
<td>Takagi-Sugeno fuzzy</td>
</tr>
<tr>
<td>HJB</td>
<td>Hamilton-Jacobi-Bellman</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned aerial vehicle</td>
</tr>
<tr>
<td>LMI</td>
<td>Linear matrix inequalities</td>
</tr>
<tr>
<td>LUB</td>
<td>Lower upper-bound</td>
</tr>
</tbody>
</table>
INTRODUCTION

In the middle of the 1960s, the idea of fuzzy sets was first introduced by Zadeh [1], which brings a new preliminary idea of the basic properties and implications of classification. Lotif Zadeh first pointed out the ambiguity concept, like human recognition and abstraction, can provide a natural way of dealing with classification problems instead of the classic, true and false, boolean logic-based problems. Later in 1974, Ebrahim Mamdani [2] brought the scheme of fuzzy logic to control a complex, nonlinear dynamic plant. The presented If-Then structures can intuitively describe human operations into a controller.

Since Tomohiro Takagi and Michio Sugeno first introduce Takagi-Sugeno (T-S) models in 1985 [3], T-S models have successfully utilized in a wide range of research, i.e., robotic system stability analysis and stabilization [4,5]. In recent decades, the linear matrix inequality (LMI)-based methods of T-S fuzzy systems have played a central role in fuzzy control research. Within the LMI framework, a lot of research on this topic has devoted much effort to numerically feasible design problems [6–8]. The LMI-based design method has accomplished great success. However, there are still some problems that cannot be resolved by the LMI-based design method.

In the past decade, polynomial fuzzy systems have been discussed and studied [9–20,36,37]. The consequent part of polynomial fuzzy system matrices include polynomials and is observed as a more general representation of the T-S fuzzy model. Due to the system matrices includes polynomials, LMI-based stability and stabilization conditions cannot be employed. Recently, a design method based on the sum-of-squares (SOS) has been successfully developed, which is regarded as a post-design method based on LMI. SOS-based design approaches have been extensively discussed in some studies, e.g., [22,23,44], etc., with numbers of design methods, including guaranteed cost control [20,21]. The studies [9–11] are the pioneering researches using polynomial fuzzy systems and controllers. From these excellent pioneer researchers, the polynomial fuzzy controller can be regarded as the general form of the T-S
fuzzy controller. The same as polynomial fuzzy systems is to T-S fuzzy systems. According to this definition, it is found that the SOS framework is a post-LMI framework [12]. This article proposes a new SOS framework with peculiar advantages beyond the LMI-framework.

The research [9]-[11] are some of the pioneer researches, since then numbers of related SOS-based fuzzy control from different point of view have been published one after another, i.e., robust control [13, 14], time-delay control [18], observer design [15, 16], etc. The research paper [20] proposed a SOS framework guaranteed cost control system, which demonstrated the utility and the benefit of SOS framework beyond LMI framework. However, there are still some difficulties in the existing SOS framework, mentioned in [12, 19, 24], need to be overcome. First of all, the construction of Lyapunov function is severely affected by $B_i(x)$ in polynomial fuzzy systems. Secondly, when a higher-order Lyapunov function is considered it is difficult to theoretically guarantee the global stability of control systems. Finally, the last difficulties, as mention above when implement the typical transformation from the original nonconvex conditions to convex conditions under the SOS framework unlike LMI framework is always equivalent, does not always equivalent under SOS framework. To conquer this dilemma, in [19, 24, 25] a so-called path-following algorithms were introduced to solve this difficulties. In this thesis, a novel structure of path-following is proposed.

This thesis proposes a new approach dealing with guaranteed cost control of polynomial fuzzy systems with nonconvex conditions via a path-following algorithm. To handle SOS nonconvex conditions issue, a typical transformation technique is commonly applied, e.g., [20, 21]; However, the typical transformation in SOS conditions cause conservativeness issues [12, 19]. It is also pointed out in [12, 19] that there are some limitations in polynomial Lyapunov function, which will lead to conservativeness results so that sometimes the global stability cannot be guaranteed. Although these difficulties only appear in SOS conversion cases, not in LMI, and sometimes it is challenging to solve nonconvex conditions in practical, the nonconvex algorithm proposed in this thesis can effectively bypass the above difficulties. As far as we know, there has not been any literature research on the application of path-following design to guaranteed cost control. Also, it should be noted that all the existing path-following techniques [19, 24, 44], etc, cannot be directly applied to guarantee cost control design. The reason is that path-following algorithms in [19, 24, 44] only require to minimize one parameter, the non-negativity checking parameters of the SOS design; on the other hand, this research requires to minimize the upper-bound of the designed cost function and the non-
negativity checking parameters of the SOS design condition in parallel. Therefore, this thesis developed a novel double-loop structure path-following design algorithm scheme.

Later in the content, in Chapter 3, first we proposes a polynomial Lyapunov function for guaranteed cost control design, that the feedback system is consists of a polynomial fuzzy system and a polynomial fuzzy controller. In this chapter copositive relaxation is introduced to give relaxation to SOS conditions. The detail of realizing a two parameters minimization in path-following algorithm scheme is illustrated in Chapter 3.3. Then, we employed a three-dimensional polynomial chaotic system with multiple inputs and a complicated nonlinear system to demonstrate the utility of the proposed algorithm. Besides that, the comparison tables of the cost function of our design algorithm with a convex design algorithm (Algorithm 1) [20] and a path-following stabilization algorithm (Algorithm 2), applied path-following algorithm to solve nonconvex stabilization conditions, are also provided.

Based on the knowledge of Chapter 3, a new type of polynomial fuzzy controllers based on an approximate solution for Hamilton-Jacobi-Bellman (HJB) inequality is introduced in Chapter 4, which reduces the use of the decision parameter. Theoretically, the new polynomial fuzzy controller gives a necessary and sufficient condition for the optimality of control with respect to the cost function. In addition, two S-procedure relaxations are introduced to bring special benefits to the SOS framework. One is implemented in polynomial fuzzy membership function by proposing an improved consideration range. Another is introduced for bringing relaxation for the considered outmost Lyapunov function level set that is contrastively invariant set. Therefore an improved double-loop structure path-following design algorithm based on minimizing the upper-bound of the cost function scheme is proposed. Another highlight of this chapter is that we proposed a particular to estimate the cost function value. Theoretically speaking, it is an indispensable and useful estimating the minimum upper limit without calculating cost function value in the design process. In the end of this chapter, a complex nonlinear system design example is employed to illustrate the effectiveness of the proposed algorithm and the cost value lower upper-bound estimation.

In Chapter 5, we extend the concept from the previous chapters to a practical parafoil wing-type unmanned aerial vehicles (UAVs) model. By applying the controller introduced in Chapter 4, to stabilize the flying height of the UAV at a specified height. Also the simulation results of estimating the lower upper-bound of the cost function $\lambda$ is provided.
Chapter 1 Introduction

This present thesis comprises six chapters, which are:

Chapter 1 is the introduction of research background and objectives.

Chapter 2 is preliminaries which provides definitions as well as mathematical tools and relaxation tools which will be used in later chapters.

Chapter 3 presents a novel nonconvex design algorithm for guaranteed cost control based on optimal polynomial fuzzy control. A new design structure which directly solves nonconvex sum-of-square design conditions for a guaranteed cost control via employ the so-called path-following algorithm is provided. Two complex nonlinear system design examples (a polynomial chaotic system and a complicated nonlinear system) are employed to illustrate the validity and applicability of the proposed nonconvex design algorithm. Besides that, the comparison tables of the cost function of the proposed algorithm with a convex design algorithm [20] and a path-following stability algorithm [9] are also provided.
Chapter 4 present a new algorithm nonconvex design algorithm for guaranteed cost control by employed a new polynomial fuzzy controller based on an approximate solution for the Hamilton-Jacobi-Bellman (HJB) inequality. Also, two relaxations are provided by bringing a peculiar benefit of the SOS framework. One is an S-procedure relaxation for the considered Lyapunov function level set that is contractively invariant set. The other is an S-procedure relaxation for design conditions obtained for polynomial membership functions redefined by variable replacements in considered ranges. Another contribution of this chapter is proposing a reasonable and practical way of estimating the lower upper-bounds of a given cost function by increasing the order of the considered polynomial function.

Chapter 5 gives a practical parafoil wing-type unmanned aerial vehicles model and we applied the algorithm presented in chapter 4 to achieve level flight. This example provides a further proof of the utility of our proposed algorithm. Also, the result of lower upper-bound estimation result is provided.

Chapter 6 summarizes the results and discussions of the previous chapters, as well as the future directions of current research.
Chapter 2 consists of basic definitions, required mathematical tools, and relaxations tools which will be applied in the subsequent chapters.

2.1 Definitions

In this present thesis, bold letters indicate matrices. And scalar, otherwise. The following gives the definitions and explanations of concepts that will be frequently used in the subsequent article.

2.1.1 Positive Definiteness

Consider \( v \in \mathbb{R} - 0 \) and suppose \( f(v) : \mathbb{R}^n \to \mathbb{R} \), where \( V = [v_1, v_2, \ldots, v_n] \). If \( f(v) \) is a positive definite then it is said that for all \( v \) and \( v \neq 0 \) satisfy \( f(v) > 0 \) and \( f(0) = 0 \). On the other hand, it is known as negative definite if satisfy \( -f(v) > 0 \) and \( -f(0) = 0 \).

Consider a quadratic polynomial function:

\[
w(V) = V^T K V, \tag{2.1}
\]

where \( V = [v_1, v_2, \ldots, v_n]^T \in \mathbb{R}^n \) and \( K \) is a \( n \times n \) symmetric matrix. Then it is say that for all \( V \neq 0 \),

\[
\begin{align*}
\text{if} & \quad V^T K V > 0 & K \quad \text{is positive definite.} \\
\text{if} & \quad V^T K V \geq 0 & K \quad \text{is positive semi-definite.} \\
\text{if} & \quad V^T K V < 0 & K \quad \text{is negative definite.} \\
\text{if} & \quad V^T K V \leq 0 & K \quad \text{is negative semi-definite.}
\end{align*}
\]
2.1.2 Sum of Squares Decomposition

The application of the sum of square decomposition and its development in various control problems can be found in [20], [51], [52]. The algorithm presented in this paper relies on the decomposition of the sum of squares of multivariate polynomials. Assume \( f(v(t)) \) is a sum-of-square (SOS) multivariate polynomial, where \( v(t) \in \mathbb{R}^n \), then it is satisfied:

\[
f(v(t)) = \sum_{i=1}^{z} g_i(v(t))^2, \tag{2.2}
\]

where \( g_1(v(t)), g_2(v(t)), \ldots, g_z(v(t)) \) are polynomials. Therefore, \( f(v(t)) \) is always positive if (2.2) is satisfied. With the SOS property it is easier to proof the nonnegativeness of \( f(v(t)) \). The SOSOPT solver [38] is applied to solve the sum of square conditions in this research.

2.2 Mathematical Tools

2.2.1 Linear Quadratic Regulator Control

The linear quadratic regulator control is a optimal control based on state-space representation. LQR uses a performance function (cost function) to find the best gain, which measures the target performance and the energy required to consume the actuator. The performance function is generally represented as:

\[
J = \int_{0}^{\infty} (X^TQX + u^TRu)dt, \tag{2.3}
\]

where \( X \) is the state, \( u \) is the controller, and \( Q \) and \( R \) are weighting matrices. The set performance function is the weighted sum of performance and effort overall time, then by solving the LQR problem, it returns the gain matrix that produces the lowest cost given the dynamics of the system.

2.2.2 Guaranteed Cost Control

The idea of guaranteed cost control is to introduce a upper-bound of a given cost function (performance index) indirectly to guaranteed the cost of the design controller to be less than the boundary.
Assume a general cost function $J$ as:

$$J = \int_0^\infty \dot{y}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \dot{y} \, dt \leq \lambda,$$

where $y$ is output vector. $Q$ and $R$ are positive symmetric weighting matrices. By the introduction of cost function upper boundary $\lambda$, a new strategy of minimizing cost function $J$ can be taken over from minimizing $\lambda$.

### 2.2.3 Path-Following Algorithm

A nonconvex condition is the condition that cross term between more than one decision variables (matrices). Transfer function is typically introduce for solving nonconvex condition; However, by this way may case conservation issue and also sometimes the transfer function is hard to obtain.

Consider the following nonconvex condition, where $\phi_g(x)$ and $\phi_h(x)$ are polynomial matrices and both of them are decision variables, then a simple nonconvex condition can be set as

$$\phi_g(x)\phi_h(x) < 0 \quad (2.4)$$

The problem is to find a solution satisfying (2.4). With a positive definite polynomial matrix $\varphi(x)$, the problem may be converted as

$$\phi_g(x)\phi_h(x) - \alpha \varphi(x) \leq 0 \quad (2.5)$$

If a solution with $\alpha < 0$ can be found it is say that,

”$ - \boldsymbol{v}^T \{ \phi_g(x)\phi_h(x) + \alpha \varphi(x) \} \boldsymbol{v} \text{ is SOS}.” \quad (2.6)"

Where $\boldsymbol{v}$ represent a vector that is independent of $x$. Note that since there exists a cross term of decision variables, $\phi_g(x)\phi_h(x)$ is the bilinear SOS condition. We consider $\delta \phi_g(x)$, $\delta \phi_g(x)$,
Chapter 2 Preliminaries

and \( \delta \phi_g(x) \) are perturbations approaches 0, then we can reasonable approximate that

\[
\phi_g(x) \phi_h(x) \simeq (\phi_g(x) \delta \phi_g(x))(\phi_h(x) \delta \phi_h(x)) \\
= \phi_g(x) \phi_h(x) + \delta \phi_g(x) \phi_h(x) \\
+ \phi_g(x) \delta \phi_h(x) + \delta \phi_g(x) \delta \phi_h(x)
\]

(2.7)

Note that the last term \( \delta \phi_g(x) \delta \phi_h(x) \) is extremely small compare with other terms. From this fact we can transform (2.6) into

\[
\" - \mathbf{v}^T \{ \phi_g(x) \phi_h(x) + \delta \phi_g(x) \phi_h(x) + \phi_g(x) \delta \phi_h(x) \\
+ \alpha \varphi(x) - \alpha \delta \varphi(x) \} \mathbf{v} \text{ is SOS } \"
\]

(2.8)

Note that \( \delta v_g, \delta v_h, \) and \( \delta \varepsilon \) are decision variables in the minimizing optimization. The minimization optimization is iteratively performed by substituting the \( N \)-th solution into the \( N \)-th iteration. According to the iteration law (2.9) decision variables update each iteration so as to minimizing \( \alpha \).

\[
\begin{align*}
\phi_{g}^{N+1}(x) &= \phi_{g}^{N}(x) + \delta \phi_{g}(x) \\
\phi_{h}^{N+1}(x) &= \phi_{h}^{N}(x) + \delta \phi_{h}(x) \\
\varphi^{N+1}(x) &= \varphi^{N}(x) + \delta \varphi(x)
\end{align*}
\]

(2.9)

At the very beginning, the initial setting of the \( \phi_h(x), \phi_g(x), \) and \( \varphi(x) \) is needed. Sometimes the initial value should be carefully select, as a result that some initial setting never lead to \( \alpha < 0 \). If there exist a feasible solution with \( \alpha < 0 \), then that is the solution for the nonconvex condition (2.4).

2.2.4 Bisection Searching Technique

Bisection searching technique is a finding method for continuous functions, which consists of repeatedly bisecting the interval and then selecting the subinterval. Assume the goal is to
find \( f(x) = 0 \), where \( f \) is a continuous polynomial function. And \( x \in [T_{\text{upper}}, T_{\text{lower}}] \) and satisfied:

\[
\begin{align*}
f(T_{\text{upper}}) &> 0; \\
f(T_{\text{lower}}) &< 0.
\end{align*}
\]

Define \( c^N = \frac{T_{\text{upper}}^N + T_{\text{lower}}^N}{2} \) then if

\[
\begin{align*}
f(c^N) &> 0 \quad \text{then} \quad T_{\text{upper}}^{N+1} = C, \\
f(c^N) &< 0 \quad \text{then} \quad T_{\text{lower}}^{N+1} = C,
\end{align*}
\]

(2.10)

where \( N \) is a counter, which adds 1 each time after (2.10). Keep repeating (2.10) until \( T_{\text{upper}}^N - T_{\text{lower}}^N < \epsilon_T \), where \( \epsilon_T \) is a small positive value. Then we can say that the approximate \( f(x) = 0 \) is when \( x = c^N \).

## 2.3 Relaxation Tools

### 2.3.1 Co-positive Relaxation

The matrix \( J = [J_{ij}] \in \mathbb{R}^{\psi \times \psi} \) is copositive, if the following holds:

\[
\beta^T J \beta = \sum_{i=1}^{\psi} \sum_{j=1}^{\psi} \beta_i \beta_j J_{ij} \geq 0,
\]

(2.11)

where \( \beta = [\beta_1, \beta_2, \ldots, \beta_\psi]^T \in \mathbb{R}^\psi \) and \( \beta_i \geq 0 \). To check the copositivity of a matrix, the following technique is used: A relaxation is to introduce \( \hat{\beta}_i = \hat{\beta}_i^2 \) and check whether (2.12) is satisfied or not.

\[
Q^s(\hat{\beta}) = (\sum_{k=1}^{\psi} \hat{\beta}_k^2)^s \sum_{i=1}^{\psi} \sum_{j=1}^{\psi} \hat{\beta}_i^2 \hat{\beta}_j^2 J_{ij} \text{ is SOS},
\]

(2.12)

where \( \hat{\beta} = [\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_\psi]^T \) and \( s \) is a nonnegative integer.
2.3.2 S-Procedure

The S-procedure is a relaxation technique that: as long as some other quadratic forms are negative, some quadratic forms can guaranteed its negativity in LMI approach [26]. It has been proved that SOS decomposition from can brings more benefit by using S-procedure [24, 27].

Given polynomials \( f_1(z(t)) \) and \( f_2(z(t)) \), define sets \( D_1 \) and \( D_2 \):

\[
D_1 := \{ z(t) \in \mathbb{R}^n : f_1(z(t)) \leq 0 \},
\]

\[
D_2 := \{ z(t) \in \mathbb{R}^n : f_2(z(t)) \leq 0 \}.
\]

If there exits a polynomial \( \sigma(z(t)) \geq 0 \) for all \( z(t) \), such that \( -f_1(z(t)) + \sigma(z(t))f_2(z(t)) \geq 0 \) for all \( z(t) \), then \( D_2 \subseteq D_1 \).

Note

For the convenience of explanation, in the following chapters, in some parts the time \( t \) are omitted as for simplifying mathematical expressions. For example, the initial condition \( x(t(0)) \) is presented as \( x_0 \) and initial states of the Lyapunov function \( V(t(0)) \) is presented as \( V_0 \). Also, the time-dependent polynomial functions, such as state-space variables \( x(t) \), control input \( u(t) \), and output \( y(t) \) are denoted as \( x, u, \) and \( y \), respectively.
This chapter addresses a novel nonconvex design algorithm for guaranteed cost control based on fuzzy systems, including T-S fuzzy systems as a special case. A set of nonconvex sum-of-squares conditions is derived from achieving guaranteed cost control of polynomial fuzzy systems. The following article presents the processes of designing the new structure which directly solves nonconvex sum-of-square design conditions for a guaranteed cost control via employ the so-called path-following algorithm. Although the sum-of-squares approach is known as a post-LMI-based design approach, there are still some problems remained to be solved. Aware that there is only one minimizing parameter in the typical path-following approach [19, 24, 44] which is used for checking the non-negativity of SOS-design conditions. However, in our algorithm, after introducing guaranteed cost control, it is required to minimize the upper-bound of a given cost function additionally. Therefore, a novel path-following double-loop structure of minimizing two parameters is realized.

In the following sections, first, the introduction of polynomial fuzzy systems and guaranteed cost control are carried out. Next, Section 3.3 illustrates the main contribution of designing novel path-following algorithm structure step by step into more details. Finally, a 3-D polynomial chaotic system with multiple inputs and a complex nonlinear system examples are employed to demonstrate the utility of the proposed algorithm. Besides that, the comparison of the cost function of our design algorithm with a convex design algorithm [20] (Convex Algorithm) and a path-following stabilization algorithm [9] (Stabilization Algorithm) are also provided in tables.
Consider the following state-space representation nonlinear dynamical system:

\[
\dot{x} = f(x) + g(x)u,
\]

(3.1)

where \(x = [x_1 \ x_2 \ \cdots \ x_n]^T\) and \(u = [u_1 \ u_2 \ \cdots \ u_m]^T\) are vectors of state vector and input, respectively. \(f(x) = [f_1(x) \ \cdots \ f_n(x)]^T\) and \(g(x) = [g_1(x) \ \cdots \ g_n(x)]^T\) are vectors of smooth nonlinear functions. Assume that \(f(x) = 0\) if and only if \(x = 0\). By applying the sector nonlinearity concept [4], the nonlinear system (3.1) can be converted into the following polynomial fuzzy model [9] with lossless of the polynomial fuzzy model conversion.

**Model Rule** \(i\):

If \(z_1\) is \(H_{i1}\) and \(\cdots\) and \(z_\ell\) is \(H_{i\ell}\)

Then \(\dot{x} = A_i(x)\hat{x}(x) + B_i(x)u\)

\(i = 1, 2, \cdots, r\),

(3.2)

where \(z_j\) \((j = 1, 2, \cdots, \ell)\) are premise variables, \(H_{ij}\) denotes the fuzzy set associated with the \(i\)-th model rule and the \(j\)-th premise variable and \(r\) is the number of rules. \(\hat{x}(x)\) is a column vector whose entries are all monomials of \(x\). A monomial in \(x\) is a function of the form \(x_1^{\xi_1}x_2^{\xi_2}\cdots x_n^{\xi_n}\), where \(\xi_1, \xi_2, \cdots, \xi_n\) are nonnegative integers. The consequent part of the polynomial fuzzy model (3.2) is represented by polynomials, incidentally, the consequent part of T-S fuzzy model is not. The polynomial fuzzy model can be described as:

\[
\dot{x} = \sum_{i=1}^{r} h_i(z)\{A_i(x)\hat{x}(x) + B_i(x)u\},
\]

(3.3)

where \(z = [z_1, z_2, \ldots, z_\ell]^T\) are the premise variables and

\[
h_i(z) = \frac{\prod_{j=1}^{\ell} H_{ij}(z_j)}{\sum_{i=1}^{r} \prod_{j=1}^{\ell} H_{kj}(z_j)}.
\]

(3.4)
$h_i(z)$ is the weight of the $i$-th model rule. And all the weight satisfied:

$$h_i(z) \geq 0,$$

$$\sum_{i=1}^{r} h_i(z) = 1 .$$  \hspace{1cm} (3.5)

Using the parallel-distributed compensation framework [6], a fuzzy controller [11] with polynomial rule consequences can be constructed from the given polynomial fuzzy model (3.2) as follows:

**Control Rule** $i$ :

If $z_1$ is $H_i1$ and $\cdots$ and $z_\ell$ is $H_{i\ell}$

Then $u = -F_i(x)\hat{x}(x)$,

$$i = 1, 2, \cdots, r,$$  \hspace{1cm} (3.6)

where $F_i(x)$ are the polynomial feedback gain. Therefore, the fuzzy controller can be presented as:

$$u = -\sum_{i=1}^{r} h_i(z)F_i(x)\hat{x}(x).$$  \hspace{1cm} (3.7)

By substituting the fuzzy controller (3.7) into the polynomial fuzzy model (3.3), the overall closed-loop system can be represented as:

$$\dot{x} = \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(z)h_j(z)\{A_i(x) - B_i(x)F_j(x)\}\hat{x}(x).$$  \hspace{1cm} (3.8)

Finally, the time derivative of Lyapunov function $V(x)$ can be presented as follows after substituting (3.8).

$$\dot{V}(x) = \frac{\partial V(x)}{\partial x} \dot{x} = \frac{\partial V(x)}{\partial x} \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(z)h_j(z)\{A_i(x) - B_i(x)F_j(x)\}\hat{x}.$$  \hspace{1cm} (3.9)
3.2 Guaranteed Cost Control System Design

From here, the time \( t \) will be omitted for simplifying mathematical expressions. The input equation and output equation of a nonlinear dynamical system are redescribed as:

\[
\dot{x} = \sum_{i=1}^{r} h_i(z)\{A_i(x)\dot{x}(x) + B_i(x)u\},
\]

\[
y = \sum_{i=1}^{r} h_i(z)C_i(x)\dot{x}.
\]

(3.10)

where

\[
u = -\sum_{i=1}^{r} h_i(z)F_i(x)\dot{x}.
\]

(3.11)

Then consider the polynomial Lyapunov function candidate \( V(x) \), a positive definite polynomial. If the time derivative of \( V(x) \) along the feedback system trajectory is satisfied, the feedback system is stable.

\[
\dot{V}(x) \leq -\dot{x}^T L(x)\dot{x},
\]

(3.12)

where

\[
\dot{x}^T L(x)\dot{x} = \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(z)h_j(z)M_{ij}(x)
\]

\[
= \tilde{y}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \tilde{y} > 0,
\]

(3.13)

and

\[
\tilde{y} = \begin{bmatrix} y \\ u \end{bmatrix} = \sum_{i=1}^{r} h_i(z) \begin{bmatrix} C_i(x) \\ -F_i(x) \end{bmatrix} \dot{x},
\]

(3.14)

where \( Q \) and \( R \) are symmetry positive definite matrices. Thus, \( L(x) \) is a positive definite matrix. Therefore, we can say that if (3.12) holds, then \( \dot{V}(x) \leq 0 \) is warranted; moreover, the stability of the closed-loop system is guaranteed.
Remark 1. It has been stated in [19], there are some remain issues to be solved while handling SOS-based design for polynomial fuzzy systems. In [9, 11, 20] applied typical transformation to deal with non-convex SOS design conditions. The typical congruence transformation is a mathematical technique that first derived nonconvex conditions into convex conditions then solve the convex conditions instead. The conservativeness occurs by transformation does not exist in LMI conversion cases; however, in SOS design case occurs. Also it is difficult to guaranteed system global stability due to $B_i(x)$ matrices restriction. The above-mentioned issue can be conquered by introducing a path-following algorithm [19]. Even though directly solving nonconvex condition are challenging, but the following proposed processes demonstrate a reasonable and reliable way to overcome.

Assumed the cost function $J$ as follows:

$$J = \int_0^\infty \hat{y}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \hat{y} \, dt. \tag{3.15}$$

Then (3.12) can be adapted as:

$$\hat{y}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \hat{y} < -\dot{V}(x). \tag{3.16}$$

(3.17) is generated by integrating (3.16) $t = [0 \infty]$

$$J = \int_0^\infty \hat{y}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \dot{\hat{y}} \, dt < - \int_0^\infty \dot{V}(x) \, dt. \tag{3.17}$$

If (3.16) holds, is reasonable to assume that the system is stable and $x$ tends to 0 when $t$ goes to infinite. Therefore, (3.17) can be rewritten as

$$J < V(x(0)). \tag{3.18}$$

By introducing $\lambda$ for minimizing the the upper-bound of the designed cost function, (3.19)
can be hold.

\[ J \leq V(x(0)) \leq \lambda. \]  \hspace{1cm} (3.19)

Thus, by minimizing \( \lambda \) as much as possible the cost value \( J \) can be minimized. Finally, recall (3.8), an SOS-based design of guaranteed cost controller can be realized with two decision variables (polynomials) \( V(x) \) and \( F_j(x) \).

\[
\min_{V(x), F_i(x)} \lambda \quad \text{subject to (3.20) - (3.22)}
\]

\[
V(x) - \epsilon(x) \quad \text{is SOS.} \hspace{1cm} (3.20)
\]

\[
-V(x(0)) + \lambda \quad \text{is SOS.} \hspace{1cm} (3.21)
\]

\[
- \left( \sum_{k=1}^{r} \hat{h}_k^2 \right) \sum_{i=1}^{r} \sum_{j=1}^{r} \hat{h}_i^2 \hat{h}_j^2 (A_{ij}(x) + M_{ij}(x)) \quad \text{is SOS,} \hspace{1cm} (3.22)
\]

where

\[
A_{ij}(x) = \frac{\partial V(x)}{\partial x} \{ A_i(x) - B_i(x)F_j(x) \} \hat{x}, \hspace{1cm} (3.23)
\]

\[
M_{ij}(x) = \hat{x}^T \begin{bmatrix} C_i(x) & \theta_0 \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} C_j(x) \\ -F_j(x) \end{bmatrix} \hat{x}. \hspace{1cm} (3.24)
\]

\( \epsilon(x) \) is a given small positive-definite polynomial. (3.22) is a nonconvex condition, since there are cross terms between decision variables (polynomials), which are \( V(x) \) and \( F_j(x) \).

By minimizing the upper-bound \( \lambda \) as much as possible, the optimal solution can be obtained.

### 3.3 Guaranteed Cost Control Based on Path-Following Algorithm

This section illustrates details into steps of how the designing novel path-following algorithm acts, and the relation between parameters will also explain in the follow-up content.

The following algorithm scheme is different from other proposed path-following algorithms [19,24,44], which present a single-loop so as to minimizing \( \alpha \) for checking the non-negativity of SOS conditions. This algorithm assembles an additional loop structure to minimize the upper-bound of a given cost function, \( \lambda \). Therefore, a novel double-loop structure,
as shown in Fig.3.1 has been carried out.

Figure 3.1: The novel path-following algorithm structure.
[Iteration Process]

Step 1: Set a iteration counter $N = 0$. The initial setting of $F_{j0}(x)$ is a feasible solution selected from the guaranteed cost control of polynomial fuzzy system with convex conditions [20]. Bisection searching technique is introduced and expected to accelerate the search speed of finding minimum $\lambda$. Thus, we set an reasonable searching area between $[\lambda_{\text{low}} \lambda_{\text{up}}]$, where $\lambda_{\text{up}} > \lambda_{\text{low}} > 0$ and $\lambda_0 = \lambda_{\text{up}}$.

Remark 2. Bisection searching technique is employed in the following steps, where Section 2.2.4 gives clear explanation of its concept. The setting of the $\lambda_{\text{up}}$ can be roughly defined at the beginning by trial-and-error, or a selection which is greater than the cost function of [20] can also be considered. Note that a larger setting of $\lambda_{\text{up}}$ can bring out more relax constrain in Step 2 in the beginning. It is still need to keep in mind that a bigger $\lambda_{\text{up}}$ setting may takes longer time to converge.

Step 2: With the parameter given by previous steps, optimize the following SOS constraint group I. The bisection searching technique is also employed for minimizing $\alpha_2$ to accelerate the searching speed in the range of $[\alpha_{\text{max}}, \alpha_{\text{min}}]$.

\[
\text{[Constrain Group I]: } \min_{V_N(x)} \alpha_2 \quad \text{subject to } (3.25) - (3.27) \\
V_N(x) - \epsilon(x) \quad \text{is SOS, } (3.25) \\
-V_N(x(0)) + \lambda_N \quad \text{is SOS, } (3.26) \\
- \left( \sum_{k=1}^{r} \hat{h}_k^2(x) \right)^s \sum_{i=1}^{r} \sum_{j=1}^{r} \hat{h}_i^2(x)\hat{h}_j^2(x) \{ \Lambda_{ijN}(x) + M_{ijN}(x) - \alpha_2 V_N(x) \} \quad \text{is SOS, } (3.27)
\]

where

\[
\Lambda_{ijN}(x) = \frac{\partial V_N(x)}{\partial x} \{ A_i(x) - B_i(x)F_{jN}(x) \} \dot{x}.
\]

If there is no any feasible solution that satisfy (3.25) - (3.27) with $F_{j0}(x)$ or $F_{jN}(x)$ even when $\alpha_2 = \alpha_{\text{up}}$. If so, go to Step 5. If a feasible solution with minimum $\alpha_2$ can be found, substitute the solution $V_N(x)$ back to (3.25) - (3.27) to double-check the feasibility using Matlab issos command.

It is highly recommended to double-checking the feasibility of the solution since in some cases the feasible solution given by sosopt command in Matlab environment might be infeasible.
sible solution. If that is the case, slightly increase $\alpha_2$ then check the constrain again. If a solution with $\alpha_2 < 0$ passed double-checking is found in this step, go to Step 5. Other case with $\alpha_2 > 0$, please go to Step 3.

**Step 3:** Applying $F_{jN}(x)$ and $V_N(x)$ obtained in Step 2, solve the following constrain group II. Bisection searching technique is also employed in this step to accelerate the searching speed of minimum $\alpha_3$.

\[ [\text{Constrain Group II}]: \min_{\delta F_j(x), \delta V(x)} \alpha_3 \quad \text{subject to (3.28) - (3.32)} \]

\[ V_N(x) + \delta V(x) - \epsilon(x) \text{ is SOS. (3.28)} \]

\[ -(V_N(x(0)) + \delta V(x(0)) + \lambda_N \text{ is SOS. (3.29)} \]

\[ \begin{aligned} & -\alpha_3(V_N(x) + \delta V(x)) \text{ is SOS. (3.30)} \\ & v_1^T \begin{bmatrix} \epsilon_v V_N^2(x) & \delta V(x) \\ \delta V(x) & I \end{bmatrix} v_1 \text{ is SOS. (3.31)} \\ & v_2^T \begin{bmatrix} \epsilon_F F_j N(x) F_{jN}^T(x) & \delta F_j(x) \\ \delta F_j^T(x) & I \end{bmatrix} v_2 \text{ is SOS. (3.32)} \end{aligned} \]

where

\[ \Lambda_{ijN}(x) = \frac{\partial V_N(x)}{\partial x} (A_i(x) - B_i(x) F_{jN}(x)) \hat{x}, \]

\[ \delta \Lambda_{ijN}(x) = \frac{\partial \delta V(x)}{\partial x} (A_i(x) - B_i(x) F_{jN}(x)) \hat{x} \]

\[- \frac{\partial V_N(x)}{\partial x} B_i(x) \delta F_j(x) \hat{x}, \]

\[ \delta M_{ijN}(x) = \hat{x}^T (F_{iN}^T(x) R \delta F_j(x) + \delta F_i^T(x) R F_{jN}(x)) \hat{x}. \]

$v_1$ and $v_2$ are vectors independent of $x$. And $\epsilon_v$ and $\epsilon_F$ are very small positive values to ensure that $\delta V$ and $\delta F_j(x)$ are small. In step 3, it is also recommended to apply double-checking. If the checking result is infeasible, then slightly increase $\alpha_3$ and recheck the requirements. After the minimum $\alpha_3$ value after double-checking is found, go to Step 4.

**Step 4:** Update $F_{jN}(x)$ with the solution solved by the SOS optimization problem.
Chapter 3 Guaranteed Cost Control System Design Using Nonconvex Conditions

(3.28)-(3.32), following iteration law:

\[
F_{jN+1}(x) = F_{jN}(x) + \delta F_{j}(x).
\] (3.33)

Set \( N = N + 1 \). With the updated \( F_{jN}(x) \) move to \textbf{Step 2}.

\textbf{Step 5:} This step presents the operation detail of minimizing \( \lambda \) by utilizing bisection searching technique. First, define \( F_{jN+1} = F_{jN} \). Next, determine the reason of entering Step 5.

\[
\begin{align*}
\lambda_{up} &= \lambda_N & \text{if reach a feasible solution with } \alpha_2 < 0 \text{ in Step 2}.\\
\lambda_{low} &= \lambda_N & \text{if no feasible solutions in Step 2}.
\end{align*}
\]

Then set \( \lambda_{N+1} = (\lambda_{up} + \lambda_{low})/2 \). If the \( 3.34 \) is satisfied, end the iteration.

\[
\lambda_{up} - \lambda_{low} < \epsilon_{\lambda},
\] (3.34)

where \( \epsilon_{\lambda} \) is a small positive value. If (3.34) is not satisfied, set \( N = N + 1 \) and move on to \textbf{Step 2}.

\textbf{Remark 3.} All design conditions expressed in the proposed design algorithm can be described by SOS, and can be solved symbolically and numerically through the developed MATLAB software [38,39] and semi-definite program (SDP) solver [40,41]. The SOS solver has some numerical reliability options, and all the SOS solutions shown in this article have been carefully provided. It is worth noting that the feasible results may vary slightly depending on the options, especially for SOS conditions with higher-order polynomials. For example, the feasibility of the solver could be chose among 'off', 'fast', 'full', and 'both'. The solver define 'fast' options by default [38]. And we always select 'both' option in this research, which brings the most carefully checking and provide the most reliable solutions. After the feasible solution is obtained in the algorithm, the so-called SOS test, the 'issos' command in the SOSOPT tool, is performed. By substituting the feasible solution obtained by 'both' into the considered SOS conditions, and execute 'issos'. In this double-checking processes we also select the most reliable checking 'both' option.
3.3.1 Minimizing Objects Relationship in Iterations

The relationship among $\alpha_2$, $\alpha_3$, and $\lambda$ during iterations is shown in Fig. 3.2 by exercising the example described in section 3.4.3. In order to present the relationship clearly, the bisection searching technique for $\lambda$ is omitted and changed to decrease progressively. When $\lambda > 130$ each time in Step 5 $\lambda_{N+1} = \lambda_N - 10$; Otherwise, $\lambda_{N+1} = \lambda_N - 1$. During $N = 1$ to 20, $\alpha_2$ value exaggeratedly changes, $\alpha_3$ also has some relatively big changes compared to $N > 20$. This behavior shows the effectiveness of the algorithm for updating $V_N(x)$ and $F_{jN}(x)$, which guides to obtain the minimum $\lambda$.

Figure 3.2: The relationship among $\alpha_2$, $\alpha_3$, and $\lambda$ according to each iteration $N$

3.4 Nonlinear System Examples

To illustrate the utility of the proposed design algorithm (Nonconvex Algorithm), this section provided two design examples and compared with a convex design algorithm [20]
Chapter 3  Guaranteed Cost Control System Design Using Nonconvex Conditions

(Convex Algorithm) and a path-following stabilization algorithm [9] (Stabilization Algorithm).
The first example is a 3-D polynomial chaotic system and another example is a complex nonlinear system, which has been widely applied in the literature [19], [20], etc. Under the guaranteed cost control framework, we compared a nonconvex design and convex design algorithm by comparing the proposed algorithm and algorithm presented in [20], which is named as Convex Algorithm in the following content. In addition, we compared our algorithm with Stabilization Algorithm, a nonconvex stabilization design algorithm using the path-following algorithm, to illustrate the introducing of guaranteed cost control contributes a significant reduction of the cost function value.

Table 3.1 and Table 3.2 present the comparison of the $\lambda$ and the cost function value $J$. The smaller $J$ value represents to lower cost, therefore the lower the better. Note that since Stabilization Algorithm is only a stabilization control, thus $\lambda$ does not exist.

3.4.1 Design Example I: 3-D Polynomial Chaotic System

The employed 3-D polynomial chaotic system with multiple inputs is used as a design example in [43], which considered its T-S fuzzy model (3.35) as follows:

$$\dot{x} = \sum_{i=1}^{2} h_i(z)\{A_i x + B_i u\}, \quad (3.35)$$

where $x = [x_1 \ x_2 \ x_3]^T$, $z = x_2$

$$A_1 = \begin{bmatrix} -2 & -5.78 & 7.89 \\ 25.89 & 7.78 & 8 \\ -15.78 & -7.89 & -2 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} -2 & 35.48 & -12.74 \\ 5.26 & -33.48 & 8 \\ 25.48 & 12.74 & -2 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 1 & -1 & -1 \\ 0 & 2 & -1 \\ -1 & 0 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} -1 & 0 & 1 \\ 1 & 1 & 0 \\ -2 & 0 & -1 \end{bmatrix},$$
Section 3.4 Nonlinear System Examples

\[
C_1 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}, \quad C_2 = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix},
\]

where the membership functions are designed as:

\[
h_1(z) = \frac{12.74 - x_2}{12.74 + 7.89}, \quad h_2(z) = \frac{x_2 + 7.89}{12.74 + 7.89}.
\]

The T-S fuzzy output model is given as:

\[
y = \sum_{i=1}^{2} h_i(z)C_i \cdot x,
\]

The fuzzy controller is given as

\[
u = \sum_{i=1}^{2} h_i(z)F_i \cdot x.
\]

The algorithm required parameter settings are given as follows: \([\alpha_{\text{max}}, \alpha_{\text{min}}] = [5000, -0.1], s = 1, x(0) = [1 \ 1 \ 1]^T, \epsilon(x) = 10^{-6}x^T x, Q = I, and R = I.\]

The solution of Nonconvex Algorithm:

\[
V = 34.4195x_1^2 + 54.5942x_1x_2 - 5.2714x_1x_3 + 23.7926x_2^2 - 8.6811x_2x_3 + 13.7457x_3^2
\]

\[
F1 = [3.0904, 0.26655, -5.8908, 3.939, 7.541, -1.0232, -14.7078, -6.8207, -3.5064];
\]

\[
F2 = [-4.4475, 0.77053, -6.0544, 34.9686, 24.7659, -6.2966, -4.7449, -2.4448, -2.6847];
\]

25
Chapter 3  Guaranteed Cost Control System Design Using Nonconvex Conditions

The solution of **Convex Algorithm**:

\[
V = 38.4049x_1^2 + 60.0138x_1x_2 - 11.8557x_1x_3 + 28.4095x_2^2 - 14.3682x_2x_3 + 18.0107x_3^2;
\]

\[
F_1 = [-1.3894, -2.679, -6.8651]
\]
\[
\quad 30.6911, 28.333, -11.0139
\]
\[
\quad -6.797, 0.3214, -6.3621];
\]

\[
F_2 = [3.0351, 2.313, -13.0889]
\]
\[
\quad 27.2435, 21.98, -7.3085
\]
\[
\quad -4.2462, 3.2164, -9.533];
\]

The solution of **Stabilization Algorithm**:

\[
V = 0.82646x_1^2 + 0.38904x_1x_2 - 0.069731x_1x_3 + 0.32843x_2^2 - 0.089574x_2x_3 + 1.224x_3^2
\]

\[
F_1 = [-2.3553, -7.2305, -30.7444]
\]
\[
\quad 4.2266, 46.6456, -9.994
\]
\[
\quad -35.4248, -15.1983, 1.7893];
\]

\[
F_2 = [-6.1751, -2.0285, -23.6906]
\]
\[
\quad 62.6109, 47.3718, 1.7118
\]
\[
\quad -4.1042, -5.6773, -11.4633];
\]

Fig. 3.3 and Fig. 3.4 show the control results and control trajectories of three different algorithms for a 3-D polynomial chaotic system. Table 3.1 lists the cost function values \( J \) and the cost function upper-bound \( \lambda \) of three different algorithms. The upper-bound of cost function \( \lambda \) does not exist in **Stabilization Algorithm** [9] since the algorithm only deals with stabilization. Comparison result gives evidence of the **Nonconvex Algorithm** proposed in this chapter obtained smaller \( J \) value than **Convex Algorithm**, the existing guaranteed cost control of convex design [20]. In addition, the comparison between **Stabilization Algorithm** and our algorithm gives evidence that when solving nonconvex design conditions, using guaranteed cost control under the path-following framework can significantly reduce the value of \( J \).
Section 3.4 Nonlinear System Examples

Figure 3.3: Design Example I: Control results of Convex Algorithm (Algorithm 1), Stabilization Algorithm (Algorithm 2), and the Nonconvex Algorithm (Proposed).
Figure 3.4: Design Example I: The controlled trajectory of Convex Algorithm (Algorithm 1), Stabilization Algorithm (Algorithm 2), and the Nonconvex Algorithm (Proposed).
Table 3.1: Cost function values $J$ and the designed upper-bound parameters $\lambda$ values of Convex Algorithm, Stabilization Algorithm, and the Nonconvex Algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\lambda$</th>
<th>$J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex Algorithm</td>
<td>120.0</td>
<td>106.4</td>
</tr>
<tr>
<td>Stabilization Algorithm</td>
<td>-</td>
<td>100.4</td>
</tr>
<tr>
<td>Nonconvex Algorithm (Chap. 3)</td>
<td>112.6</td>
<td>75.5</td>
</tr>
</tbody>
</table>

3.4.2 Design Example II: A Complex Nonlinear System

Assume the following polynomial fuzzy model:

$$\dot{x} = \sum_{i=1}^{2} h_i(z) \{A_i(x)\dot{x} + B_i(x)u\},$$

where $x = \dot{x} = [x_1 \ x_2]^T$, $z = x_1$ and $A_i(x)$ and $B_i(x)$ matrices are given as

$$A_1(x) = \begin{bmatrix}
-1 + x_1 + x_1^2 + x_1x_2 - x_2^2 & 1 \\
-a & -1
\end{bmatrix},$$

$$A_2(x) = \begin{bmatrix}
-1 + x_1 + x_1^2 + x_1x_2 - x_2^2 & 1 \\
0.2172a & -1
\end{bmatrix},$$

$$B_1(x) = \begin{bmatrix} x_1 \\ b \end{bmatrix}, \quad B_2(x) = \begin{bmatrix} x_1 \\ b \end{bmatrix},$$

$$C_1 = C_2 = I$$

where $a$ and $b$ are constant values. The membership functions are defined as follows:

$$h_1(z) = \frac{\sin x_1 + 0.2172x_1}{1.2172x_1}, \quad h_2(z) = \frac{x_1 - \sin x_1}{1.2172x_1}.$$

Fig. 3.5 demonstrates the control system behavior without control, i.e., $u = 0$, and Fig. 3.6 demonstrates the control results of the complex nonlinear system applying the proposed algorithm. The polynomial fuzzy controller is designed as

$$u = \sum_{i=1}^{2} h_i(z)F_i(x)\dot{x}. \quad (3.39)$$

The polynomial fuzzy model is reduced to the benchmark design example used in [19] and [20] with $a = 1$ and $b = 0$. The algorithm required parameter settings are designed as follows:
Chapter 3  Guaranteed Cost Control System Design Using Nonconvex Conditions

Figure 3.5: Design Example II: The behavior of complex nonlinear system with $u=0$.

Figure 3.6: Design Example II: The control results of proposed Nonconvex Algorithm.
\([\alpha_{\text{max}}, \alpha_{\text{min}}] = [5000, -0.1] \), \(s = 1\), \(x(0) = [10 \ 10]^T\), \(\epsilon(x) = 10^{-6}\hat{x}^T\hat{x}\), \(Q = I\), and \(R = I\).

Table 3.2 presents a clear comparison of the design upper-bound parameters \(\lambda\) and the cost function values \(J\) at \(a = 1\) and \(b = 0\).

Table 3.2: Minimizing upper-bound \(\lambda\) and Cost function values \(J\) for three different algorithm \((a = 1,b = 0)\).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>(\lambda)</th>
<th>(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex Algorithm</td>
<td>634.2</td>
<td>253.83</td>
</tr>
<tr>
<td>Stabilization Algorithm</td>
<td>-</td>
<td>591.44</td>
</tr>
<tr>
<td>Nonconvex Algorithm (Chap. 3)</td>
<td>661.7</td>
<td>172.77</td>
</tr>
</tbody>
</table>

The solution of the proposed Nonconvex Algorithm:

\[
V = 3.9854 \ x_1^2 + 2.6313 \ x_2^2 \\
F_1 = [1.4818 \ x_1 + 1.155, \ 2.1174 \ x_1 + 0.3713] \\
F_2 = [2.0176 \ x_1 + 2.1544, \ -0.69145 \ x_1 - 0.79865].
\]

The solution of Convex Algorithm:

\[
V = 3.941 \ x_1^2 + 2.401 \ x_2^2 \\
F_1 = [3.566 \ x_1 + 0.1153 \ x_2 + 0.017402, \ -0.11118] \\
F_2 = [3.7885 \ x_1 + 0.13028 \ x_2 - 0.0346, \ 0.10708].
\]

The solution of Stabilization Algorithm:

\[
V = 5 \ x_1^2 + 5 \ x_2^2 \\
F_1 = [2.0898 \ x_1 + 0.43787 \ x_2 + 0.83347, \ 0.43787 \ x_1 + 0.3249 \ x_2 + 0.019445] \\
F_2 = [2.0893 \ x_1 + 0.43849 \ x_2 + 0.83852, \ 0.43849 \ x_1 + 0.32492 \ x_2 + 0.090347].
\]

The Table 3.2 shows that the proposed Nonconvex Algorithm gives much smaller \(J\) value than the Convex Algorithm and Stabilization Algorithm, which illustrates the practicality of the Nonconvex Algorithm (Chap. 3).
3.4.3 Comparing Feasible Area: Using Design Example II

This subsection presents the feasible areas of the proposed Nonconvex Algorithm and compared with Convex Algorithm. Figure 3.7 is a bar graph that presented the feasible areas (x−y axis represents different a and b settings) and individual cost function value J are express by the height of each bars (z-axis) within the area of $a \in [3, 4, ..., 11]$ and $b \in [0, 1, ..., 10]$.

From the figure we can claim that the proposed Nonconvex Algorithm obtained more relax result than Convex Algorithm, which brings out a much wider feasible area. Moreover, comparing J values under the same a and b settings where both algorithms are feasible, the proposed Nonconvex Algorithm obtains significantly smaller values than Convex Algorithm.

Note that, for every set of a and b, the initial $F_0(x)$ for the proposed Nonconvex Algorithm is the solution $F(x)$ obtained by Convex Algorithm at $a = 1$ and $b = 0$. Therefore, the practicality of the proposed Nonconvex Algorithm through this example can be observed.
Figure 3.7: Feasible areas and cost function values $J$ (1) Convex Algorithm, (2) Nonconvex Algorithm.
The New Polynomial Fuzzy Controller and Lower Upper-Bound Estimation

This chapter further enhances the algorithm proposed in the previous chapter. Based on the approximate solution of the Hamilton-Jacobi-Bellman (HJB) inequality and a set of SOS design conditions, the new proposed algorithm gives a new polynomial fuzzy controller to achieve guaranteed cost control. In addition, two \( S \)-procedure relaxations were introduced. One is an \( S \)-procedure relaxation for the considered Lyapunov function level set that is contractively invariant set. The other is an \( S \)-procedure relaxation for design conditions obtained for polynomial membership functions redefined by variable replacements in considered ranges.

This chapter presents the processes of designing the new polynomial fuzzy controller and introduce \( S \)-procedure relaxations, then introduces the new double-loop iteration structure which directly solves nonconvex sum-of-square design conditions for a guaranteed cost control via employ the so-called path-following algorithm. Finally, to illustrate the effectiveness and the improvement of the new proposed algorithm, the new Nonconvex Algorithm is compared with the Convex Algorithm [20] and the Nonconvex Algorithm proposed in Chapter 3.

Another focus of this chapter is to provide a particular method, that is, lower upper-bound estimation, to estimate the cost value of the design cost function by increasing the order of the polynomial function under consideration. The same benchmark example is applied to present the accuracy of the estimation.
4.1 HJB Inequality and Stabilization Conditions of Polynomial Fuzzy Controller

This section gives a novel type of polynomial fuzzy controllers based on an approximate solution for the Hamilton-Jacobi-Bellman inequality and a set of SOS design conditions to realize the guaranteed cost control for (3.10). The cost function is designed as follows:

\[
J = \int_{0}^{\infty} (y^T Q y + u^T R u) \, dt,
\]

where \(Q\) and \(R\) are positive definite symmetric matrices. The polynomial fuzzy controller based on the approximate solution of the Hamilton-Jacobi-Bellman (HJB) inequality is designed as follows:

\[
u = -\frac{1}{2} R^{-1} \left( \sum_{i=1}^{r} h_i(z) B_i(x) \right)^T \left( \frac{\partial V(x)}{\partial x} \right)^T,
\]

where \(V(x)\) is a Lyapunov function of input and output equation (4.1). Since the polynomial fuzzy controller (4.2) relates to a solution of the Hamilton-Jacobi-Bellman inequality of nonlinear systems, the new defined controller (4.2) is expected to provides a lower cost of the considered cost function compared with the previous polynomial fuzzy controller [10, 11]:

\[
u = - \sum_{i=1}^{r} h_i(z) F_i(x) \hat{x}(x).
\]

\(F_i(x)\) is also a decision variable (polynomial), if the polynomial fuzzy controller (4.3) is used, like in Chapter 3. Therefore, one of the benefit of introducing (4.2) is that the decision variable (polynomial) can be reduced in the guaranteed cost controller design. From the perspective of computational complexity, the reduction in the number of decision variables brings great advantages.

By substituting (4.2) into (4.1), the entire closed loop system can be obtained, as shown below:

\[
\dot{x} = \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(z) h_j(z) \left( A_i(x) \hat{x}(x) \right)
\]

\[- \frac{1}{2} B_i(x) R^{-1} B_j^T(x) \left( \frac{\partial V(x)}{\partial x} \right)^T.
\]

36
Theorem 4.1. \( V(x) \) becomes a Lyapunov function, if there exists a polynomial \( V(x), \beta_{ij}(x), \) and \( \theta_p(x, \eta) \) satisfied constrains \((4.5)-(4.9)\) with scalar \( \alpha < 0 \), which proves the non-negativity of the constrain. Thus, the polynomial fuzzy controller \((4.3)\) stabilizes the system \((4.1)\) and it is satisfied that \( J \leq V(x(0)) \leq \lambda \), where \( V(x) \) becomes a Lyapunov function.

\[
V(x) - \epsilon V(x) \text{ is SOS,} \\
- V(x(0)) + \lambda \text{ is SOS,} \\
\beta_{ij}(x) \text{ is SOS, } i, j = 1, 2, \ldots, r, \\
\theta_p(x, \eta) \text{ is SOS, } p = 1, 2, \ldots, \xi, \\
- \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(\eta) h_j(\eta) \left( \frac{\partial V(x)}{\partial x} A_i(x) \hat{x} - \frac{1}{4} \frac{\partial V(x)}{\partial x} S_{ij}(x) \left( \frac{\partial V(x)}{\partial x} \right)^T \hat{x}^T C_i(x)^T Q C_j(x) \hat{x} + \beta_{ij}(x) \{ V(x(0)) - V(x) \} - \alpha V(x) \right) \\
+ \sum_{p=1}^{\xi} \theta_p(x, \eta) (\eta_p - \eta_p^{\text{min}})(\eta_p - \eta_p^{\text{max}}) \text{ is SOS,} \tag{4.9}
\]

where

\[
S_{ij}(x) = B_i(x) R^{-1} B_j^T(x). \tag{4.10}
\]

\( \beta_{ij}(x) \) are positive polynomials with \( \beta_{ij}(0) = 0 \). \( \theta_p(x, \eta) \) are positive polynomials with \( \theta_p(0, \eta) = 0 \). \( \epsilon V(x) \) is a radially unbounded positive-definite polynomial. From the fact that \( \lambda \) is the upper-bound of \( J \), by minimizing \( \lambda \) as much as possible, we can design the guaranteed cost controller. \( \lambda \) is the design upper-bound of the cost function \( J \), therefore a guaranteed cost controller is designed by minimizing the upper-bound as much as possible.

\textbf{Proof:} Consider a polynomial Lyapunov function candidate \( V(x) \). If \((4.5)\) is satisfied, then \( V(x) > 0 \) at \( x \neq 0 \).
The time derivative of $V(x)$ can be represented as

$$
\dot{V}(x) = \frac{\partial V(x)}{\partial x} \dot{x} = \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(z) h_j(z) \left( \frac{\partial V(x)}{\partial x} A_i(x) \dot{x} - \frac{1}{2} \frac{\partial V(x)}{\partial x} S_{ij}(x) \left( \frac{\partial V(x)}{\partial x} \right)^T \right). 
$$

(4.11)

The aim of this theorem is to derive SOS conditions which minimizes the upper-bound of cost function (4.1). A common used guaranteed cost control is considered as:

$$
\dot{V}(x) \leq -y^T Q y - u^T R u, 
$$

(4.12)

where $V(x)$ is a Lyapunov function which satisfies $J \leq V(x(0))$. And $\lambda$ as the upper-bound of cost function is introduced such that $J \leq V(x(0)) \leq \lambda$. By minimizing $\lambda$ as much as possible, guaranteed cost controller (4.2) can be generated. A set of SOS conditions which satisfying $\dot{V}(x) \leq -y^T Q y - u^T R u$ only if $V(x) - V(x(0)) \leq 0$ is generated, which means that it is not always required satisfying $\dot{V}(x) \leq -y^T Q y - u^T R u$ when $V(x) - V(x(0)) < 0$.

By introducing $S$-procedure, Section 2.3.2, the following relaxation can be realized.

$$
-(\dot{V}(x) + y^T Q y + u^T R u) + \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(z) h_j(z) \beta_{ij}(x) \{V(x) - V(x(0))\} \geq 0 , 
$$

(4.13)

where $\beta_{ij}(x) > 0$. Note that it is satisfied that $\dot{V}(x) < 0$ at $x \neq 0$ when $V(x) - V(x(0)) \leq 0$. In other words, (4.4) is asymptotically stable.

Substitute (4.11) into (4.13)

$$
-\sum_{i=1}^{r} \sum_{j=1}^{r} h_i(z) h_j(z) \left( \frac{\partial V(x)}{\partial x} A_i(x) \dot{x} \right) - \frac{1}{4} \frac{\partial V(x)}{\partial x} S_{ij}(x) \left( \frac{\partial V(x)}{\partial x} \right)^T + \dot{x}^T C_i(x)^T Q C_j(x) \dot{x} + \beta_{ij}(x) \{V(x(0)) - V(x)\} \geq 0. 
$$

(4.14)
Next, apply $S$-procedure relaxation to polynomial fuzzy membership functions to (4.14).

\[
- \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(\eta)h_j(\eta) \left( \frac{\partial V(x)}{\partial x} A_i(x) \hat{x} \right)
- \frac{1}{4} \frac{\partial V(x)}{\partial x} S_{ij}(x) \left( \frac{\partial V(x)}{\partial x} \right)^T \hat{x}^T C_i(x)^T QC_j(x) \hat{x}
+ \beta_{ij}(x) \{ V(x(0)) - V(x) \}
+ \sum_{p=1}^{\xi} \theta_p(x, \eta)(\eta_p^{\text{min}})(\eta_p - \eta_p^{\text{max}}) \geq 0.
\] (4.15)

Thus, (4.9) implies (4.15) when $\alpha < 0$.

Integrating (4.15) from $t = 0$ to $t = \infty$, we obtain

\[
J = \int_{0}^{\infty} \left( y^T Q y + u^T R u \right) dt
\leq - \int_{0}^{\infty} \dot{V}(x) dt
- \int_{0}^{\infty} \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(\eta)h_j(\eta)\beta_{ij}(x) \{ V(x(0)) - V(x) \} dt
+ \int_{0}^{\infty} \sum_{p=1}^{\xi} \theta_p(x, \eta)(\eta_p^{\text{min}})(\eta_p - \eta_p^{\text{max}}) dt.
\] (4.16)

Since the closed-loop system (4.4) is asymptotically stable,

\[
J \leq V(x(0))
- \int_{0}^{\infty} \sum_{i=1}^{r} \sum_{j=1}^{r} \gamma_{ij}(x, \eta) \{ V(x(0)) - V(x) \} dt
+ \int_{0}^{\infty} \sum_{p=1}^{\xi} \theta_p(x, \eta)(\eta_p^{\text{min}})(\eta_p - \eta_p^{\text{max}}) dt,
\] (4.17)

where $\gamma_{ij}(x, \eta) = h_i(\eta)h_j(\eta)\beta_{ij}(x)$. Hence,

\[
J \leq V(x(0)) - I_V - I_{\eta},
\] (4.18)
where

\[
I_V = \int_0^\infty \sum_{i=1}^r \sum_{j=1}^r \gamma_{ij}(x, \eta)\{V(x(0)) - V(x)\}dt,
\]

\[
I_\eta = -\int_0^\infty \sum_{p=1}^\xi \theta_p(x, \eta)(\eta_p - \eta_p^{min})(\eta_p - \eta_p^{max})dt.
\]

(Q.E.D.)

**Remark 4.** It is challenging to considered \(I_V\) and \(I_\eta\) in the experiment Therefore, the condition (4.6) is employed instead of (4.18). And by minimizing the upper-bound, the cost function \(J\) can be minimized.

### 4.2 A New Path-Following Based Design

A guaranteed cost controller can be designed by minimizing \(\lambda\) according to the SOS condition in Theorem 4.1. This section illustrates details into steps of how the designing improved a new design based on path following acts, and the relation between parameters will also explain in the upcoming content. Fig. 4.1 is the flowchart of the new proposed algorithm.

**[Iteration Process]**

**Step 1:** Define \(N\) as an iteration counter. Set \(N = 0\), \(\beta_{ijN}(x) = 0\), and \(\lambda_N = \lambda_{upper}\), where \(\lambda_{upper}\) is a fixed positive constant value.Then select an initial \(V_0(x)\) based on linear-quadratic regulator (LQR) solution using the Riccati equation. The process of generating a set of proper initial \(V_0(x)\) will be described in detail by design examples in section 4.2.1. Yet, \(\beta_{ij0} = 0\) is simply designed as initial. Move on to **Step 2**.

**Step 2:** Based on Theorem 4.1, the small disturbances \(\delta V(x)\) and \(\delta \beta_{ij}(x)\) were introduced to solve the following SOS optimization problems. In this step, \(V_N(x)\) and \(\beta_{ijN}(x)\) are fixed according to the precious steps, not decision variables (polynomials); On the other hand, \(\delta V(x)\), \(\delta \beta_{ij}(x)\), and \(\theta_p(x, \eta)\) are the decision variables(polynomials). The bisection searching technique (Section 2.2.4) is employed to \(\alpha_2\) for accelerating the searching speed.

40
Section 4.2 A New Path-Following Based Design

Figure 4.1: The new path-following algorithm structure.

Step 1
Select a linear–quadratic regulator generated $V_N$.
Set $\beta_{ijN} = 0$, $\lambda_N = \lambda_{upper}$, and $N = 0$.

Step 2
Fixed $V_N$ and $\beta_{ijN}$ then minimizing $\alpha_2$
according to [Constrain Group I].
(Decision variables: $\delta V$, $\delta \beta_{ij}$, and $\theta_p$)

Termination condition

End

Continue

$V_N \leftarrow V_N + \delta V$

Step 3
Fixed $V_N$ then minimizing $\alpha_3$
according to [Constrain Group II].
(Decision variables: $\beta_{ijN}$ and $\theta_p$)

Termination condition

End

Continue

$N + 1 \rightarrow N$

$V_{N+1} \leftarrow V_N$
$\beta_{ijN+1} \leftarrow \beta_{ijN}$
$\lambda_{N+1} \leftarrow \lambda_N$

No

$\alpha_3 < 0$?

Yes

$\lambda_{N+1} \leftarrow (1 - \tau)\lambda_N$

End
\( \alpha_{\text{upper}} \) is defined as a large value which is used to determine whether any feasible solutions can be obtained. \( \alpha_{\text{lower}} \) is the lower-bound of \( \alpha_2 \), which is a small negative value. Satisfy \( \alpha_{\text{lower}} < 0 < \alpha_2 < \alpha_{\text{upper}} \).

[Constrain Group I]:

\[
\begin{align*}
\min_{\delta V(x), \delta \beta_{ij}(x), \theta_p(x, \eta)} & \quad \alpha_2 \\
\text{subject to } & \quad (4.19) - (4.25) \\
V_N(x) + \delta V(x) - \epsilon_V(x) & \quad \text{is SOS,} \quad (4.19) \\
\beta_{ij}N(x) + \delta \beta_{ij}(x) & \quad \text{is SOS, } i, j = 1, 2, \cdots, r, \quad (4.20) \\
\theta_p(x, \eta) & \quad \text{is SOS, } p = 1, 2, \cdots, \xi, \quad (4.21) \\
-\{V_N(x(0)) + \delta V(x(0))\} + \lambda_N & \quad \text{is SOS,} \quad (4.22) \\
-\sum_{i=1}^{r} \sum_{j=1}^{r} h_i(\eta) h_j(\eta) \left( \left( \frac{\partial V_N(x)}{\partial x} + \frac{\partial \delta V(x)}{\partial x} \right) A_i(x) \hat{x} \right) \\
-\frac{1}{4} \frac{\partial V_N(x)}{\partial x} S_{ij}(x) \left( \frac{\partial V_N(x)}{\partial x} + \frac{\partial \delta V(x)}{\partial x} \right)^T \\
-\frac{1}{4} \frac{\partial \delta V(x)}{\partial x} S_{ij}(x) \left( \frac{\partial V_N(x)}{\partial x} \right)^T \\
+\hat{x}^T C_i(x)^T QC_j(x) \hat{x} - \alpha_2 \{V_N(x) + \delta V(x)\} \\
+\beta_{ij}N(x)\{V_N(x(0)) + \delta V(x(0)) - V_N(x) - \delta V(x)\} \\
+\delta \beta_{ij}(x)\{V_N(x(0)) - V_N(x)\} \\
+\sum_{p=1}^{\xi} \theta_p(x, \eta)(\eta_p - \eta_p^{\min})(\eta_p - \eta_p^{\max}) & \quad \text{is SOS,} \quad (4.23) \\
\end{align*}
\]

\[
\begin{align*}
\nu_1^T & = \begin{bmatrix} \epsilon_1 V_N^2(x) & \delta V(x) \\ \delta V(x) & 1 \end{bmatrix} \nu_1 \quad \text{is SOS,} \quad (4.24) \\
\nu_2^T & = \begin{bmatrix} \epsilon_2 \beta_{ij}^2 N(x) & \delta \beta_{ij}(x) \\ \delta \beta_{ij}(x) & 1 \end{bmatrix} \nu_2 \quad \text{is SOS,} \quad (4.25) \\
\end{align*}
\]

where \( \epsilon_V(x) \) is a radially unbounded positive-definite polynomial, which is a relax variable to avoid \( V_N(x) + \delta V(x) = 0 \) state, maintain the positivity. \( \nu_1 \) and \( \nu_2 \) are vectors that are independent of \( x \). \( \theta_p(x, \eta) \) (\( p = 1, 2, \cdots, \xi \)) are polynomials. \( \epsilon_1 \) and \( \epsilon_2 \) are small positive constants for ensuring that \( \delta V(x) \) and \( \delta \beta_{ij}(x) \) are small. After the minimum \( \alpha_2 \) is found, update the solution:

\[
V_N(x) \leftarrow V_N(x) + \delta V(x), \quad (4.26)
\]
then go to **Step 3.** ‘←’ stands for substitution.

**[Terminal Scenario in Step 2]:** The terminal scenario in Step 2 is when no feasible solution, which satisfy (4.19) - (4.25) can be obtain, even when \( \alpha_2 = \alpha_{upper} \). When the scenario described above occurs, terminate the iteration.

**Step 3:** With \( V_N(x) \), updated in Step 2, solve the following SOS conditions. Since in conditions (4.27) - (4.30), \( \beta_{ij}(x) \) is a decision polynomial (variable); therefore, it is no need to be updated in Step 2. The Bisection searching technique is also employed to \( \alpha_3 \), the idea and design are the same as \( \alpha_2 \) described in Step 2, thus \( \alpha_3 \in [\alpha_{lower}, \alpha_{upper}] \).

**[Constrain Group II]:**

\[
\begin{align*}
\min_{\beta_{ij}(x), \theta_p(x, \eta)} \alpha_3 & \quad \text{subject to (4.27) - (4.30)} \\
-V_N(x(0)) + \lambda_N & \quad \text{is SOS,} \quad (4.27) \\
\beta_{ij}(x) & \quad \text{is SOS, } i, j = 1, 2, \cdots, r, \quad (4.28) \\
\theta_p(x, \eta) & \quad \text{is SOS, } p = 1, 2, \cdots, \xi, \quad (4.29) \\
- \sum_{i=1}^{r} \sum_{j=1}^{r} h_i(\eta)h_j(\eta) \left( \frac{\partial V_N(x)}{\partial x} - A_i(x) \hat{x} \right) & \\
- \frac{1}{4} \frac{\partial V_N(x)}{\partial x} - S_{ij}(x) \left( \frac{\partial V_N(x)}{\partial x} \right)^T & \\
+ \hat{x}^T C_i(x)^T Q C_j(x) \hat{x} & \\
+ \beta_{ij}(x) \{ V_N(x(0)) - V_N(x) \} - \alpha_3 V_N(x) & \\
+ \sum_{p=1}^{\xi} \theta_p(x, \eta)(\eta_p - \eta_p^{min})(\eta_p - \eta_p^{max}) & \quad \text{is SOS(4.30)}
\end{align*}
\]

If a solution with \( \alpha_3 < 0 \) is obtained, go to **Step 4.** If the minimum \( \alpha_3 \) which satisfy (4.27) - (4.30) is greater than zero, update variables (polynomials) according to the following rules:

\[
\begin{align*}
V_{N+1}(x) & \leftarrow V_N(x), \\
\beta_{ij_{N+1}}(x) & \leftarrow \beta_{ij}(x), \\
\alpha_{3,N+1} & \leftarrow \alpha_3, \\
N & \leftarrow N + 1 \quad (4.31)
\end{align*}
\]
go to Step 2.

[Terminal Scenario in Step 3]: One common terminal scenario is no feasible solution, which satisfy (4.27) - (4.30) can be obtain, even \( \alpha_3 = \alpha_{\text{upper}} \). Another scenario is after \( c \)-th iteration, the program starts to compare the latest minimum \( \alpha_3 \) to the one in the past, where \( c \) is a positive constant integer

\[
|\alpha_{3,N} - \alpha_{3,N-c}| < \mu, \ c < N, \tag{4.32}
\]

where \( \mu \) is a very small positive value. \( \alpha_3 \), in some cases, stabilized around a positive value or slightly increased then decrease. Both situations imply \( \alpha_3 \) cannot be minimized any more. When the scenarios described above occur, terminate the iteration.

**Step 4:** The task of Step 4 is to reserve solution information and minimizing \( \lambda \). Entering **Step 4** means that a solution with \( \lambda_N \) (the current minimum \( \lambda \) value) that satisfies Theorem 4.1 (4.5)-(4.9) has been obtained.

\( \lambda^*, \ V^*(x), \ \beta^*_{ij}(x) \), and \( \theta^*_p(x, \eta) \) are reserve as the candidate of final optimal solution.

\[
\lambda^* \leftarrow \lambda_N,  \\
V^*(x) \leftarrow V_N(x),  \\
\beta^*_{ij}(x) \leftarrow \beta_{ijN}(x),  \\
\theta^*_p(x, \eta) \leftarrow \theta_p(x, \eta). \tag{4.33}
\]

Then reset:

\[
V_N(x) \leftarrow V^*(x),  \\
\beta_{ijN}(x) \leftarrow \beta^*_{ij}(x). \tag{4.34}
\]

And minimize \( \lambda \) by rule:

\[
\lambda_N \leftarrow \lambda^* - \tau \lambda^*, \tag{4.35}
\]

where \( \tau \) is a positive value which is greater than 1, then set \( N \leftarrow N + 1 \) and go back to **Step 2**.
Optimized Solution: The optimal solution is \( V^*(x) \), \( \beta^*_i(x) \), and \( \theta^*_p(x, \eta) \), if there exist. And \( \lambda^* \) becomes the minimum \( \lambda \) value.

Remark 5. As we mention in Remark 3, in order to present the most reliable solution, selection the proper solver checking options is an issue that needs to be carefully treated. This means, since this research mainly deals with high-order SOS polynomials, we check the polynomial (by substituting feasible solutions into the considered SOS conditions) to determine whether it is an SOS polynomial in the most rigorous way. The most reliable option is the ‘both’ option. Once the checking result report ‘infeasible,’ we will strictly determine ‘infeasible.’ This double-checking processes are essential for providing trustworthy solutions.

Remark 6. In our experiment (4.24) and/or (4.25) in Step 2 can be optional. These SOS conditions are used to ensure that the decision variables (polynomials) are small disturbances. However, due to these constraints, optimization sometimes requires a longer calculation time to obtain a solution from the given initial setting. Even if we do not consider (4.24) and/or (4.25), the final solution is always satisfied Step 3, which means that is also satisfied Theorem 4.1.

4.2.1 Benchmark Example

This section provides a complex nonlinear system design example to illustrate the practicality and effectiveness of the proposed design algorithm. This example deals with a generalized version of a complex nonlinear system, which was first given in [11]. As a generalized version, the following 2-rules polynomial fuzzy model with parameters \( a \) and \( b \) are considered.

\[
\dot{x} = \sum_{i=1}^{2} h_i(z) \{ A_i(x)\dot{x} + B_i(x)u \},
\]

(4.36)
where \( \mathbf{x} = \hat{\mathbf{x}} = [x_1 \ x_2]^T \) and \( z = x_1 \). \( A_i(\mathbf{x}) \) and \( B_i(\mathbf{x}) \) matrices are given as

\[
A_1(\mathbf{x}) = \begin{bmatrix}
-1 + x_1 + x_1^2 + x_1x_2 - x_2^2 & 1 \\
-a & -1
\end{bmatrix},
\]

\[
A_2(\mathbf{x}) = \begin{bmatrix}
-1 + x_1 + x_1^2 + x_1x_2 - x_2^2 & 1 \\
0.2172a & -1
\end{bmatrix},
\]

\[
B_1(\mathbf{x}) = \begin{bmatrix}
 x_1 \\
b
\end{bmatrix}, \quad B_2(\mathbf{x}) = \begin{bmatrix}
 x_1 \\
b
\end{bmatrix},
\]

\[
C_1(\mathbf{x}) = C_2(\mathbf{x}) = I
\]

where

\[
h_1(z) = \frac{\sin x_1 + 0.2172x_1}{1.2172x_1},
\]

\[
h_2(z) = \frac{x_1 - \sin x_1}{1.2172x_1}.
\]

It should be emphasized that LMI-based design techniques cannot be applied to complex nonlinear system design examples, because \( A_i(\mathbf{x}) \) and \( B_i(\mathbf{x}) \) are given as polynomial matrices.

Introducing \( \mathcal{S} \)-procedure relaxation concept to the membership functions; therefore, it can be redefined as

\[
h_1(\eta) = \frac{\eta_1}{1.2172} + \frac{0.2172}{1.2172},
\]

\[
h_2(\eta) = -\frac{\eta_1}{1.2172} + \frac{1}{1.2172},
\]

where \( \eta = \eta_1, \eta_1 = \frac{\sin x_1}{x_1} \), \( \eta_1^{\text{min}} = -0.216 \), and \( \eta_1^{\text{max}} = 1 \).

[Initial \( V(\mathbf{x}) \) Candidate Generator]:
Consider \( \mathbf{x}(0) = [10 \ 10]^T \) and defined \( Q = I \) and \( R = I \). The initial settings of \( V(\mathbf{x}) \), i.e., \( V_0(\mathbf{x}) \), are performed as follows. First, a symmetric range for each state variable is considered as \( x_p \in [-10 \ 10] \) for \( p = 1, 2 \), so as to include the initial state as a vertex on the \( x_1 \)-\( x_2 \) space. For the domain \( D = \{(x_1, x_2)| \ |x_1| \leq 10, |x_2| \leq 10\} \) generated by the considered ranges, the
following nine grid points are selected as representative points on the domain $D$.

\[
(\bar{x}_1, \bar{x}_2) = \{(10, 10), (10, 0), (10, -10), (0, 10), (0, 0), (0, -10), (-10, 10), (-10, 0), (-10, -10)\}. 
\]

Regardless of the initial state, the range can be changed according to the situation under consideration. Yet, it is one of the reasonable design methods to choose according to the scope of the initial state. The resolution can also be changed by considering the calculation requirements and/or time allowed in the computing environment. By replacing the $\bar{x}_1$ and $\bar{x}_2$ in each grid with $x_1$ and $x_2$ in $A_i(x)$ and $B_i(x)$ elements, respectively. The following constant matrices $\bar{A}_1$, $\bar{A}_2$, $\bar{B}_1$, and $\bar{B}_2$ can be obtained.

\[
\bar{A}_1 = \begin{bmatrix}
-1 + \bar{x}_1 + \bar{x}_1^2 + \bar{x}_1 \bar{x}_2 - \bar{x}_2^2 & 1 \\
-a & -1
\end{bmatrix},
\]
\[
\bar{A}_2 = \begin{bmatrix}
-1 + \bar{x}_1 + \bar{x}_1^2 + \bar{x}_1 x_2 - \bar{x}_2^2 & 1 \\
0.2172a & -1
\end{bmatrix},
\]
\[
\bar{B}_1 = \begin{bmatrix}
\bar{x}_1 \\
b
\end{bmatrix}, \quad \bar{B}_2 = \begin{bmatrix}
\bar{x}_1 \\
b
\end{bmatrix}.
\]

($A^*, B^*$) pairs are present in varying proportions:

\[
A^* = \bar{w}^* \bar{A}_1 + (1 - \bar{w}^*) \bar{A}_2, \]
\[
B^* = \bar{w}^* \bar{B}_1 + (1 - \bar{w}^*) \bar{B}_2, \]

where $0 \leq \bar{w}^* \leq 1$. The resolution of $\bar{w}^*$ may be changed according to allowable computational requirement and/or time. The following experiment, we considered three different proportion cases $\bar{w}^* = 0, 0.5, 1.0$ for each grid points (4.37), therefore there are $9 \times 3$ different initial settings. By applying 27 pairs of ($A^*, B^*$) to the Riccati equation, 27 initial settings of $V_0(x)$ LQR solutions can be obtained. The 27 initial settings of $V(x)$ will be analysed after implementing the iteration, and the result which obtains the smallest value of $\lambda$ among all the settings will be selected finally.

In the design algorithm, the parameters are set as $\lambda_{upper} = 800$, $c = 20$, $\tau = 0.01$, $\alpha_{upper} = 5000$, $\epsilon_v(x) = 10^{-6}(x_1^2 + x_2^2)$, $\epsilon_1 = 0.005$, and $\mu = 0.001$. As mentioned in Remark
To the best of our knowledge, [20] is the only research which proposed SOS-based guaranteed cost design. The existing SOS-based guaranteed cost design [20] is compared with the proposed method in this chapter. The result presents in Table 4.1 shows the utility of the proposed algorithm, none of the feasible solution can be found for \((a, b) = (3, 1)\) in the existing SOS-based guaranteed cost design. The value of \(J\) is calculated through simulations with the initial state \(x(0)\).

![Figure 4.2: Benchmark Example: the behavior of complex nonlinear system with u=0.](image)

Table 4.1: Comparison results of \(\lambda\) and \(J\) for benchmark example \((a, b) = (3, 1)\)

<table>
<thead>
<tr>
<th>Methods</th>
<th>(\lambda)</th>
<th>(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex Algorithm [20]</td>
<td>2595.50</td>
<td>2552.30</td>
</tr>
<tr>
<td>Nonconvex Algorithm (chap. 3)</td>
<td>391.10</td>
<td>201.23</td>
</tr>
<tr>
<td>Nonconvex Algorithm (chap. 4)</td>
<td>376.20</td>
<td>44.89</td>
</tr>
</tbody>
</table>

Figure 4.4 shows the control result of the polynomial fuzzy controller designed by the proposed algorithm.
In our design algorithm, the solutions for $a = 3$ and $b = 1$ are obtained as follows:

\[
\begin{align*}
V(x) &= 2.3114x_1^2 + 0.084877x_1x_2 + 1.3656x_2^2, \\
\beta_{11}(x) &= 0.0024062x_1^2 - 0.0020801x_1x_2 + 0.0063104x_2^2, \\
\beta_{12}(x) &= 0.0000051149x_1^2 - 0.00000818x_1x_2 + 0.00038029x_2^2, \\
\beta_{21}(x) &= 0.0055232x_1^2 - 0.0029544x_1x_2 + 0.001539x_2^2, \\
\beta_{22}(x) &= 0.0055232x_1^2 - 0.0029544x_1x_2 + 0.001539x_2^2, \\
\theta_1(x, \eta) &= 2.7136x_1^4 + 1.6169x_1^3x_2 \\
&\quad + 1.0925x_1^3x_2^2 + 0.015431x_1x_2^3 + 0.0012211x_2^4.
\end{align*}
\]

4.3 Lower Upper-Bound Estimation: Benchmark Example

This section proposes a new design framework based on minimizing the upper-bound of the cost function $J$. Theoretically speaking, it is an indispensable and useful estimating the minimum upper-bound without calculating $J$ in the design process. In the linear LQR case, $J = V(x(0)) = \lambda$. However, in the case of nonlinear guaranteed cost design in the previous
Figure 4.4: Control result for the initial state $a = 3$ and $b = 1$. 
chapters, since the derived SOS conditions are sufficient condition, there is usually a gap between $J$ and its upper-bound $\lambda$. The gap between $J$ and $\lambda$ are different due to different design conditions, e.g., the gap between the results present in Table 3.2 and Table 4.1 are about 489 and 332, respectively. Consequently, it is tricky to deliver the analytical optimal solution.

The most natural way to obtain the value of $J$ is to carry out simulations. However, it is not an inherent way of solving in the systematic control designs. Therefore, it is worth to estimate the lower upper-bound in the design process without calculating $J$. This section shows that a lower upper-bound estimation can be achieved by increasing the order of polynomial functions $V(x)$.

The notation $[d]$ is introduced to note the order of the polynomial functions, where $d$ is a positive integer. For example, $V_{[d+2]}(x)$ means a $(d + 2)$-th order polynomial function.

In the below content the notation $[d]$ is introduced to indicates the order of the polynomial function $V(x)$, e.g. $V_{[d+2]}(x)$ means a $(d + 2)$-th order polynomial function.

Theorem 4.2 gives a set of SOS conditions to achieve the lower upper-bound estimation in our design framework, and takes the following simple polynomial fuzzy controller as an example.

$$ u = -\frac{1}{2} R^{-1} \left( \sum_{i=1}^{r} h_i(z) B_i(x) \right)^T \left( \frac{\partial V_{[2]}(x)}{\partial x} \right)^T $$

(4.39)

However, the SOS condition which will be illustrated in Theorem 4.2 can be easily extended to estimate the lower upper-bound of a higher-order polynomial fuzzy controller.

**Theorem 4.2.** Assumed $V_{[2]}(x)$ is a solution which satisfying (4.5)-(4.9). With the polynomial fuzzy controller (4.39), if there exist a polynomial $V_{[d]}(x)$ ($d = 4, 6, 8, \cdots$), $\beta_{ij}(x)$, and $\theta_p(x, \eta)$ such that (4.40)-(4.44) are satisfied with scalar $\alpha < 0$, then it can be expected to have a smaller $\lambda$ (or at least the same value as $\lambda$ obtained in Theorem 4.1). In addition, $\lambda$
satisfies that $J \leq V_d(x(0)) - I_V - I_\eta \leq \lambda - I_V - I_\eta$.

$$V_d(x) - \epsilon_V(x) \text{ is SOS,}$$

$$-V_d(x(0)) + \lambda \text{ is SOS,}$$

$$\beta_{ij}(x) \text{ is SOS, } i, j = 1, 2, \ldots, r,$$

$$\theta_p(x, \eta) \text{ is SOS, } p = 1, 2, \ldots, \xi,$$

$$-\sum_{i=1}^r \sum_{j=1}^r h_i(\eta) h_j(\eta) \left( \frac{\partial V_d(x)}{\partial x} A_i(x) \hat{x} \right) + \hat{x}^T C_i(x)^T QC_j(x) \hat{x}$$

$$+ \frac{1}{4} \frac{\partial V_2(x)}{\partial x} S_{ij}(x) \left( \frac{\partial V_2(x)}{\partial x} \right)^T + \beta_{ij}(x) \{V_2(x(0)) - V_2(x)\} - \alpha V_2(x)$$

$$+ \sum_{p=1}^{\xi} \theta_p(x, \eta)(\eta_p - \eta_{p_{\text{min}}})(\eta_p - \eta_{p_{\text{max}}}) \text{ is SOS.}$$

\(\textbf{Proof:}\) It can be directly obtained from Theorem 4.1.

(Q.E.D.)

\textbf{Remark 7.} In Theorem 4.2, $V_2(x)$ is given not a decision polynomial (variable). Theorem 4.2 presents convex SOS conditions with respect to the decision variables (polynomials) $V_d(x), \beta_{ij}(x),$ and $\theta_p(x, \eta)$ when $\lambda$ is given. A simple observation in terms of being represented by (4.12) is that

$$\dot{V}_2(x) \leq \dot{V}_d(x) \leq -(x^T Q x + u^T R u)$$

implies

$$J \leq V_d(0) \leq V_2(0).$$

If the coefficients of higher-order (more than the 2nd order) terms in $V_d(x)$ are all zero, it is clear that $V_d(x)$ simply reduces to $V_2(x)$. That is, a less conservative result is at least obtained by increasing $d$. Thus, by minimizing $\lambda$ (e.g., with the bisection searching technique) subject to (4.40)-(4.44), obtaining a smaller upper-bound of the cost function value $J$ can
be expected. In summary, Theorem 4.2 gives a practical and reasonable calculation way of a lower upper-bound for a given controller.

**Remark 8.** Theorem 4.2 shows that a larger $d$ is expected to obtain a smaller upper-bound. On one hand, higher order settings cause computational difficulty due to larger computational requirement of the SOS solver in practice although higher order settings are highly expected to bring less conservative results with smaller upper-bounds.

### 4.3.1 Complex Nonlinear System Design Example

Refer to a complex non-linear system design example, which uses the same settings as the previous example.

Table 4.2 presents the minimum upper-bound $\lambda$ with different order of $V_{[d]}$, where $d = 2, 4, 6$. In Table 4.2 we can observe that the $\lambda$ value when $d = 4$ is much smaller than when $d = 2$, which is close to the value of $J$ that we obtained in the previous section. Therefore,

<table>
<thead>
<tr>
<th>Methods</th>
<th>$\lambda$</th>
<th>$J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d = 2$</td>
<td>435</td>
<td>250</td>
</tr>
<tr>
<td>$d = 4$</td>
<td>258</td>
<td></td>
</tr>
<tr>
<td>$d = 6$</td>
<td>258</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: The minimum $\lambda$ in different order of $V$. 

53
Chapter 4 The New Polynomial Fuzzy Controller and Lower Upper-Bound Estimation

The 2-th order solutions are:

\[ V_{[2]}(x) = 3.303188x_1^2 + 0.0166548x_1x_2 + 1.0273185x_2^2, \]
\[ \beta_{11}(x) = 0.0024062x_1^3 - 0.0020801x_1x_2 + 0.0063104x_2^3, \]
\[ \beta_{12}(x) = 0.0057984x_1^2 - 0.0072251x_1x_2 + 0.0071286x_2^2, \]
\[ \beta_{21}(x) = 0.0057984x_1^2 - 0.0072251x_1x_2 + 0.0071286x_2^2, \]
\[ \beta_{22}(x) = 0.0047069x_1^2 - 0.0080765x_1x_2 + 0.0036694x_2^2, \]
\[ \theta_1(x, \eta) = 0.1468355x_1^3 - 0.0099616x_1^2x_2 
\quad + 0.133478x_1^2x_2^2 - 0.0033938x_1x_2^3 + 0.0101705x_2^4. \]

The 4-th order solutions are:

\[ V_{[4]}(x) = 0.013101x_1^4 - 0.063572x_1^3x_2 + 0.086933x_1^2x_2^2
\quad + 0.0092953x_1x_2^3 + 0.025359x_2^4 + 0.45949x_1^3 
\quad -1.2617x_1^2x_2 - 0.43325x_1x_2^2 - 0.19311x_2^3 
\quad + 7.9917x_1^2 + 0.73262x_1x_2 + 1.0291x_2^2, \]
\[ \beta_{11}(x) = 0.010398x_1^4 - 0.019926x_1^3x_2 + 0.013759x_1^2x_2^2 
\quad - 0.0040904x_1x_2^3 + 0.0005451x_2^4 - 0.011487x_1^3 
\quad + 0.0044679x_1^2x_2 + 0.0036725x_1x_2^2 - 0.0020989x_2^3 
\quad + 0.011263x_1^2 + 0.0005879x_1x_2 + 0.0030336x_2^2, \]
\[ \beta_{12}(x) = 0.011552x_1^4 - 0.022476x_1^3x_2 + 0.015635x_1^2x_2^2 
\quad - 0.0045807x_1x_2^3 + 0.00056543x_2^4 - 0.0083558x_1^3 
\quad + 6.1462 \times 10^{-5}x_1^2x_2 + 0.005926x_1x_2^2 - 0.0022165x_2^3 
\quad + 0.012419x_1^2 - 0.0038268x_1x_2 + 0.0034371x_2^2, \]

(4.47)
\[ \beta_{21}(x) = 0.011552x_1^4 - 0.022476x_1^3x_2 + 0.015635x_1^2x_2^2 \\
+0.0045807x_1x_2^3 + 0.00056543x_2^4 - 0.0083558x_1^3 \\
+6.1462 \times 10^{-5}x_1^2x_2 + 0.005926x_1x_2^2 - 0.0022165x_2^3 \\
+0.012419x_1^2 - 0.0038268x_1x_2 + 0.0034371x_2^2, \]

\[ \beta_{22}(x) = 0.0098488x_1^4 - 0.020052x_1^3x_2 + 0.014312x_1^2x_2^2 \\
-0.0041627x_1x_2^3 + 0.00043037x_2^4 - 0.0020149x_1^3 \\
-0.0044675x_1^2x_2 + 0.006195x_1x_2^2 - 0.001402x_2^3 \\
+0.0013171x_1^2 - 7.3678 \times 10^{-5}x_1x_2 + 0.0012941x_2^2, \]

\[ \theta_1(x, \eta) = 0.084334x_1^6 - 0.24975x_1^5x_2 + 0.30405x_1^4x_2^2 \\
-0.17393x_1^3x_2^3 + 0.11476x_1^2x_2^4 - 0.0041316x_1x_2^5 \\
+0.00067074x_2^6 + 1.1954x_1^5 - 1.9982x_1^4x_2 \\
-0.42304x_1^3x_2^3 + 0.15416x_1^2x_2^4 - 0.31837x_1x_2^5 \\
+0.0043619x_2^6 + 5.2012x_1^4 - 1.0562x_1^3x_2 \\
+1.8183x_1^2x_2^2 + 0.081738x_1x_2^3 + 0.28733x_2^4 \\
+0.91183x_1^3 + 3.9219x_1^2x_2 + 1.0049x_1x_2^2 \\
-1.4966x_2^3 + 3.6755x_1^2 - 2.7301x_1x_2 + 2.1964x_2^2. \]

As mentioned above, the gap between the value of \( J \) and the upper-bound obtained through Theorem 4.2. The gap also includes the margin due to \( I_V + I_\eta \), as mentioned in Remark 4. Although there is a gap still the higher-order, a larger \( d \), achieve a smaller upper-bound is expected.

**Remark 9.** Since \( b = 1 \) (i.e., \( b \neq 0 \)) in this design example, we note that the order of Lyapunov functions in the existing SOS-based design [20] should be two in order to guarantee the global stability of the control system. More exactly, Lyapunov functions whose degree is more than two become rational polynomial (not polynomial). It is directly related to the fact that the global stability of the control system cannot be generally guaranteed in the existing SOS-based design.
This chapter extends the concept from the previous chapters to a practical parafoil wing-type unmanned aerial vehicles (UAVs) model. By applying the controller introduced in Chapter 4, to stabilize the flying height of the UAV at a specified height. Also the simulation results of estimating the lower upper-bound of the performance function $\lambda$ is provided.

5.1 System Configuration of the Parafoil Wing-Type UAV

The powered paraglider-type UAV as shown in Fig. 5.1 has lines (cables) connected from the front and rear edges of the parafoil to the direction control bar on the body, which is controlled by a servo motor. Table 5.1 shows the dimensions/specifications for the UAV platform [47]. Fig. 5.2 presents the parafoil wing-type UAV platform configuration which
consist of an electric speed controller, a DC motor, a server motor, a gyro, a magnetometer, a processor, an electric speed controller, a wireless module unit, etc. The wireless module is used to communicate with a computer on the ground station for monitoring the flying condition. A global positioning system (GPS) at every 100 ms measures the position of the UAV. This UAV system has two outputs. One is a propeller controlled by a DC motor for altitude control. The other is a direction control bar controlled by a servo motor, which is used to change the flight direction. The change of control bar angle causes a change in the direction of the lift/drag force generated by the parafoil. As a result, the UAV can turn left or right. In other words, since the UAV platform has never left and right brakes on the left and right wings, only servo motors are used to control its direction. This mechanical structure means that the speed and altitude of the UAV cannot be controlled separately. Thus it is difficult to control.

Table 5.1: Dimensions and specifications of UAV platform.

<table>
<thead>
<tr>
<th>Body</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (L×D×H)</td>
<td>390×75×285 [mm]</td>
</tr>
<tr>
<td>Weight</td>
<td>1.196 [kg]</td>
</tr>
<tr>
<td>Four-bladed propeller (diameter × pitch)</td>
<td>8 × 6 [inch]</td>
</tr>
<tr>
<td>Distance between body center and parafoil center</td>
<td>1.2 [m]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parafoil</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>0.170 [kg]</td>
</tr>
<tr>
<td>Length of arc, span</td>
<td>1.52, 0.62 [m]</td>
</tr>
<tr>
<td>Area</td>
<td>0.97 [m²]</td>
</tr>
</tbody>
</table>
5.2 Mathematical Model of UAV

Figure 5.3: Variables and parameters in a rigid body dynamics of UAV model.

Considering the parafoil wing-type UAV as a one-rigid body model, the equation of motion in the x-z plane in the absolute coordinate system, as Fig. 5.3 shown, can be expressed as the following three-degree-of-freedom model.

\[
(M + m)\ddot{z}(t) = L(t)\cos\varphi(t) - D(t)\sin\varphi(t) + T(t)\sin\theta(t) - (M + m)g. \tag{5.1}
\]

\[
(M + m)\ddot{x}(t) = T(t)\cos\theta(t) - D(t)\cos\varphi(t) - L(t)\sin\varphi(t). \tag{5.2}
\]

\[
I_\theta \ddot{\theta}(t) = l\{D(t)\cos(\theta(t) + \gamma - \varphi(t)) - L(t)\sin(\theta(t) + \gamma - \varphi(t))\} + dT(t). \tag{5.3}
\]

where

\[
v^2(t) = \dot{x}^2(t) + \dot{z}^2(t),
\]

\[
L(t) = \frac{1}{2}C_L\rho S v^2(t),
\]

\[
D(t) = \frac{1}{2}C_D\rho S v^2(t),
\]

\[
\varphi(t) = \tan^{-1}\left(\frac{\dot{z}(t)}{\dot{x}(t)}\right).
\]
The definition of each variables can be refer to the Table 5.3.

Table 5.2: Lists of the parafoil wing-type UAV variables.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(t)$</td>
<td>$x$ position (m)</td>
</tr>
<tr>
<td>$z(t)$</td>
<td>$z$ position (m)</td>
</tr>
<tr>
<td>$\varphi(t)$</td>
<td>Velocity vector angle (rad.)</td>
</tr>
<tr>
<td>$\theta(t)$</td>
<td>Pitch angle (rad.)</td>
</tr>
<tr>
<td>$\gamma(t)$</td>
<td>Relative angle of canopy to body (rad.)</td>
</tr>
<tr>
<td>$T(t)$</td>
<td>Thrust (N)</td>
</tr>
<tr>
<td>$D(t)$</td>
<td>Drag force (N)</td>
</tr>
<tr>
<td>$L(t)$</td>
<td>Lift Force (N)</td>
</tr>
<tr>
<td>$u(t)$</td>
<td>System input (N)</td>
</tr>
</tbody>
</table>

After a series of derivations and assume that the UAV flies around the equilibrium point (assuming that it is flying in a straight line).

\[
\dot{z}(t) = \dot{z}_r(t) \tag{5.4}
\]
\[
\dot{x}(t) = \dot{x}_r(t) + v_{xn} \tag{5.5}
\]
\[
T(t) = u(t) + T_n \tag{5.6}
\]
\[
\theta(t) = \theta_r(t) + \theta_n \tag{5.7}
\]

where $\theta_n = 15.3396(\text{deg.})$, $v_{xn} = 6.5598(\text{m/s})$, and $T_n = 4.4584(\text{N})$, which are obtained by real flying experiment. The above equations of motion are nonlinear dynamics around a considered equilibrium point. Also the angle ($\varphi(t)$) with the air flowing into the parafoil can be approximated to near 0. The following mathematical model of UAV can be derived:

Table 5.3: List of the parafoil wing-type UAV parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>$1.700 \times 10^{-1}$ (kg)</td>
<td>Mass of canopy</td>
</tr>
<tr>
<td>$M$</td>
<td>1.197 (kg)</td>
<td>Mass of body</td>
</tr>
<tr>
<td>$C_L$</td>
<td>$4.660 \times 10^{-1}$</td>
<td>Lift coefficient</td>
</tr>
<tr>
<td>$C_D$</td>
<td>$1.640 \times 10^{-1}$</td>
<td>Drag coefficient</td>
</tr>
<tr>
<td>$\rho$</td>
<td>1.293 (kg/m$^3$)</td>
<td>Air density</td>
</tr>
<tr>
<td>$l$</td>
<td>$8.358 \times 10^{-1}$ (m)</td>
<td>Length of canopy cable</td>
</tr>
<tr>
<td>$d$</td>
<td>$4.030 \times 10^{-2}$ (m)</td>
<td>Distance between center of gravity of body and thrust point</td>
</tr>
<tr>
<td>$S$</td>
<td>$9.424 \times 10^{-1}$ (m$^2$)</td>
<td>Wing area</td>
</tr>
<tr>
<td>$I_b$</td>
<td>$6.900 \times 10^{-3}$ (kgm$^2$)</td>
<td>Moment around y-axis (body)</td>
</tr>
</tbody>
</table>
where $\mathbf{q}$ vector is defined as

$$\mathbf{q} = \begin{bmatrix} z_r(t) \\
\dot{z}_r(t) \\
\theta_r(t) \\
\dot{\theta}_r(t) \end{bmatrix}$$

A polynomial model is obtained as

$$\mathbf{q}(t) = \mathbf{A}(\mathbf{q}(t))\mathbf{q}(t) + \mathbf{B}(\mathbf{q}(t))u(t), \quad (5.10)$$

where $\mathbf{q} = [z_r(t), \dot{z}_r(t), \theta_r(t), \dot{\theta}_r(t)]^T$. Since $x(t)$ is used as the x position of PPG, the state vector is defined as $\mathbf{q}(t)$. $\mathbf{A}(\mathbf{q}(t))$ and $\mathbf{B}(\mathbf{q}(t))$ matrices are given as

$$\mathbf{A}(\mathbf{q}(t)) = \begin{bmatrix} 0 & 1 & 0 & 0 \\
0 & \frac{1}{2(M+m)}C_L\rho Sz_r(t) - \frac{1}{2(M+m)}C_D\rho S \left( v_{xn} + \frac{\dot{z}_r(t)}{v_{xn}} \right) & v_{xn} & T_n \cos \theta_n \cos \gamma \\
0 & 0 & 0 & 1 \\
0 & \frac{1}{I_b}A_{42}(\dot{z}_r(t), \theta_r(t)) & \frac{1}{I_b}A_{43} & 0 \end{bmatrix} \quad (5.11)$$

$$A_{42}(\dot{z}_r(t), \theta_r(t)) = t \left( \frac{1}{2}C_D\rho S \left( v_{xn} \sin(\theta_n + \gamma) + \dot{z}_r(t) \cos(\theta_n + \gamma) \right) \\
+ \frac{1}{2}C_L\rho S \left( v_{xn} \cos(\theta_n + \gamma) - \dot{z}_r(t) \sin(\theta_n + \gamma) \right) \right)$$

$$+ l \left( \frac{1}{2}C_D\rho S \left( \frac{\ddot{z}_r(t)}{v_{xn}} - \theta_r(t) \dot{z}_r(t) \right) \sin(\theta_n + \gamma) \\
- \frac{1}{2}C_L\rho S \left( \theta_r(t) \dot{z}_r(t) - \frac{\ddot{z}_r(t)}{v_{xn}} \right) \cos(\theta_n + \gamma) \right) \quad (5.12)$$

$$A_{43} = -\frac{1}{2}C_D\rho S l v_{xn}^2 \sin(\theta_n + \gamma) - \frac{1}{2}C_L\rho S l v_{xn}^2 \cos(\theta_n + \gamma) \quad (5.13)$$
The above polynomial model looks complicated, but note that most of them are constants. Only $\dot{z}_r(t)$, $z_r^2(t)$, $\theta_r(t)$ and their cross terms in $A(q(t))$ and $B(q(t))$ matrices are variables. It is not a polynomial fuzzy model, just a polynomial. Hence, the $S$-procedure relaxation for the polynomial membership functions are not available for this system. This means that we do not have to consider it in the design process. We can design the HJB-based guaranteed cost controller as just one rule polynomial fuzzy model.

### 5.3 Path-Following Based Design for Guaranteed Cost Control for the UAV system

This section we applied the same design processes described in Section 4.2. However, the employed UAV designed model is a single rule polynomial fuzzy model. Thus, in the following content we will explain how to applied our algorithm to an practical single rule polynomial fuzzy model.

Applying Theorem 4.1 and the design processes written in Chapter 4.2, the path-following based design for guaranteed cost control for the UAV system can be realized. The UAV system model is a one rule polynomial fuzzy model, thus the controller is designed without employed $S$-procedure relaxation for polynomial membership functions.

**[Iteration Process]**

**Step 1:** Define $N$ as an iteration counter. Set $N = 0$, $\beta(q) = 0$, and $\lambda_N = \lambda_{upper}$, where $\lambda_{upper}$ is a positive constant relatively larger than the LQR solution. Then select an initial $V_0(q)$ based on linear-quadratic regulator (LQR) solution using the Riccati equation. The details of generating a set of proper initial $V_0(q)$ will be described in detail in the design UAV example in Section 5.4. Move on to **Step 2**.
Figure 5.4: The flowchart of the iteration.
Step 2: Based on Theorem 4.1, the small disturbances $\delta V(x)$ and $\delta \beta(x)$ were introduced to solve the following SOS optimization problems. In this step, $V_N(x)$ and $\beta_N(x)$ are fixed according to precious steps, therefore they are not decision variables (polynomials); On the other hand, $\delta V(x)$ and $\delta \beta(x)$ are decision variables (polynomials). The bisection searching technique is employed to $\alpha_2$ for accelerating the searching speed. $\alpha_{upper}$ is defined as a large value, which is used to determine whether any feasible solutions can be obtained. $\alpha_{lower}$ is the lower-bound of $\alpha_2$, which is a small negative value. Satisfy $\alpha_{lower} < 0 < \alpha_2 < \alpha_{upper}$.

[Constrain Group I]:

\[
\min_{\delta V(q), \delta \beta(q)} \alpha_2 \quad \text{subject to (5.15) - (5.20)}
\]

\[
V_N(q) + \delta V(q) - \epsilon_V(q) \quad \text{is SOS, (5.15)}
\]

\[
\beta_N(q) + \delta \beta(q) \quad \text{is SOS (5.16)}
\]

\[
-\{V_N(q(0)) + \delta V(q(0))\} + \lambda_N \quad \text{is SOS, (5.17)}
\]

\[
\left(\frac{\partial V_N(q)}{\partial q} + \frac{\partial \delta V(q)}{\partial q}\right)A_i(q)\dot{q} - \frac{1}{4} \frac{\partial V_N(q)}{\partial q}S(q)\left(\frac{\partial V_N(q)}{\partial q} + \frac{\partial \delta V(q)}{\partial q}\right)^T
\]

\[
\left(\frac{\partial V_N(q)}{\partial q} S(q) \left(\frac{\partial V_N(q)}{\partial q} + \frac{\partial \delta V(q)}{\partial q}\right)^T
\right)
\]

\[
\dot{q}^T C(q)^T QC(q) \dot{q} - \alpha_2 \{V_N(q) + \delta V(q)\}
\]

\[
+ \beta_N(q)\{V_N(q(0)) + \delta V(q(0)) - V_N(q) - \delta V(q)\}
\]

\[
+ \delta \beta(q)\{V_N(q(0)) - V_N(q)\} \quad \text{is SOS, (5.18)}
\]

\[
v_1^T \begin{bmatrix} \epsilon_1 V_N^2(q) & \delta V(q) \\ \delta V(q) & 1 \end{bmatrix} v_1 \quad \text{is SOS, (5.19)}
\]

\[
v_2^T \begin{bmatrix} \epsilon_2 \beta_N^2(q) & \delta \beta(q) \\ \delta \beta(q) & 1 \end{bmatrix} v_2 \quad \text{is SOS, (5.20)}
\]

where $\epsilon_V(q)$ is the radial unbounded positive definite polynomial and is a slack variable that keeps $V_N(q) + \delta V(q)$ positive. $v_1$ and $v_2$ are vectors that are independent of $q$. $\epsilon_1$ and $\epsilon_2$ are very small positive constants for guaranteed that $\delta V(q)$ and $\delta \beta(q)$ are small. After minimizing $\alpha_2$ subject to (5.15) - (5.20), if there is a solution with minimum $\alpha_2$, update
$V_N(x)$ by the rule:

$$V_N(q) \leftarrow V_N(q) + \delta V(q), \quad (5.21)$$

then go to Step 3. $\alpha_{upper}$ is a large number and denotes the upper limitation to judge whether any feasible solutions or not.

[Terminal Scenario in Step 2]: The terminal scenario in Step 2 is when no feasible solution, which satisfy (5.15) - (5.20) can be obtain, even when $\alpha 2 = \alpha_{upper}$. When the scenario described above occurs, terminate the iteration.

Step 3: With the updated $V_N(q)$, solve the following SOS optimization problem which is also not a decision variable (polynomial). The following conditions are derived based on Theorem 4.1.

[Constrain Group II]:

$$\min_{\beta(q)} \alpha_3 \quad \text{subject to } (5.22) - (5.24)$$

$$-V_N(q(0)) + \lambda_N \text{ is SOS,} \quad (5.22)$$

$$\beta(q) \text{ is SOS,} \quad (5.23)$$

$$\frac{\partial V_N(q)}{\partial q} A(q) \hat{q} - \frac{1}{4} \frac{\partial V_N(q)}{\partial q} S(q) \left( \frac{\partial V_N(q)}{\partial q} \right)^T$$

$$+ \hat{q}^T C(q)^T QC(q) \hat{q}$$

$$+ \beta(q) \{ V_N(q(0)) - V_N(q) \} - \alpha_3 V_N(q) \text{ is SOS.} \quad (5.24)$$

If a solution with $\alpha_3 < 0$ is obtained, go to Step 4. If the minimum $\alpha_3$ which satisfy (5.22) - (5.24) is positive, update variables (polynomials) in the order described below:

$$V_{N+1}(q) \leftarrow V_N(q),$$

$$\beta_{N+1}(q) \leftarrow \beta(q),$$

$$\alpha_{3,N+1} \leftarrow \alpha_3,$$

$$N \leftarrow N + 1, \quad (5.25)$$

go to Step 2.
[Terminal Scenario in Step 3]: One common terminal scenario is no feasible solution, which satisfy (5.22) - (5.24) can be obtain, even $\alpha_3 = \alpha_{upper}$. Another scenario is after cth-iteration, the program starts to compare the latest minimum $\alpha_3$ to the one in the past.

\[ |\alpha_{3,N} - \alpha_{3,N-c}| < \mu, \ c < N, \]  

(5.26)

where $c$ is a positive constant integer and $\mu$ is a very small positive value. Note that, $\alpha_3$, in some cases, stabilized around a positive value or slightly increased then decrease. Both situations imply $\alpha_3$ cannot be minimized any more. When the scenarios described above occur, we judge the algorithm has reach its limitation. Therefore, terminate the iteration.

**Step 4:** The task of Step 4 is to reserve feasible solution information and minimizing $\lambda$. Entering **Step 4** means that a latest solution with $\lambda_N$ (the current minimum $\lambda$ value) that satisfies (5.22) - (5.24) has been obtained.

$\lambda^*, V^*(x)$, and $\beta^*(x)$ are reserve as the candidate of final optimal solution of the UAV system.

\[
\begin{align*}
\lambda^* & \leftarrow \lambda_N, \\
V^*(q) & \leftarrow V_N(q), \\
\beta^*(q) & \leftarrow \beta(q).
\end{align*}
\]  

(5.27)

Then reset $N = 0$,

\[
\begin{align*}
V_N(q) & \leftarrow V^*(q), \\
\beta_N(q) & \leftarrow \beta^*(q).
\end{align*}
\]  

(5.28)

And minimize $\lambda$ by rule:

\[
\lambda_N \leftarrow \lambda^* - \tau \lambda^*,
\]  

(5.29)

where $\tau$ is a positive value which is greater than 1, then go back to **Step 2**.
Section 5.4 Simulation Results of UAV Model and Comparison

Optimal Solution: The optimal solution is $V^*(q)$, and $\beta^*(q)$, if there exist. And $\lambda^*$ becomes the minimum $\lambda$ value.

5.4 Simulation Results of UAV Model and Comparison

The results of applying Section 5.3 to the UAV model will be demonstrate in the following content. Two initial states, $q(0) = [-5 0 0 0]^T$ and $q(0) = [5 0 0 0]^T$, performance cost, cost function upper-bound, and solutions are provided.

The closed-loop system of a single-rule polynomial fuzzy system of the UAV model can be represented as:

$$\dot{x} = A(q) \dot{q} + B(q) u.$$  

The output equation is designed as:

$$y = C(q) \dot{q},$$

where $C = I$. The cost function is defined as:

$$J = \int_0^\infty \dot{q}^T \begin{bmatrix} C & 0 \\ -F & 0 \end{bmatrix} \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} C \\ -F \end{bmatrix} \dot{q} \, dt,$$

where $R = I$, $Q = [1 0 0 0; 0 0.5 0 0; 0 0 0.5 0; 0 0 0 0.5]$. [Linear-Quadratic regulator for $V(q)$ Initial Set]

The set of initial settings of $V_0(q)$ are generate through following processes. First, we defined a symmetric range for state variables, which are $z_r(t)$, $\dot{z}_r(t)$, $\theta_r(t)$, and $\dot{\theta}_r(t)$. Analysing $A$ and $B$, it is obvious that $z_r(t)$ and $\dot{\theta}_r(t)$ does not exist. Therefore, only the symmetric ranges of $\dot{z}_r(t)$ and $\theta_r(t)$ are considered. For the domain with the considered ranges $D = \{ (\dot{z}_r(t), \theta_r(t)) \mid |\dot{z}_r(t)| \leq 10, |\theta_r(t)| \leq 20(\text{degree}) \}$, the following 33 grid points are
chosen as representative points of the domain $D$.

$$(\dot{z}_r(t), \theta_r(t)) = \{(−10, −20), (−10, 0), (−10, 20), (−8, −20), (−8, 0), (−8, 20),$$

$$(−6, −20), (−6, 0), (−6, 20), (−4, −20), (−4, 0), (−4, 20),$$

$$(−2, −20), (−2, 0), (−2, 20), (0, −20), (0, 0), (0, 20),$$

$$(2, −20), (2, 0), (2, 20), (4, −20), (4, 0), (4, 20),$$

$$(6, −20), (6, 0), (6, 20), (8, −20), (8, 0), (8, 20),$$

$$(10, −20), (10, 0), (10, 20)\}.$$

The resolution can also be changed by considering the calculation requirements and/or time allowed in the computing environment.

The parameters setting in the path-following algorithm are as following: $\epsilon_1 = \epsilon_2 = 0.05,$ $\lambda_{upper} = 500,$ $\alpha_{Max} = 5000,$ $c = 20,$ and $\mu = 0.0001.$

For the case $q(0) = [−5 0 0 0]^T,$ the solution of proposed method is

$$J = 79.93,$$

$$\lambda = 120.11,$$

$$V = 1317.8263 \theta^2_r + 6.2972 \theta_r z_r + 1.8125 \theta_r \dot{\theta}_r - 315.8458 \theta_r \dot{z}_r$$

$$+ 4.8016 z^2_r + 0.049207 z_r \dot{z}_r + 12.4038 z_r \dot{z}_r + 0.71174 \dot{\theta}^2_r$$

$$+ 0.056611 \dot{\theta}_r \dot{z}_r + 41.9198 \dot{z}^2_r,$$

$$\beta = 12.5635 \theta^2_r − 0.15515 \theta_r z_r − 1.0948 \theta_r \dot{\theta}_r − 3.7752 \theta_r \dot{z}_r$$

$$+ 0.002163 z^2_r + 0.0077061 z_r \dot{z}_r + 0.028962 z_r \dot{z}_r + 0.052545 \dot{\theta}^2_r$$

$$+ 0.2175 \dot{\theta}_r \dot{z}_r + 0.31213 \dot{z}^2_r.$$
Figure 5.5: Control results with initial state: $q_0 = [-5 0 0 0]^T$. 
The proposed designed algorithm is also compared with the existing SOS-based guaranteed cost design algorithm [20]; however there is no any feasible solution for neither $Q = [-5 0 0 0]^T$ nor $Q = [5 0 0 0]^T$ initial state.
Figure 5.7: Control results with initial state: $q_0 = [5 \ 0 \ 0 \ 0]^T$. 
Figure 5.8: Control output with initial state: $q_0 = [5 \ 0 \ 0 \ 0]^T$.

**Remark 10.** For this UAV case, we found out it is difficult to obtain a feasible solution while minimizing $\lambda$. An alternative way is to set the $SOSOPT$ option to "fast" in the first place and save all solutions in to $(\alpha < 0) \; \lambda^*, \; V^*, \; \beta^*$. Finally after the program terminated, check those solutions generated through "fast" option by "both" options. And select the solution with the minimum $\lambda$ that pass "both" checking. In this way, the is it still possible to obtain a solution when facing minimizing difficulties, but also can guaranteed the reliability of the solution.
5.5 Lower Upper-Bound Estimation: The UAV Model

The section employed the UAV model to demonstrate the practicality of the proposed lower upper-bound estimation. For more details, please refer to Section 4.3.

Applying Theorem 4.2, the following (5.33)-(5.36) condition specifically derived for the UAV model can be obtained. Following up the two initial state been applied in the previous Section 5.4, \( q_0 = [5 \ 0 \ 0 \ 0]^T \) and \( q_0 = [-5 \ 0 \ 0 \ 0]^T \), which cost function value \( J \) are 79.93 and 99.12, representatively. The following experiments show by increasing \( d \), indicates the order of the polynomial Lyapunov function, can easily estimate the cost function value without operating simulations.

\[
V_{[d]}(q) - \epsilon_V(q) \text{ is SOS}, \quad (5.33)
\]
\[
-V_{[d]}(q(0)) + \lambda \text{ is SOS}, \quad (5.34)
\]
\[
\beta(q) \text{ is SOS}, \quad (5.35)
\]
\[
\begin{align*}
&- \left( \frac{\partial V_{[d]}(q)}{\partial q} A_i(q) \dot{q} \\
&- \frac{1}{2} \frac{\partial V_{[d]}(q)}{\partial q} S(q) \left( \frac{\partial V_{[2]}(q)}{\partial q} \right)^T \\
&+ \dot{q}^T C(q)^T QC(q) \dot{q} \\
&+ \frac{1}{4} \frac{\partial V_{[2]}(q)}{\partial q} S(q) \left( \frac{\partial V_{[2]}(q)}{\partial q} \right)^T \\
&+ \beta(q) \{V_{[2]}(q(0)) - V_{[2]}(q)\} - \alpha V_{[2]}(q) \right) \text{ is SOS}, \quad (5.36)
\end{align*}
\]

where \( V_{[2]}(q) \) is a solution which satisfied (5.15)-(5.20). If there is a solution that satisfied (5.33)-(5.36) with \( \alpha < 0 \), then a smaller \( \lambda \) is expected.

5.6 Results of UAV Lower Upper-Bound Estimation

Table 5.4 shows the lower upper-bound estimation of the UAV model with initial state \( q_0 = [5 \ 0 \ 0 \ 0]^T \) and \( d = 2, 4, 6, 8 \). On the other hand, Table 5.5 shows the case with initial state \( q_0 = [-5 \ 0 \ 0 \ 0]^T \) and \( d = 2, 4, 6 \).

Observe the results of the two cases, when the order of the polynomial Lyapunov function
Chapter 5  Practical System Design: Unmanned Aerial Vehicles

Table 5.4: Minimum $\lambda$ value of different degrees ($q_0 = [-5 0 0 0]^T, J = 79.93$)

<table>
<thead>
<tr>
<th>Methods</th>
<th>$\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d = 2$</td>
<td>120.11</td>
</tr>
<tr>
<td>$d = 4$</td>
<td>81.74</td>
</tr>
<tr>
<td>$d = 6$</td>
<td>81.00</td>
</tr>
<tr>
<td>$d = 8$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

Table 5.5: Minimum $\lambda$ value of different degrees ($q_0 = [5 0 0 0]^T, J = 99.12$)

<table>
<thead>
<tr>
<th>Methods</th>
<th>$\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d = 2$</td>
<td>120.11</td>
</tr>
<tr>
<td>$d = 4$</td>
<td>105.00</td>
</tr>
<tr>
<td>$d = 6$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

$(d)$ increases, $\lambda$ decreases. From the Table 5.4, the greatest order that can be reached is 6-order where $\lambda_{[6]} = 81.00$ and $J = 79.93$; and the Table 5.5 which the maximum order can be reach is 4-order where $\lambda_{[4]} = 105.00$ and $J = 99.12$. In summary, both experimental results illustrate that our proposed method can estimate the cost function value $J$ by increasing the order of the polynomial Lyapunov function.
CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

This thesis has presented a novel idea of designing guaranteed cost control system using nonconvex conditions and solved via path-following algorithm. A set of nonconvex SOS conditions are derived successfully to achieve guaranteed cost control of polynomial fuzzy systems. The proposes algorithm can directly solve the guaranteed cost control under the nonconvex SOS design condition via path-following algorithm, which bypasses the conservation issue caused by applying the typical congruence transformation technique. Moreover, an analytical way to estimate the cost function value without implement simulation was proposed.

Chapter 3 presented a novel double-loop structure path-following design algorithm scheme that minimize the upper-bound of the designed performance function and the non-negativity checking parameters of the SOS design condition in parallel. Also, a copositive relaxation is introduced to brings out more relax SOS conditions representation. And a 3-D polynomial chaotic system and a complex benchmark nonlinear system were employed to demonstrate the effectiveness of the proposed algorithm. Besides that, the comparison tables of the performance function of our design algorithm with a convex design algorithm (Convex Algorithm) [20] and a path-following stabilization algorithm (Stabilization Algorithm) [9] were also provided. Both tables illustrated that our proposed Nonconvex Algorithm obtained the lowest cost of the designed cost function evaluation amount three algorithms.

In Chapter 4, a new type of polynomial fuzzy controller is introduced. Taking the benefit of the new polynomial fuzzy controller based on Hamilton-Jacobi-Bellman inequality, which is theoretically considered as an optimal controller, additionally, the number of decision variables (polynomials) can be reduced. Two $S$-procedure relaxations were introduced in the designed processes, and the results show that the relaxation technique had successfully
relaxed the SOS polynomial conditions. Moreover, a more simple form of the double-loop path-following scheme is designed by a different way of minimizing cost function upper-bound. Also, in the paragraph demonstrated the use of LQR to generate a set of polynomial sets as appropriate initial settings through examples. Another contribution of this chapter is proposing a particular strategy for estimating the cost function value by lower upper-bound estimation. The cost function values can be estimated by fixing the second-order controller and increased the order of the Lyapunov function. Finally, a benchmark example is applied to illustrate the validity of our design algorithm and demonstrated the utility of the low upper-bound estimation method.

In Chapter 5, a practical unmanned aerial vehicles system, which is known as a complex nonlinear system is employed to illustrate the applicability and validity of the proposed algorithm presented in Chapter 4. This chapter results give further proof of the effectiveness of our proposed algorithm and demonstrate the potential of carrying out the real flight experiment in the future.

### 6.2 Future Work

Based on the current results, in the future the following aspects are worth exploring.

First, all the examples provided in this thesis only carry out are 2nd-order Lyapunov functions, it is possible to carry out even better performance after increasing the order of Lyapunov function, i.e., 4th-order, 8th-order. However, it is challenging while extending the algorithm to higher-order for the path-following design structure due to initial setting requirements and also the computational issue. In the 2-nd order path-following structure, we introduce the LQR algorithm to generate potential reasonable variable initials, yet the LQR algorithm can only generate linear controllers. In other words, the strategy of relying on the LQR algorithm to generate initial settings cannot achieve for order higher than 2-nd order. One considered initial setting for higher-order is:

\[
V_{[n]} = V_{[n-2]} \left(1 + \epsilon V_{[n-2]}\right), \quad N > 2.
\]  

(6.1)

where \(V_{[n-2]}\) is a optimal solution of the result in \(n-2\)-order. Take \(n = 4\) as an example, the initial setting of 4-th order can be seen as the optimal solution \(V_{[2]}\) times a small 2nd order
polynomial, which can possibly give a reasonable and potential initial setting for higher-order case.

Secondly, the parafoil wing-type unmanned aerial vehicles model that we employed in Chapter 5 is a simplified mathematical model. The results shows the potential of applying our proposed algorithm to a more complicated model. And our goal is to carried out a real flight experiment in the future to demonstrate an expanded usability of the proposed algorithm [47, 48].

In addition to powered paraglider-type UAV, another type of UAV is flying-wing-type UAV, which is considered to be a very efficiency UAV, as shown in Figure 6.1. The control of flying-wing-type UAVs is challenging because they do not have a vertical stabilizer, that is, no rudder. Therefore, from the perspective of control theory and practice, the trajectory tracking stability of flying-wing-type UAVs is still a challenging problem. Moreover, a so-called vertical take-off and landing (VTOL)-type UAV, shown in Figure 6.2, which has a propeller on each side that can be angled forward or upward, like a multirotor aircraft. This kind of mechanism design gives the advantage of adaptability to the ground during take-off and landing.

In the future, the proposed algorithm could be applied to flying-wing-type or VLOT UAVs. It is expected to achieve efficient flight and guaranteed flying performance in simulations and real flight experiments.
Figure 6.2: The vertical take-off and landing-type UAV.
REFERENCES


References


References


References


List of Publications

Journal paper:


International conference papers:


Other publication: