Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Torjus Klafstad

An Approximation to Model Predictive Control with a Modest Online Computational Demand

Master's thesis in Cybernetics and Robotics
Supervisor: Professor Morten Hovd
June 2019

NTNU
Norwegian University of
Science and Technology

Torjus Klafstad

An Approximation to Model Predictive Control with a Modest Online Computational Demand

Master's thesis in Cybernetics and Robotics
Supervisor: Professor Morten Hovd
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Sammendrag

Modellprediktiv regulering (MPC) er en industriledende form for regulatordesign viden brukt i samtidsapplikasjoner. Det høye beregningskravet assosiert med online optimalisering begrenser derimot antallet systemapplikasjoner hvor MPC kan bli anvendt. Eksplisitt MPC løser MPC-optimaliseringen offline, og lagrer denne løsningen som stykkvis affine funksjoner. Når størrelsen på problemet blir større vil dette føre til enorme krav på lagringskapasitet, noe som motvirker hensikten med en billigere løsning. Andre metoder som approksimerer oppførselen til MPC er dermed av stor interesse innenfor forskning.

En lovende metode som approksimerer MPC basert på Controlled Contractive Sets er presentert i [1]. Metoden utnytter høyere grads Lyapunov-funksjoner for å finne kontraktive sett med større volum. Metoden gir bedre resultater enn tidligere implementasjoner av Controlled Contractive Sets, men problemformuleringen som ble presentert hadde mangler. KKT-betingelsene ble brukt til å formulere optimaliseringsproblemet, men betingelsene gitt var ikke tilstrekkelige, kun nødvendige. Kontraksjonsbegrensningen på Lyapunovfunksjonen må være konveks for at KKT-betingelsene skal være tilstrekkelige, så dette må legges til i problemformuleringen. Formuleringen burde også utvides til å ta hensyn til ulineær dynamikk, da dette øker anvendeligheten til metoden.
Den tilgjengelige programvareimplementasjonen produserte nye kontraktive sett som brøt tilstandsbegrensningene. Dette begrenser anvendeligheten til metoden betydelig, så en løsning på dette problemet må også bli funnet.

Konveksiteten til Lyapunovfunksjonen blir bygd inn i problemformuleringen som en ny begrensning. En funksjon er konveks hvis den Hessiske matrisen assosiert med funksjonen er positiv semidefinitt. Denne begrensningen blir lagt til i problemformuleringen.
Bruddene på tilstandsbegrensningene er vanskeligere å løse. Opphavet til problemet blir forsøkt funnet, dessverre med ubetydelige resultater. Gjennom eksplisitte begrensninger på det kontraktive settet og en økning på det tillatte pådraget, blir et konvekst, kontrollert, kontraktivt sett, som er innenfor tilstandsbegrensningene, funnet.

Den ufullstendige problemformuleringen i [1] blir vist til å produsere ikke-konvekse nivåsett, som impliserer at avstanden fra optimum ikke er garantert. Problemet blir rettet opp når konveksitetsbegrensningen blir lagt til. Metoden produserer nå verifiserbart konvekse nivåsett som garanterer avstanden til optimalitet. Opphavet til problemene med brudd på tilstandsbegrensningene blir ikke funnet, et prioriteringspunkt for videre forskning. En økning på tillatt pådrag, og eksplisitte begrensninger på formen til nivåsettet fører til et kontrollert kontraktivt sett innenfor tilstandsbegrensningene. Dette er presentert som et resultat, selv om det kommer på bekostning av en endring i systembeskrivelsen.

Abstract

Model Predictive Control (MPC) is an industry-leading control scheme widely used in present day applications. The computational demands of online optimization, however, limits the number of systems where the control scheme can be applied. Explicit MPC solves the optimization problem offline, and stores the solution as piece-wise affine functions. An increase in the problem size eventually leads to an enormous storage demand, which directly opposes the intent of a cheaper more applicable solution. Hence other methods for of approximating MPC are still thoroughly researched.

A promising method of MPC approximation based Controlled Contractive Sets is presented in [1]. The method revolves around the use of higher degree Lyapunov functions in the design of the contractive sets. Although the method produced qualitatively better results than previous implementations of controlled contractive sets, the problem formulation presented was incomplete. The KKT conditions presented were not sufficient for optimality. The contraction constraint was not specified to be convex, so this needs to be added to the problem formulation. The formulation should be extended to include the dynamics of nonlinear systems as well to allow for broader applications of the method.

The software implementation supplied produced contractive sets that violated the state constraints. This is problematic as it limits the applicability of the method, so this problem had to be fixed as well.
First, the convexity of the Lyapunov function is guaranteed by including a new constraint in the problem formulation. For a polynomial to be convex, the associated Hessian matrix (matrix of second derivatives) must be positive semi-definite. This constraint is added to the problem formulation.
The state constraint violations are more difficult to solve. An attempt is made to discern the origin of the problems, sadly with little success. Through explicit limitations imposed on the contractive set and an increase in the allowed input, a convex, controlled contractive set within the state constraints is found.

The incomplete formulation of [1] is shown to be able to produce non-convex level sets, which implies that the distance from the optimum is not guaranteed. Adding the convexity constraint corrects the issue. The method now produces verifiably convex level sets, guaranteeing the distance to the optimum. The root of the problem of the state constraint violations is not found, and should be prioritized in further research. Still, a controlled contractive set within the state constraints is found and presented.

Preface

This thesis is written as a part of the requirements for the degree of Master of Science in Cybernetics and Robotics from the Norwegian University of Science and Technology, Trondheim, Norway. The work presented was produced during the spring of 2019 and submitted in June of the same year.

This thesis is a continuation of the method developed in [1], which is an approximation to Model Predictive Control based on Sum of Squares programming. The task description:

- Conduct a literary review of the relevant area, as well as the YALMIP-software for posing and solving SoS-problems.
- Verify the examples of [1], and examine the effect of demanding of the contractive function(that guarantees stability) to be convex.
- Expand the methodology to include input affine rational nonlinear dynamics.
- If there is time remaining, develop an approximately minimum time MPC based on corresponding techniques.

The software implementation of the methodology in [1] I was supplied was unfortunately disorganized and poorly documented. As a consequence, I didn't discover that the code version I was working on did not coincide with the published problem formulation. This severely halted my progress, until a more recent software version was discovered a few weeks before the deadline. This version did not, however, produce results anywhere close to the results presented in [1], and I spent the last weeks trying to get the code to work. This work gave many interesting insights and results, but also caused me to be unable to complete the last two points of the task. The results I have found are presented in the thesis.

The software used for this project is commercially available: YALMIP[2], an optimization toolbox, and MOSEK[3], a Semi-definite Program solver. Both are used in the MATLAB coding environment.

I would like to extend my deepest thanks to my supervisor, prof. Morten Hovd, for tirelessly helping and guiding me, especially during the final few weeks.
Thanks to my family for proofreading and the constructive feedback on my convoluted way of expressing myself.
Thank you to the people of G242-243, the Cubic Army of Four Days, OV, and my fireteam.
Last of all, thank you Victoria, for pushing me to always do my best. Your support is greatly appreciated.

Torjus Klafstad Oslo, June 25, 2019

Contents

List of Figures

Nomenclature

E-MPC	Explicit Model Predictive Control
LF	Lyapunov Function
LMI	Linear Matrix Inequality
PSD	Positive Semi-Definite
PWA	Piece-Wise Affine
SDP	Semi-Definite Program(ming)
SOS	Sum of Squares

Chapter 1

Introduction

1.1 Motivation

The motivation for this thesis stems from a desire for easily applicable and available optimization-based control methods. Today, Model Predictive Control (MPC) is the state of the art for optimization-based control. Traditional MPC solves an optimization problem online, which often requires complex online calculations, and is dependent on complicated code in online implementations. This has reduced the application of MPC to relatively slow systems, and systems which aren't safety critical (or are based on stabilizing control at a lower level). A control method that approximates the behavior of MPC without the need for online optimization would be good for the diversity of control theory, and would provide a more cost-effective solution for more applications.
Simplifying the online computations necessarily makes the offline computations more demanding. Rapid improvement in both the hardware and software for computation and optimization, however, diminishes this constraint. Sophisticated solvers for multiple difficult problems now exist, making it possible to experiment with different approaches for finding an MPC approximation with low online computational demand.

1.2 Background

1.2.1 Model Predictive Control

MPC[4] has been widely used in oil and processing industry since the 1980s. It is an optimization-based control structure. The basic principle of MPC is solving the constrained input optimization problem at every timestep for a finite time horizon, and applying the first input value to the system. The invention of MPC was motivated by a need for a sophisticated control method for industry. Specifically, a control scheme capable of adhering to the multitude of constraints present in industrial systems.

1.2.2 Drawbacks

Solving an optimization problem online obviously limits the areas where an application of MPC is possible, given the computational demands of optimization. The traditional implementation of MPC is limited to systems

1. with sufficiently slow dynamics to accommodate the relatively large computational times
2. that aren't, due to the error prone nature of optimization software, safety critical to the operation
3. where the cost of the high-end computational hardware required is a non-issue
4. with any combinations of the above possibilities.

Most systems in modern industry is better served with an optimization-based controller, but the limitations mentioned above prevents this. Naturally, a solution for this has been sought, and quite well developed, in several forms of Explicit MPC. Different applications, among them [5], were surveyed in [6].

1.2.3 Explicit MPC

The explicit MPC of [5] formulates the MPC problem as a multi-parametric problem, which is done by solving the optimization problem offline. This reduces the control law to a piece-wise affine function of the current state, instead of the solution to an optimization problem. The solution, however, demands an increasing amount of memory when the dimensions of the system increase, limiting the application to systems with a modest amount of states and inputs, as well as relatively short prediction horizons.

1.2.4 Approximating MPC with Controlled Contractive Sets

Another way to approximate the behaviour of MPC is through the use of controlled contractive sets. Instead of storing the entire solution space of MPC, this method finds a control law that qualitatively mimics MPC behaviour. Controlled contractive sets are used in the method of [7]. This method, however, may give highly complex contractive sets. Another method presented in [8] uses a simple, contractive polyhedral set. However, the method is of fixed complexity, so the size of the set cannot be increased by increasing the complexity of the set representation.

A method for approximating MPC with low online computational demands based on controlled contractive sets was developed and presented in [1]. This method is based on Sum of Squares programming, an optimization tool that has gained increased attention since the publication of [9]. In [1], the contractive set is expanded by increasing the polynomial degree of the corresponding Lyapunov and feedback functions. The method was presented, however, with a critical flaw.

This thesis aims to correct the flaw in the formulation of [1], and discuss the resulting problems and gains.

1.3 Thesis Overview

Chapter 2 is a literature review of the relevant subjects discussed in this thesis, namely the KKT conditions, Sum of Squares programming, and Lyapunov Stability for Discrete-time Systems. In Chapter 3 the problem formulation of [1] is presented, discussed, analyzed, and finally corrected in Chapter 4. Chapter 5 discusses the methodology used for the implementation on a system in MATALB. The results of chapter 5 are presented and discussed in chapter 6. A conclusion is drawn and further work is presented ultimately in chapter 7.

Chapter 2

Fundamental Theory

In this chapter, the fundamental theory used later in this thesis is described. Section 2.1 describes the Karush-Kuhn-Tucker(KKT)-conditions, a part of the fundamentals of optimization theory. In Section 2.2, Sum of Squares optimization is described, with emphasis on the benefits of the theory. Finally, Section 2.3 gives a brief summary on Lyapunov control theory for continuous and discrete time systems.

From this chapter on, it is assumed that the reader possesses fundamental knowledge of linear algebra, optimization theory, control theory, and Cybernetics as a field of study. Knowledge on Semi-Definite Programming is recommended, but not required to firmly understand the rest of the thesis.

2.1 Optimization And KKT-conditions

This section gives a short overview of the KKT-conditions used in optimization theory, and the cases where they provide sufficient conditions for optimality. For more on the KKT-conditions, and convex optimization in general, see [10].

A classic optimization is usually given on the following form:

$$\text{Optimize } f(x) \tag{2.1a}$$
$$\text{subject to}$$
$$h_i(x) \leq 0, i = 1, \ldots, n_i \tag{2.1b}$$
$$g_i(x) = 0, i = 1, \ldots, n_e \tag{2.1c}$$

where x is the optimization variable, $f(x)$ is the objective function, $h_i(x)$ are the inequality constraint functions, and $g_i(x)$ are the inequality constraint functions. n_e and n_i are, respectively, the number of equality constraints and the number of inequality constraints. The KKT-conditions for an optimization problem are conditions that must be fulfilled at the optimum for the problem, and hence are always necessary conditions for the optimum. For now, let the optimization be to find the minimum $\min f(x)$. Then the Lagrangian function is defined as:

$$\mathcal{L}(x) = f(x) + \sum_{i=1}^{n_e} \mu_i g_i(x) + \sum_{j=1}^{n_i} \lambda_i h_j(x) \tag{2.2}$$

where μ_i, λ_j are called the Lagrange multipliers. At the optimum the gradient of the Lagrangian is zero:

$$\nabla\mathcal{L}(x^*) = \nabla f(x^*) + \sum_{i=1}^{n_e} \mu_i^* \nabla g_i(x^*) + \sum_{j=1}^{n_i} \lambda_j^* \nabla h_j(x^*) = 0 \tag{2.3}$$

where an asterisk denotes the optimal value.

The value of the Lagrange multipliers describe, in a sense, how much could be gained by expanding the corresponding constraint in the direction of the gradient of the objective function. A zero value indicates that nothing would be gained, i.e. the constraint is inactive. Obviously, this has no influence on the equality constraints, which are always active. This gives rise to the condition *complimentary slackness*:

$$\lambda_i^* h_i(x^*) = 0, i = 1, \ldots, n_i. \tag{2.4}$$

Equation (2.4) has the intuitive interpretation that either the inequality constraint is active at the optimum i.e. $h_i(x^*) = 0$, or inactive, resulting in a zero value multiplier λ_i^*.

Another property of the Lagrangian multipliers is that their value is negative if the corresponding constraint is violated. Hence another condition is

$$\lambda_i^* \geq 0, i = 1, \ldots, n_i. \tag{2.5}$$

The constraints must, naturally, be fulfilled at the optimum as well. This gives the KKT-conditions as below:

$$\nabla f(x^*) + \sum_{i=1}^{n_e} \mu_i^* \nabla g_i(x^*) + \sum_{j=1}^{n_i} \lambda_j^* \nabla h_j(x^*) = 0 \tag{2.6a}$$

$$h_i(x) \leq 0, i = 1, \ldots, n_i \tag{2.6b}$$

$$g_i(x) = 0, i = 1, \ldots, n_e \tag{2.6c}$$

$$\lambda_i^* \geq 0, i = 1, \ldots, n_i \tag{2.6d}$$

$$\lambda_i^* h_i(x^*) = 0, i = 1, \ldots, n_i \tag{2.6e}$$

which are, for any minimization problem, *necessary conditions* for optimality.

If the optimization problem (2.1) describes a *convex* optimization problem, the KKT-conditions are actually *sufficient conditions* for optimality. To have a convex optimization problem, the objective function $f(x)$ and the inequality constraints $h_i(x)$ must be convex, and the equality constraints $g_i(x)$ affine functions. This property will be used to correct the problem formulation of [1] in Chapter 3.

2.2 Sum of Squares Programming

Sum of Squares (SOS) programming is a used to guarantee the non-negativity of a polynomial. It is based on SOS decomposition of polynomials, which provides a sufficient condition for the non-negativity of a polynomial. A polynomial $f(\mathbf{x})$, in n variables where $\mathbf{x} \in \mathbb{R}^n$, that can be written as a sum of squared terms is intuitively non-negative for all $\mathbf{x}$. Determining if a polynomial is non-negative can be shown to be NP-hard [9]. The polynomial having an SOS decomposition, however, is a condition guaranteeing global positivity that can be tested in polynomial time.

A polynomial $f(x)$ can be written in the form

$$f(x) = \sum_k c_k x_1^{a_{k_1}} x_2^{a_{k_2}} \dots x_n^{a_{k_n}}$$

where a_{k_i} are non-negative integers. A monomial $m_(x)$ is one of the k terms in a polynomial $f(x)$, the coefficient c_k not included:

$$m_k(x) = x_1^{a_{k_1}} x_2^{a_{k_2}} \dots x_n^{a_{k_n}}.$$

As an example, take the following polynomial

$$g(x_1, x_2) = 2x_1^2 - 3x_1^3 x_2 + x_1 x_2^2.$$

The corresponding vector of monomials is

$$m_g(x_1, x_2) = \begin{bmatrix} x_1^2 & x_1^3 x_2 & x_1 x_2^2 \end{bmatrix}^T.$$

A sum of squares polynomial is a polynomial of the form $\mathbf{v}(x)^T \mathbf{Q} \mathbf{v}(x)$, where $\mathbf{v}(x)$ is some vector of monomials, and $\mathbf{Q}$ is a square positive semi-definite matrix. The terms in $\mathbf{v}(x)$ are clearly not independent of each other, and thus the matrix $\mathbf{Q}$ will not be unique. Take this example from [9, p. 41]:

$$f(x) = 2x_1^4 + 2x_1^3 x_2 - x_1^2 x_2^2 + 5x_2^4 \tag{2.7}$$

$$= \begin{bmatrix} x_1^2 & x_2^2 & x_1 x_2 \end{bmatrix} \begin{bmatrix} 2 & 0 & 1 \\ 0 & 5 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_1 x_2 \end{bmatrix} \tag{2.8}$$

$$= \mathbf{v}(x)^T \mathbf{Q} \mathbf{v}(x) \tag{2.9}$$

From the leading principle minors of $\mathbf{Q}$ it can clearly be seen that the matrix is not PSD. It is also clear that the elements of $\mathbf{v}(x)$ are dependent on each other; for instance $x_1^2 \cdot x_2^2 = (x_1x_2)^2$. This can be used to modify $\mathbf{Q}$ to be PSD. Introduce the variable λ:

$$\mathbf{Q} = \begin{bmatrix} 2 & -\lambda & 1 \\ -\lambda & 5 & 0 \\ 1 & 0 & -1+2\lambda \end{bmatrix} \tag{2.10}$$

Choosing $\lambda = 3$ makes $\mathbf{Q}$ PSD(two positive, and one zero eigenvalue).

2.2.1 S-procedure

Positivstellensatz?

The form of SOS programming mostly used in this thesis is the S-procedure. The S-procedure is a well known optimization method for enforcing the positivity of a function in a region. Say you want a function $f(x) > 0$ wherever another function $g(x) < 0$. This is solved by optimizing over the condition

$$f(x) + s(x)g(x) > 0 \tag{2.11}$$

for an SOS polynomial $s(x)$. In other words, it is possible, by optimizing over and finding an SOS-polynomial $s(x)$, to enforce the positivity of a function in a defined area.

2.2.2 Solving SOS Programs

SOS optimization programs are solved by posing the problem as an SDP. The process is quite similar to solving Linear Matrix Inequalities(LMIs), which are discussed in detail in [11]. In short, SOS programs are a more specialized case of LMIs.

A constraint of the form (2.11) is one of a few versions of an SOS constraint. Generally, an SOS constraint is an equation that is constrained by having an SOS shape. Common between all SOS constraints is that they are convex, assuming linear dependence on decision variables in the optimization formulation, and linearity in the free variables resulting from the algebraic dependence of elements in the monomial vector $\mathbf{v}(x)$. Combining SOS constraints with a convex optimization criterion formulates a convex SDP problem.

The actual solving of an SOS program is usually handled by an SDP solver, which is quite too technical to explain here. The reformulation of an SOS program into a form possible for an SDP solver to recognize can be done with the help of software with built-in SOS functionality, e.g. YALMIP[12] or SOSTOOLS[13]. The functionality allows the formulation of a constraint as SOS. It is possible to formulate an SOS problem explicitly within these environments(at least in YALMIP), however using the built-in functions will provide as good or better results with less coding time.

For implementations in YALMIP, only a few things need to be initialized to be able to solve the program. Naturally, an optimization criterion and at least one SOS constraint must be defined. For the SOS constraint, the maximal order of the polynomial sought must be defined as well. YALMIP sets up the monomial vector associated with the problem, and utilizes the SDP solver to find the monomial coefficients, which is equivalently the matrix $\mathbf{Q}$. Naturally, $\mathbf{Q}$ must be PSD. YALMIP automatically reduces the complexity of the solution, and with certain options it is possible to impose a block structure on $\mathbf{Q}$ to make the solution more efficient. More documentation on YALMIP can be found on the website `https://yalmip.github.io`.

Solution Validation

The optimized matrix $\mathbf{Q}$ may be, due to the nature of optimization software, numerically ill-conditioned. Although the matrix will be positive definite, some eigenvalues will be very small. This could imply that the original polynomial is in fact not SOS, so the validity of the solution needs to be tested.

One way of testing to see if the original polynomial $f(x)$ is SOS is to look at the difference between the polynomial and the calculated SOS decomposition:

$$f(x) - \mathbf{v}(x)^T \mathbf{Q} \mathbf{v}(x) \tag{2.12}$$

The *residual* of the decomposition is defined as the largest coefficient of (2.12). [12] shows that positivity is guaranteed provided:

1. $\min eig(\mathbf{Q}) \geq 0$
2. $\min eig(\mathbf{Q}) \geq n_{mon} \times abs(residual)$

where n_{mon} is the number of monomials in $\mathbf{v}(x)$.

2.3 Lyapunov Control

Lyapunov-based control is widely used within contemporary technology and theory. In simple terms, the control theory shows that if the energy of a system decreases over time, the system is stable. The control part is choosing a feedback controller that causes the system's energy to decrease. More in-depth on Lyapunov Control can be found in [14], or any other textbook on non-linear systems. A short overview is given below, with specifics for the discrete-time case.

2.3.1 Continuous Systems

Control theory for continuous systems revolves around finding a feedback $u = u(x(t))$ that stabilizes the system. In Lyapunov Control theory, this is done by finding an energy-like function (usually called a Lyapunov function) $V(x)$ that fulfills the following properties:

- Positive definite
- $V(0) = 0$
- Continuous
- Radially unbounded ($V(x) \to \infty$ as $\|x\| \to \infty$)
- $\dot{V}(x) < 0 \; \forall x \neq 0$

The inequality $\dot{V}(x) < 0 \; \forall x \neq 0$ has the input substituted in and is solved for the input, thus granting a feedback that stabilizes the system.

2.3.2 Discrete-time Systems

For discrete-time systems there are a few differences. In discrete time, the time derivative of a function is replaced with the function value at the next timestep. Lyapunov stability is achieved when the function value decreases from one timestep to another. This replaces the last property for the continuous systems with

$$V(x_{k+1}) < V(x_k) \forall x. \tag{2.13}$$

The other properties of the function stay the same.

The method for achieving Lyapunov stability in discrete-time systems used in [1] is called controlled contractive sets. This method will be used on the work done in this thesis, and is presented, along with most of the procedure in [1], in Chapter 3.

Chapter 3

An Overview of the Procedure and Methodology of Munir et.al.(2018)

This chapter serves to clarify and give an overview on the different methods and procedures of [1]. Most of the work of this thesis is based on both the results and the methodology of the article, and showing the respective contributions is of importance. All the following information is summarized from [1].

3.1 Contractive sets

Consider the constrained control of the linear discrete time system:

$$x_{k+1} = Ax_k + Bu_k \tag{3.1}$$

where $x_k \in \mathbb{R}^{n_x}$ is the current state of the system and $u_k \in \mathbb{R}^{n_u}$ the current input. x_{k+1} are the states in the next timestep. Both x and u are constrained, with u subject to $U = \{u_k | H_u u_k \leq \mathbf{1}\}, H_u \in \mathbb{R}^{n_{pu} \times n_u}$. x is constrained to X where $X = \{x_k | H_x x_k \leq \mathbf{1}\}, H_x \in \mathbb{R}^{n_{px} \times n_x}$.

Definition 1. Given a function $V : \mathbb{R}^{n_x} \to \mathbb{R}$, the level set of $V(x)$ for a scalar α is the set $S_\alpha = \{x | V(x) \leq \alpha\}$.

Proposition 1. *Consider a function $V(x) : \mathbb{R}^{n_x} \to \mathbb{R}$ satisfying the following properties:*

A1 positive definite, with $V(0) = 0$

A2 continuous,

A3 radially unbounded

Then

1. *All level sets $S_\alpha$exist and are bounded for*$0 \leq \alpha < \infty$
2. *If* $\beta < \alpha, S_\beta \subset S_\alpha$

Proof is in [1].

Definition 2. Consider a continuous and radially unbounded function $V : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$. A level set S_α is controlled γ-contractive with respect to (3.2b) for a given $\gamma \in (0, 1)$, if $\forall x_k \in S_\alpha, \exists u \in U$ such that $x_{k+1} \in S_{\gamma\alpha}$.

Now, consider the following optimization-based problem formulation:

$$\min_{u_k, x_{k+1}} \frac{1}{2} x_{k+1}^T Q x_{k+1} + \frac{1}{2} u_k^T R u_k \tag{3.2a}$$

subject to

$$x_{k+1} = Ax_k + Bu_k \tag{3.2b}$$

$$H_u u_k \leq \mathbf{1} \tag{3.2c}$$

$$V(x_{k+1}) \leq \gamma V(x_k) \tag{3.2d}$$

Q and R are weighting matrices for the states and input, respectively.
Proposition 2. *Let $V(x)$ be a function fulfilling the properties of Proposition 1, and let $V(x) = \alpha$. Then, if*

1. *the corresponding level set S_α is controlled γ-contractive, and*
2. $S_\alpha \subseteq X$

the control action obtained as a solution of (3.2) *guarantees an exponentially stability of the closed loop which in addition fulfills input and state constraints over S_α.*

It is clear from Proposition 2 that the function $V(x)$ works as Lyapunov function for the system (3.2) inside the set $S_{\bar{\alpha}}$ where $\bar{\alpha}$ is the maximum value of α giving $S_\alpha \subseteq X$.

3.2 Ellipsoidal Contractive Sets

Ellipsoidal contractive sets are contractive sets corresponding to a quadratic Lyapunov function. Finding ellipsoidal contractive sets serves an important purpose for this method: they are the starting point for finding larger contractive sets. The approach for finding the largest constrained ellipsoidal set is described in [15].

Consider the system described in the previous section. The largest ellipsoid fulfilling the contractive, state, and input constraints is given as $\Omega = \{x_k \in \mathbb{R}^{n_x} | x_k^T P^{-1} x_k \leq 1\}$.

The set should be found adhering to a linear feedback law $u_k = u = Kx_k$, which is done by maximizing the logarithm of the determinant of P, subject to the input, state, and contraction constraints. This will yield an optimal P and K. The problem formulation, describing the input constraints as $-u_{j,max} \leq u_j \leq u_{j,max}$, is:

$$\max_{P,K,u} logdet(P) \tag{3.3a}$$

$$\text{subject to}$$

$$\begin{bmatrix} \gamma P & P(A+BK)^T \\ (A+BK)P & P \end{bmatrix} \geq 0 \tag{3.3b}$$

$$\begin{bmatrix} \mathbf{1} & H_{x,i}P \\ PH_{x,i}^T & P \end{bmatrix} \geq 0, \forall i = 1, \dots, n_c \tag{3.3c}$$

$$\begin{bmatrix} u_{j,max}^2 & K_j P \\ PK_j^T & P \end{bmatrix} \leq 0, \forall j = 1, \dots, n_u \tag{3.3d}$$

Where $\gamma \in (0, 1)$ is the contraction factor, and n_c, n_u are the number of state constraints and inputs, respectively. $H_{x,i}$ and K_j represent the i'th and j'th row of H_x and K respectively. Both P and K are unknown, so the problem formulation (3.3) is bilinear in the terms of those matrices. This can be solved by introducing the new variable $Y = KP$. The values of K and can be extracted from Y later since an optimal P is found.

The largest unconstrained ellipsoid contained within Ω is denoted
$\Omega_{uc} = \{x_k \in \mathbb{R}^{n_x} | x_k^T P_{uc}^{-1} x_k \leq 1\}$. While Ω is found by optimizing over all possible K, K_{uc} is the optimal unconstrained feedback gain found by solving (3.2), and has the form $K_{uc} = -(R + B^T QB)^{-1} B^T QA$ for the linear feedback $u_k = K_{uc} x_k$. Finding the set Ω_{uc} is just modifying (3.3) to include that $\Omega_{uc} \in \Omega$. This is done by modifying the state constraint to be

$$P - P_{uc} \geq 0.$$

The problem formulation then becomes:

$$\max_{P,K,u} logdet(P) \tag{3.4a}$$

$$\text{subject to}$$

$$\begin{bmatrix} \gamma P & P(A+BK)^T \\ (A+BK)P & P \end{bmatrix} \geq 0 \tag{3.4b}$$

$$P - P_{uc} \geq 0 \tag{3.4c}$$

$$\begin{bmatrix} u_{j,max}^2 & K_j P \\ PK_j^T & P \end{bmatrix} \leq 0, \forall j = 1, \dots, n_u. \tag{3.4d}$$

Let $P_1 = P^{-1}$ and $P_0 = P_{uc}^{-1}$. In summary, there are then two ellipsoidal contractive sets associated with the problem (3.2):

- The set $\Omega = \{x_k \in \mathbb{R}^{n_x} | x_k^T P_1 x_k \leq 1\}$, the largest controlled γ-contractive set obtained by optimizing over all linear state feedbacks.

- The set $\Omega_{uc} = \{x_k \in \mathbb{R}^{n_x} | x_k^T P_0 x_k \leq 1\}$, the ellipsoidal set where γ contractiveness is achieved with the linear state feedback $u_k = K_{uc} x_k$ within the set Ω.

3.3 Controller Design for Higher Order Lyapunov Functions

The sets Ω and Ω_{uc} are limited by the quadratic structure of the Lyapunov function. Although the quadratic function has the inherent trait of being convex, a higher order Lyapunov function has the benefit of yielding more complex shapes. This allows an increase in the size of the contractive set. [1] now focuses on increasing the contractive set by allowing for higher order Lyapunov functions.

3.3.1 Approximating the Optimal Solution

The problem is reformulated to better allow the input to have a higher order dependency on the state. x_{k+1} is removed from the problem formulation using (3.2b), and the problem is now formulated as $\mathcal{P}_u$:

$$\min_{uk} \frac{1}{2} u_k^T H u_k + x_k^T F u_k \tag{3.5a}$$

subject to

$$H_u u_k \leq \mathbf{1} \tag{3.5b}$$

$$V(x_{k+1}) \leq \gamma V(x_k) \tag{3.5c}$$

Here $H = (B^T Q B + R), F = A^T Q B$. The above problem has Lagrangian

$$\mathcal{L}(u_k) = \frac{1}{2} u_k^T H u_k + x_k^T F u_k + \lambda_u^T (H_u u_k - \mathbf{1}) + \lambda_q (V(x_{k+1}) - \gamma V(x_k)) \tag{3.6}$$

with the corresponding KKT-conditions:

$$H u_k + F^T x_k + H_u^T \lambda_u + V_{k+1}^1(u_k)\lambda_q = 0 \tag{3.7a}$$

$$H_u u_k - \mathbf{1} \leq 0 \tag{3.7b}$$

$$V(x_{k+1}) - \gamma V(x_k) \leq 0 \tag{3.7c}$$

$$\lambda_u \geq 0 \tag{3.7d}$$

$$\lambda_q \geq 0 \tag{3.7e}$$

$$\lambda_u^T (H_u u_k - \mathbf{1}) + \lambda_q (V(x_{k+1}) - \gamma V(x_k)) = 0 \tag{3.7f}$$

$V^1_{k+1}(u_k)$ is the first derivative of $V(x_{k+1})$ with respect to u_k.
Next, consider the optimization problem $\mathcal{P}_c$:

$$\min_{c,u_k,\lambda_u,\lambda_q} \quad c \tag{3.8a}$$

subject to (3.7a)-(3.7e) and

$$-\lambda_u^T(H_u u_k - \mathbf{1}) - \lambda_q(V(x_{k+1}) - \gamma V(x_k)) \leq c \tag{3.8b}$$

The solution to problem $\mathcal{P}_c$ is denoted c^*, u_k^*, λ_u^*, λ_q^*. Clearly, if $c^* = 0$, then u_k^* minimizes the problem $\mathcal{P}_u$, as that would fulfill the equality constraint (3.7f) and the other constraints are shared. The constraints (3.7a)-(3.7e) clearly prohibit $c^* < 0$. Since $\mathcal{P}_c$has the same constraints as $\mathcal{P}_u$, any feasible solution to $\mathcal{P}_c$is also feasible for $\mathcal{P}_u$. u_k^* will, as long as $c^* \neq 0$, be a suboptimal solution to $\mathcal{P}_u$.
A new definition:

$$J(u_k^*(c)) = \frac{1}{2}(u_k^*)^T H u_k^* + x_k^T F u_K^* \tag{3.9}$$

is the value of $\mathcal{P}_u$with the value u_k^* from the solution of problem $\mathcal{P}_c$. Correspondingly, $J(u_k^*(0))$ denotes the optimal value of u_k for the problem $\mathcal{P}_u$.

Lemma 1. *Consider an optimal solution ($c^*, u_k^*, \lambda_u^*, \lambda_q^*$) to $\mathcal{P}_c$, with $c^* > 0$. Then u_k^* is a suboptimal solution to $\mathcal{P}_u$, with $J(u_k^*(c)) - J(u_k^*(0)) < c$.*

Proof. This proof follows the approach in [16].
For any feasible u_k

$$\begin{aligned} \frac{1}{2}u_k^T H u_k + x_k^T F u_k &\geq \frac{1}{2}u_k^T H u_k + x_k^T F u_k \\ &+ \begin{bmatrix}\lambda_u^* \\ \lambda_q^*\end{bmatrix}^T \begin{bmatrix} H_u u_k - \mathbf{1} \\ V(x_{k+1}) - \gamma V(x_k)\end{bmatrix} = M(u_k). \end{aligned} \tag{3.10}$$

Next, minimize both sides subject to constraints (3.7b) and (3.7c). Thus

$$J(u_k^*(0)) \geq \min_{\text{(3.7b),(3.7c)}} M(u_k) \geq \min_{u_k \in \mathbb{R}^{n_u}} M(u_k). \tag{3.11}$$

$M(u_k)$ is clearly the Lagrangian equation (3.6) of $\mathcal{P}_u$, where $\lambda_u = \lambda_u^*, \lambda_q = \lambda_q^*$. The unconstrained minimization of $M(u_k)$ yields (3.7a), again with $\lambda_u = \lambda_u^*, \lambda_q = \lambda_q^*$. Minimizing $M(u_k)$ unconstrained will thus yield $u_k = u_k^*$, giving

$$J(u_k^*(0)) \geq M(u_k^*)$$

Multiplying the above inequality by -1 and adding $J(u_k^*(c))$ to both sides shows that

$$J(u_k^*(c)) - J(u_k^*(0)) \leq c^* \tag{3.12}$$

□

In clarifying terms: solving the minimization problem $\mathcal{P}_c$yields a suboptimal u_k^* for $\mathcal{P}_u$that is at max c^* from the optimum. This is done to approximate a solution to $\mathcal{P}_u$ circumventing the equality constraint (3.7f).

3.3.2 Problem Reformulation

A problem with the formulation $\mathcal{P}_c$arises as x_{k+1} is substituted out. With u_k expressed as a polynomial function of x_k, $V(x_{k+1})$ will have a higher order dependence on the polynomial coefficients of u_k. This will later cause problems with the controller design formulation, hence the system equation is reintroduced to the problem formulation.
The new starting point is the original formulation (2.1), which has KKT conditions:

$$Ru_k - B^T\lambda_e + H^T\lambda_u = 0 \tag{3.13a}$$
$$Qx_{k+1} + \lambda_e + \lambda_q\nabla V(x_{k+1}) = 0 \tag{3.13b}$$
$$x_{k+1} - Ax_k - Bu_k = 0 \tag{3.13c}$$
$$H_u u_k - \mathbf{1} \leq 0 \tag{3.13d}$$
$$V(x_{k+1}) - \gamma V(x_k) \leq 0 \tag{3.13e}$$
$$\lambda_u \geq 0 \tag{3.13f}$$
$$\lambda_q \geq 0 \tag{3.13g}$$
$$-\lambda_u^T(H_u u_k - \mathbf{1}) - \lambda_q(V(x_{k+1}) - \gamma V(x_k)) = 0 \tag{3.13h}$$

The operator ∇ is the partial derivative with respect to x_{k+1}. The complimentarity constraint is again relaxed. Solutions with a commensurate relaxation are sought, where u_k is sought as a function of only the present state x_k. The model equations are added as an extra term in all other constraints containing both x_k and x_{k+1}. This modification is justified by the property that the system equations are always fulfilled. The new formulation:

$$\min c \tag{3.14a}$$
$$\text{subject to (3.13d), (3.13f), (3.13g), and}$$
$$Ru_k - B^T\lambda_e + H^T\lambda_u + \mu_1^T(x_{k+1} - Ax_k - Bu_k) = 0 \tag{3.14b}$$
$$Qx_{k+1} + \lambda_e + \lambda_q\nabla V(x_{k+1}) + \mu_2^T(x_{k+1} - Ax_k - Bu_k) = 0 \tag{3.14c}$$
$$V(x_{k+1}) - \gamma V(x_k) + \mu_3^T(x_{k+1} - Ax_k - Bu_k) \leq 0 \tag{3.14d}$$
$$-\lambda_u^T(H_u u_k - \mathbf{1}) - \lambda_q(V(x_{k+1}) - \gamma V(x_k)) + \mu_4^T(x_{k+1} - Ax_k - Bu_k) \leq c \tag{3.14e}$$

The multipliers $\mu_1, \mu_2, \mu_3, \mu_4$ as well as λ_e are polynomial functions of x_{k+1}, x_k with no positivity constraint. λ_q and λ_u are also polynomials in the current and next state, but with the explicitly stated positivity constraints.

The ultimate task of this procedure is to evaluate the input online, hence u_k con only be a function of x_k and not x_{k+1}. The reformulated problem enforces this. The multipliers μ_i are optimized to cancel out the terms including x_{k+1}. In addition, they ensure that the equations they are inserted in hold on the manifold of system trajectories (where the system equation (3.1) is fulfilled).

The degrees of freedom in this optimization are the polynomial coefficients of the multipliers, the feedback function, and the Lyapunov function.

The controller design attempts to find a feasible and approximately optimal solution to (3.2) for the set

$$S = \{x|V(x) \leq 1\}$$

Beginning with the set $S = \Omega$ that has the corresponding Lyapunov function $V(x) = x^T P_1 x$, the set is iteratively enlarged, helped by increasing the allowed polynomial degree of $V(x)$. The optimal controller for the set Ω_{uc} is already known to be $K_{uc}x_k$, and noting that $\Omega uc \subseteq S$ it is clear that the controller design can be split in two parts. Inside the set Ω_{uc} the controller $K_{uc}x_k$ is used, and outside of Ω_{uc} but inside S another controller is found through optimization. Define this new set as:

$$S_C = \{x_k|p(x_k) > 0\} \tag{3.15}$$

where

$$p(x_k) = -(1 - V(x_k))(1 - x_k^T P_0 x_k) \tag{3.16}$$

Using the S-procedure as described in section 2.2.1, the constraints of (3.14) can be enforced in the region where $p(x_k)$ is positive. The complimentarity constraints (3.14e) can be relaxed more when the state is far from the origin. This is done by changing the relaxation c to $cx_k^T x_k$.
The state constraints are not yet a part of the problem formulation. They are given by $H_k x_k \leq \mathbf{1}$, which can be described by the intersection of the n_c=number of constraints on x_k unbounded sets $C_r = \{x_k|H_{x,r} \leq 1\}$ where $H_{x,r}$ is the r'th row of H_x. The state constraints then become:

$$(1 - H_{x,r}x_k) - \sigma_r(x_k)(1 - V_j(x_k)) \geq 0, r = 1, \dots, p_x \tag{3.17}$$

The iterative increase of the set S must be added to the formulation as well. The property $S_{j-1} \subseteq S_j$ is fulfilled provided:

$$(1 - V_j(x_k)) - s_6(x_k)(1 - V_{j-1}(x : k)) \geq 0 \tag{3.18}$$

The full problem formulation giving an approximately optimal solution to $\mathcal{P}_u$ then becomes:

$$\min c \tag{3.19a}$$

$$\text{subject to}$$

$$Ru_k - B^T\lambda_e + H_u^T\lambda_u + \mu_1^T(x_{k+1} - Ax_k - Bu_k) = 0 \tag{3.19b}$$

$$Qx_{k+1} + \lambda_e + \lambda_q \nabla V(x_{k+1}) + \mu_2^T(x_{k+1} - Ax_k - Bu_k) = 0 \tag{3.19c}$$

$$H_u u_k - \mathbf{1} + s_1(x_k)p(x_k) \leq 0 \tag{3.19d}$$

$$V(x_{k+1}) - \gamma V(x_k) + \mu_3^T(x_{k+1} - Ax_k - Bu_k) + s_2(x_k, x_{k+1})p(x_k) \leq 0 \tag{3.19e}$$

$$\lambda_u - s_3(x_k)p(x_k) \geq 0 \tag{3.19f}$$

$$\lambda_q - s_4(x_k)p(x_k) \geq 0 \tag{3.19g}$$

$$\begin{aligned} -\lambda_u^T(H_u u_k - \mathbf{1}) - \lambda_q(V(x_{k+1}) - \gamma V(x_k)) + \\ \mu_4^T(x_{k+1} - Ax_k - Bu_k) + s_5(x_k, x_{k+1})p(x_k) \leq c x_k^T x_k \end{aligned} \tag{3.19h}$$

$$(1 - V_j(x_k)) - s_6(x_k)(1 - V_{j-1}(x_k)) \geq 0 \tag{3.19i}$$

$$(1 - H_{x,r}x_k) - \sigma_r(x_k)(1 - V_{j-1}(x_k)) \geq 0, r = 1, \dots, n_c \tag{3.19j}$$

where $s_{1,\dots,6}$ and σ_r are SOS-polynomials.

3.3.3 Solving the SOS Problem

The problem (3.19) can be solved using SOS programming. All except the equality constraints (3.19b) and (3.19c) are of SOS type. Note, however, that the problem is bilinear in several terms, so a direct optimization would be highly non-convex and difficult to solve directly. A common approach to solving bilinear problems is dividing the problem into several linear problems by keeping some of the optimization variables constant while optimizing over the other. This method requires some initial values for the variabes initially kept constant. For this problem, the procedure chosen is:

- First optimize $\lambda_u, \lambda_q, \lambda_e, s_i,$ and σ_r with given $V(x_k) = x^T P_1 x_k$ and u_k.
- Then optimize $V(x_k)$ and u_k with $\lambda_u, \lambda_q, \lambda_e, s_i,$ and σ_r from the previous step.

For subsequent iterations, the newest values of the coefficients are used.
To enforce that the set S increases from one iteration to the next, some points outside the current set are chosen. In the next iteration, the Lyapunov function is further constrained to include the points, guaranteeing the enlargement of the set. A larger set will eventually be unobtainable. Then it is possible to increase the polynomial degree of $V(x_k)$ or u_k, however this will always result in larger systems with increasingly longer computational times.

3.4 Problem Solution Algorithm

The sequence for achieving a larger contractive set is straightforward. First, the sets Ω and Ω_{uc} are found using the method described in Section 3.2, which always exist for the linear system framework. Next, a level set S larger than Ω should be found. The only part of the set that is of interest is where $p(x)$ is positive. When the system trajectory is inside the set Ω_{uc}, the control law should switch to the controller K_{uc}.

Algorithm 1: Algorithm to obtain a larger contractive set

```
Input: A contractive ellipsoid $\Omega$ with control law $u_k$. Maximum allowed degree ($x_{deg}$)
       for the Lyapunov function, maximum allowed degree ($u_{deg}$) for the control law,
       and maximum acceptable measure of sub-optimality ($c_{max}$). A small number
       $increment$ used to specify a point outside of $S_{j-1}$.
Output: A larger contractive set of degree $\leq x_{deg}$ with control law of degree $\leq u_{deg}$.
1  Set $j = 0, S_j = \Omega$ such that $S_j = \{x_k | V(x_k) \leq 1\}$ and set solution = feasible.
2  while solution is feasible do
3      Set $j = j + 1$
4      Find boundary points of the set $S_{j-1}$ by solving for $V(x_k) == 1$ along rays in
         directions defined by vectors from the origin to points uniformly distributed on the
         $n_x$-dimensional unit sphere.
5      Solve the SOS problem (4.5) by optimizing for $\lambda_u, \lambda_q, \lambda_e, s_i, \mu_s$ and $\sigma_r$ while
         keeping $V(x)$ and $u_k$ fixed.
6      Check which point contracts the most by applying the control law $u_k$. Select that
         point $point$.
7      Find a new point outside $S_{j-1}$ by incrementing on $point$;
         $point_{new} = point + point \times increment$.
8      Solve the SOS problem (4.5) by optimizing for $V(x)$ and $u_k$ while keeping
         $\lambda_u, \lambda_q, \lambda_e, s_i, \mu_s$ and $\sigma_r$ fixed and ensuring that $point_{new}$ is in $S_j$.
9      if solution is feasible and $c \leq c_{max}$ then
10         Update $V(x_k), u_k, \lambda_u, \lambda_q, \lambda_e, s_i, \mu_s$ and $\sigma_r$.
11     end
12 end
```

Algorithm 1 outlines the implementation of the method. The choice of the point $point_{new}$ will be discussed in Section 5.3. The rest is quite straightforward. Initially, the set S is chosen equal to Ω before it is expanded, with the corresponding input. As discussed in the previous section, the bilinearity of the problem is circumvented by iteratively optimizing over the multipliers, then the LF and input.

Chapter 4

A Discussion on the Solution of Munir et.al.(2018) and Improvements

4.1 Placeholder

The general optimization problem discussed in [1], and which will be modified in this chapter, is one of the following form:

$$\min_{u_k, x_{k+1}} \frac{1}{2} x_{k+1}^T Q x_{k+1} + \frac{1}{2} u_k^T R u_k \tag{4.1a}$$

subject to

$$x_{k+1} = A x_k + B u_k \tag{4.1b}$$

$$H_u u_k \leq \mathbf{1} \tag{4.1c}$$

$$V(x_{k+1}) \leq \gamma V(x_k) \tag{4.1d}$$

The system (4.1) is described with the following Karush-Kuhn-Tucker(KKT) conditions:

$$H u_k + F^T x_k + H_u^T \lambda_u + V_{k+1}^1(u_k)\lambda_q = 0 \tag{4.2a}$$

$$H_u u_k - \mathbf{1} \leq 0 \tag{4.2b}$$

$$V(x_{k+1}) - \gamma V(x_k) \leq 0 \tag{4.2c}$$

$$\lambda_q \geq 0 \tag{4.2d}$$

$$\lambda_u \geq 0 \tag{4.2e}$$

$$\lambda_u^T (H_u u_k - \mathbf{1}) + \lambda_q V(x_{k+1}) - \gamma V(x_k) = 0 \tag{4.2f}$$

However, the KKT-conditions (4.2) are only necessary conditions for a convex formulation. With the current problem formulation, the distance from the optimum cannot be guaranteed without somehow verifying the convexity of the entire feasible region. There are no hints in [1] that this has been attempted, and thus the results of [1] are partially invalidated, in particular the value of c. In other words, the contractive set S may yield an input u_k that is far from optimal. This may be the source of the low value $4.54 \cdot 10^{-12}$ found for c presented in the article, which could be invalid.

Figure 4.1 shows the Lyapunov function after two iterations of the current formulation, without an explicit constraint on the convexity of the Lyapunov function. Clearly, the set $V(x_k) \leq 1$ is non-convex. This shows that the current formulation can result in non-convex problems, and thus no guarantee for the distance to optimum. Note also that the set violates the state constraints. This will be discussed further in chapter 6.

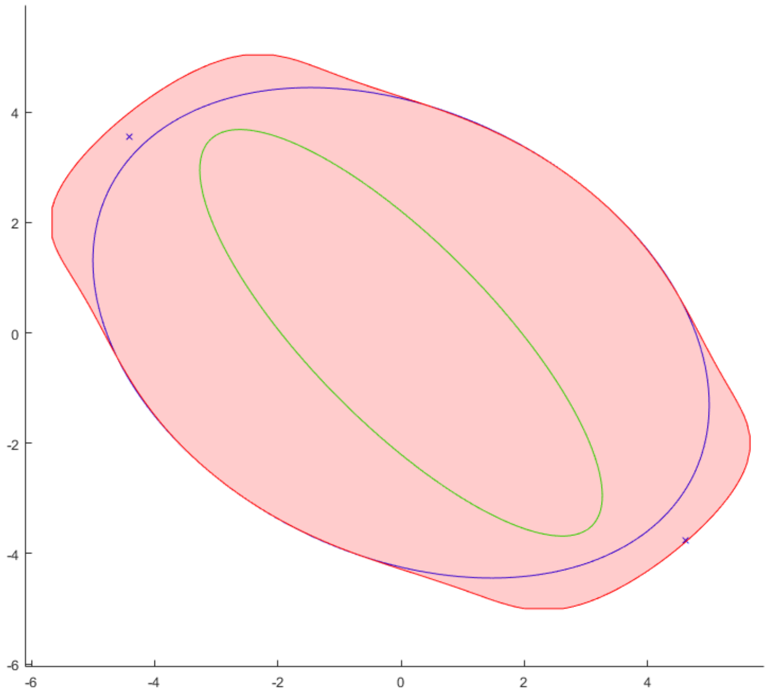

Figure 4.1: The Lyapunov function found by using the unmodified method of [1] in pink

4.2 Completing the Formulation

Not much needs to be added to formulation for the KKT-conditions to be sufficient and not necessary. The currently missing constraint is the convexity of the Lyapunov function $V(x_k)$. This is obviously not necessary when the Lyapunov function is quadratic, but for higher degree functions it is needed. The convexity of a polynomial is equivalent with the positive semi-definiteness of its Hessian matrix [17, p. 7]. Formulated as an inequality constraint:

$$\nabla^2 V(x_k) \geq 0 \tag{4.3}$$

Changing the formulation to an SOS problem:

$$z^T(\nabla^2 V(x_k))z \geq 0 \tag{4.4}$$

where $z \in \mathbb{R}^{n_x}$ is a variable not dependent on x. If necessary, the constraint can be relaxed to only be positive outside of $p(x)$, however global convexity should be sufficient. Since the constraint is an SOS constraint it is also convex.

The constraint (4.4) can now be added to (4.1), and, following the same approach as in section 3.3, the complete problem formulation now becomes:

$$\min c \tag{4.5a}$$

subject to

$$Ru_k - B^T\lambda_e + H_u^T\lambda_u + \mu_1^T(x_{k+1} - Ax_k - Bu_k) = 0 \tag{4.5b}$$

$$Qx_{k+1} + \lambda_e + \lambda_q\nabla V(x_{k+1}) + \mu_2^T(x_{k+1} - Ax_k - Bu_k) = 0 \tag{4.5c}$$

$$H_u u_k - \mathbf{1} + s_1(x_k)p(x_k) \leq 0 \tag{4.5d}$$

$$V(x_{k+1}) - \gamma V(x_k) + \mu_3^T(x_{k+1} - Ax_k - Bu_k) + s_2(x_k, x_{k+1})p(x_k) \leq 0 \tag{4.5e}$$

$$\lambda_u - s_3(x_k)p(x_k) \geq 0 \tag{4.5f}$$

$$\lambda_q - s_4(x_k)p(x_k) \geq 0 \tag{4.5g}$$

$$\begin{aligned} -\lambda_u^T(H_u u_k - \mathbf{1}) - \lambda_q(V(x_{k+1}) - \gamma V(x_k)) + \\ \mu_4^T(x_{k+1} - Ax_k - Bu_k) + s_5(x_k, x_{k+1})p(x_k) \leq c x_k^T x_k \end{aligned} \tag{4.5h}$$

$$(1 - V_j(x_k)) - s_6(x_k)(1 - V_{j-1}(x_k)) \geq 0 \tag{4.5i}$$

$$(1 - H_{x,r}x_k) - \sigma_r(x_k)(1 - V_{j-1}(x_k)) \geq 0, r = 1, \ldots, n_c \tag{4.5j}$$

$$z^T(\nabla^2 V(x_k))z \geq 0 \tag{4.5k}$$

This problem formulation guarantees the convexity of the contraction constraint.

Chapter 5

Methodology

The method presented in this thesis is an improvement on the method in [1], and as such builds on the computer code written in relation to that paper. Coding, optimization, and simulations are all run in MATLAB, utilizing the YALMIP [2] toolbox for SOS programming and MOSEK as an SDP-solver. This implementation follows the same approach for solving the bilinear problem as in [1], namely by iteratively solving linear sub-problems. From [1]:

- First optimize $\lambda_u, \lambda_q, \lambda_e, s_i$ and σ_r with set $V(x_k)$ and u_k.
- Then optimize $V(x_k)$ and u_k with $\lambda_u, \lambda_q, \lambda_e, s_i$ and σ_r from above, only now with the added constraint (4.4)

The algorithm for finding a new contractive point is the same; search the boundary of the Lyapunov function, find the point that contracts the most, and increase in that direction.

5.1 Adding the Convexity of the Lyapunov Function

As discussed in the previous chapter, the new addition to the system is the constraint that the Lyapunov function is convex. It was shown in section 4.2 that this is equivalent to an added inequality constraint

$$z^T(\nabla^2 V(x_k))z \geq 0 \tag{5.1}$$

. This constraint is an SOS constraint, and will be treated as such in the software implementation. z can be defined in YALMIP as an SDP-variable (*sdpvar*) with dimension equal to the state x. The Hessian matrix of the Lyapunov function, $\nabla^2 V(x_k)$, can be found with a simple function in YALMIP, $hessian()$.

The optimization iterates over solving for the multipliers and then the Lyapunov function and input. The new constraint, which is independent of any of the multipliers, then only needs to be added to the LF and input optimization.

5.2 Deciding the Order of the Polynomials

To formulate an SOS-problem in YALMIP, the maximal order of the polynomial to optimize over must be given explicitly. Here, this includes the order of the polynomials $\lambda_e, \lambda_q, \lambda_u, \mu_{1,\dots,4}, s_{1,\dots,6}, \sigma_r, u_k$ and $V(x_k)$. The choice of the order of these polynomials is constrained by the equality constraints (5.2) and (5.3):

$$Ru_k - B^T\lambda_e + H_u^T\lambda_u + \mu_1^T(x_{k+1} - Ax_k - Bu_k) = 0 \tag{5.2}$$

$$Qx_{k+1} + \lambda_e + \lambda_q\nabla V(x_{k+1}) + \mu_2^T(x_{k+1} - Ax_k - Bu_k) = 0 \tag{5.3}$$

For the equalities to hold, the maximum order of the polynomials must be matched between terms for them to cancel out. For example, choosing $deg(V(x_k)) = 6$ puts a lower bound on μ_2 and λ_e for them to match out the term $\lambda_q\nabla V(x_{k+1})$. This will further constrain the degrees of μ_1 and λ_u, and basically propagates throughout the problem equations. The positivity constraints must be upheld as well, so naturally all SOS-variables must be of even degree, as well as λ_q and λ_u. This is important to remember when initializing the system as not allowing these degrees of freedom may cause the problem to be unsolveable.

The size of the monomial vector increases exponentially with the polynomial degree, which again increases the amount of decision variables, greatly increasing runtime. This serves as a limiter on the highest degree of the polynomials. At one point, the optimization will thus fail if the polynomial degree is increased.

5.3 Choosing a New Point to be Included in the Next Level Set

The procedure for choosing a new point to be included in the next iteration's level set is not implemented exactly as described in [1]. The procedure is flawed, and should be reworked on later iterations.
The code can be seen in the attachments, named `Point_Vx_2D_mod.m`.

Algorithm 2 outlines the process for choosing a new point outside the current set. The process guarantees a point within the state constraints, and tuning the parameter $increment$ makes it possible to find larger and larger sets. The basic idea is quite primitive: find the point on the boundary of S that contracts the most, then push that point out by a factor $increment$. If the new point is outside the state boundaries, the point is made void and the next most contractive point is evaluated.

```
Algorithm 2: Algorithm to find a point outside the current level set
Input: A contractive set $S_{j-1} = \{x|V(x) \leq 1\}$ with control law $u = u(x) = K_u x$. The
       degree ($V_{deg}$) for the Lyapunov function. A small number $increment$ used to
       specify a point outside of $S_{j-1}$. The state constraints $H_x x_k \leq 1$
Output: A point $point_{new}$ outside of the set $S_{j-1}$ to be included in the next level set $S_j$
1  Partition the constrained state space uniformly.
2  Find the set of points $x_e$ that satisfy $V(x_e) \approx 1$
3  For the points $x_e$, calculate the next state $x_{e_p}$ using the control law $K_u x_e$:
   $x_{e_p} = Ax_e + BK_u x_e$
4  For the points $x_{e_p}$, calculate the Lyapunov function value $V(x_{e_p})$. Call this set of values
   $V_e$
5  while new point not found do
6      Choose the point $point$ corresponding to the smallest value in $V_e$
7      $point_{new} = point + point \times increment$
8      if $point_{new}$ is within state constraints then
9          new point found
10     else
11         set $V_e(point) = 1000$
12     end
13 end
```

Although the procedure is robust, it is far from optimal. For one, the way of finding the points x_e is now done by creating a grid over the constrained state space and evaluating the Lyapunov function value at each of the points. This is not only time and memory demanding, but the resolution limits the number of points x_e that can be found. The way the state space is gridded makes the more curved areas on the edge of S poorly represented. This is somewhat mitigated by allowing $V(x_e) \approx 1$, although this relaxation again makes the representation of the edge of S worse, as the exact equality is not upheld.

Chapter 6

Results and Discussion

All of the plots shown in this chapter follow a color and shape convention. The x-axis shows the value of state x_1, and the y-axis that of x_2. The largest, ellipsoidal, controlled γ-contractive set with linear feedback for the constrained problem, Ω, is always depicted in blue. The set Ω_{uc} is always in blue. Only the boundaries of these sets are shown, in an effort to make the plots easier to analyze. The set $V(x_{k+1}) \leq 1$ always appears in light red. The point to be included in the next level set $point_{new}$ is shown as a blue X. If there are other uncertainty elements present, they will be described to the best of the author's ability.

This chapter is divided into sections describing the different results, with a discussion of the results and their implications at the end of each section.

6.1 The System

The improved method is only applied to a simple linear system:

$$x_{k+1} = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0.22 \\ 0.2 \end{bmatrix} u_k \tag{6.1}$$

with the constraints $-5 \leq x_{k,j} \leq 5$ for $j = 1, 2$ and $-2 \leq u_k \leq 2$, and contraction factor $\gamma = 0.90$. Note that this is the same system initially considered in [1], which is a conscious choice done to closely inspect the differences between the methods.

Initially, multiple different systems were to be experimented on. The problems faced during the implementation, however, caused only the linear system (6.1) to be considered.

6.2 Adding the Convexity Constraint

6.2.1 Non-Convex Results

One of the chief contributions of this thesis is the addition of the convexity constraint on the Lyapunov function to the problem formulation. Before this addition, the problem would not be guaranteed convex for higher order Lyapunov functions, and thus the KKT-conditions would only be necessary conditions. In Figure 6.1, a result from the incomplete formulation is shown. This is after 2 iterations. By visual inspection alone, the light red level set $V(x_k) \leq 1$ is clearly not convex. After 5 iterations, the results are even less convex, as shown in Figure 6.2. Not only is the level set highly non-convex, but it falls inside the ellipsoid Ω in a few areas as well.

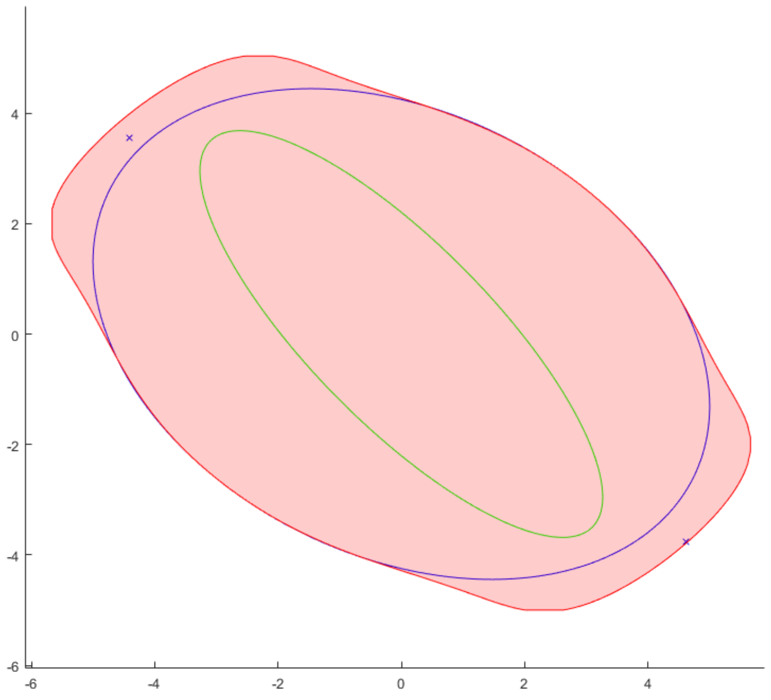

Figure 6.1: Non-convex level set from the old formulation, $j = 2$

Note that both level sets violate the state constraints $-5 \leq x_k \leq 5$ at several points, the plot in Figure 6.2 is just cut off at the edges. This will be further discussed in Section 6.3.

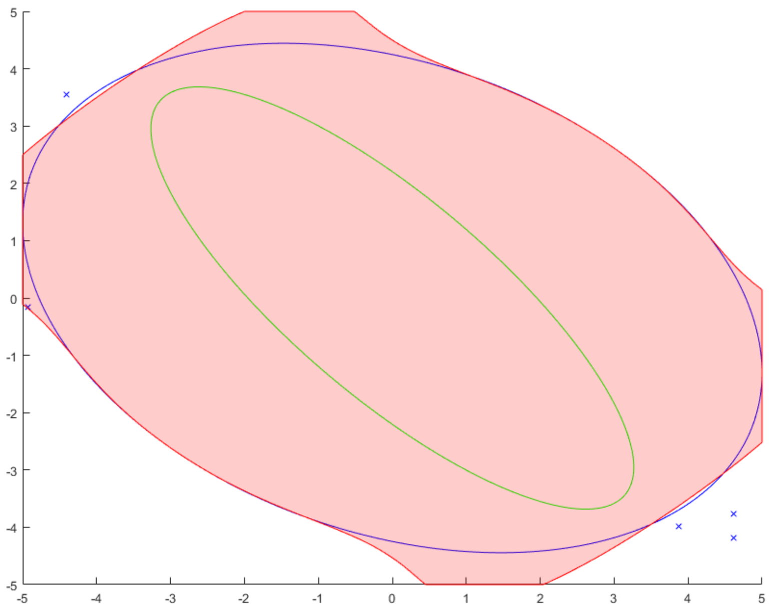

Figure 6.2: Non-convex level set from the old formulation, $j = 5$

The clearly non-convex level sets in Figures 6.1,6.2 prove that the formulation presented in [1] is incomplete.
Next, the convexity constraint on the Lyapunov function is added to the formulation.

6.2.2 Convex results

Figure 6.3 shows the level set after the convexity constraint has been added. The results are promising: the new level set is verifiably convex. The area of the level set is also increased from before the addition. Most important, the optimization problem is now convex.
However, the problem of state constraint violation still persists.

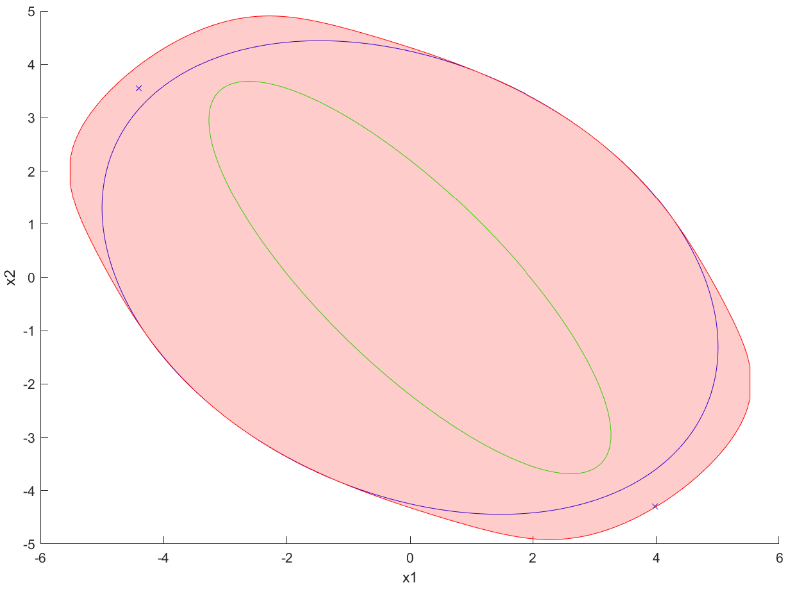

Figure 6.3: Convex level set from the new formulation, $j = 2$

6.2.3 Discussion

These results serve to prove that the incomplete formulation of [1] can yield a non-convex function and hence cannot guarantee the distance from the optimum. The addition of the constraint (4.4) yields a convex LF and hence a convex optimization problem that has guarantee for the distance to the optimum and general optimality. In this aspect, the work on adding the convexity constraint can only be seen as a success.

However, there are larger problems present. The state constraints are violated to a high degree, and the violation increases with each iteration. In the following sections, several methods for fixing this problem will be presented, analyzed, and discussed.

6.3 Violation of the State Constraints

The state constraint violation is a new problem that was not present or discussed in [1]. Although the code used to produce the results in Figures 6.1-6.2 was supplied by the authors of [1], the results produced do not match the results in the article. There are hard violations of the state constraints. It is not clear what is causing these violations. To find the root of the problem, some simple methods for holding the level set within the state constraints were attempted.

6.3.1 Hard Constraint on State Violation

A primitive method for keeping the level set within the state constraints utilize a property of the level set Ω. A constraint $V_j(x_c) \approx 1$ is added to the system, where x_c are the points in the set Ω that touch the state constraints. Due to the nature of the optimization used to find Ω, these points always exist for one of the state variables. The approximate condition is to give the solver some leeway as exact solutions are inaccurate with numerical solvers.

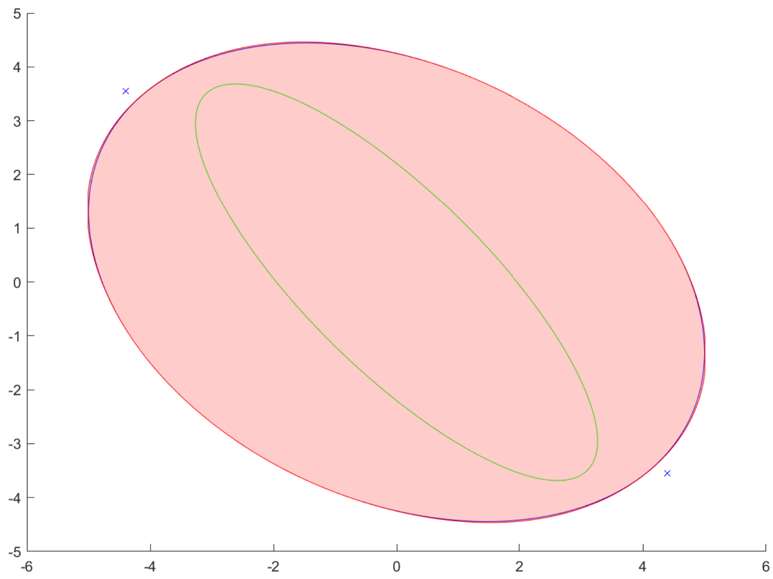

Figure 6.4: New method with hard constraints on state violation

With this added constraint, the set $V_j(x_k) \leq 1$ is guaranteed to not violate the state constraints at the given points. Now, the algorithm should try to expand the set in other directions while keeping the set points constant. Unfortunately, the method did not work, as can be seen in Figure 6.4. The new set is identical to the previous set, with no noticeable volume increase from one iteration to the next. The points sought to be included in the next set, depicted as blue crosses, are not included in the new set either.

Analysis

It is clear that the contractive set is somehow stopped from being expanded without breaking the state constraints. Although the method described above is simplistic, some increase in the volume should be expected when allowing for a Lyapunov function with allowed polynomial degree $deg(V(x_k)) = 8$. This result furthers the notion that the process of solving the problem is flawed.

6.3.2 Alternative Choice of New Point

Another way to examine how the solver works, is trying to choose specific points to be included based on previous results. In [1], results such as the one in in Figure 6.5 are presented.

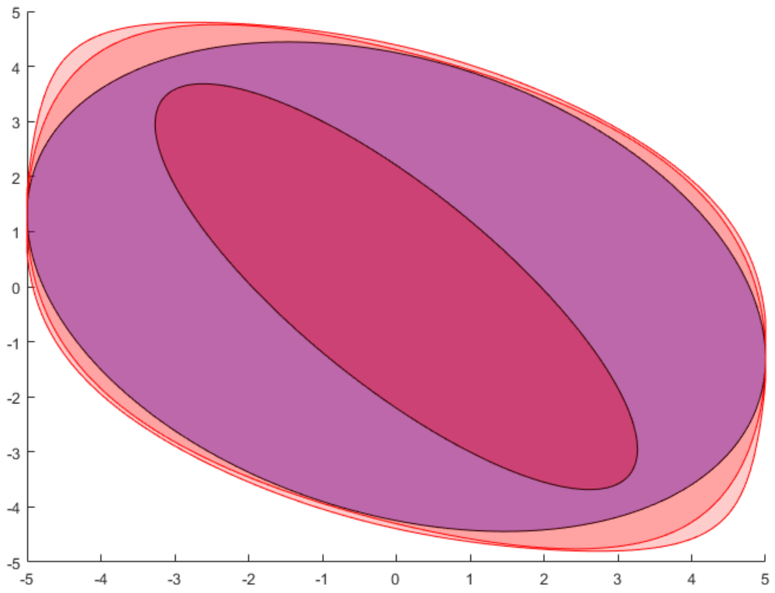

Figure 6.5: Explicit reproduction of the results in [1]

Note that the state constraints are not violated. The level sets found are seemingly expanding towards the upper left and lower right of the constrained state space, most likely because the contraction constraint is easier to fulfill in those directions. Note also that the largest set in light color only touches the set Ω in the points where Ω touches the state constraints. This implies that such a set should be possible to find. However, with the supplied code, explicitly choosing a point near the long edge of the ellipsoid Ω causes the program to crash, not finding a solution. This is without the convexity constraint, so theoretically the software implementations should be nearly identical. Still, there is a large discrepancy in the results, and more methods must be tried.

Lowering the Point Increment

Figure 6.6 shows the level set found when reducing the point increment variable $incr = 0.01$. Even with such a small point increase (the next point to be included is right outside the ellipsoid Ω), the set found still violates the state constraints.

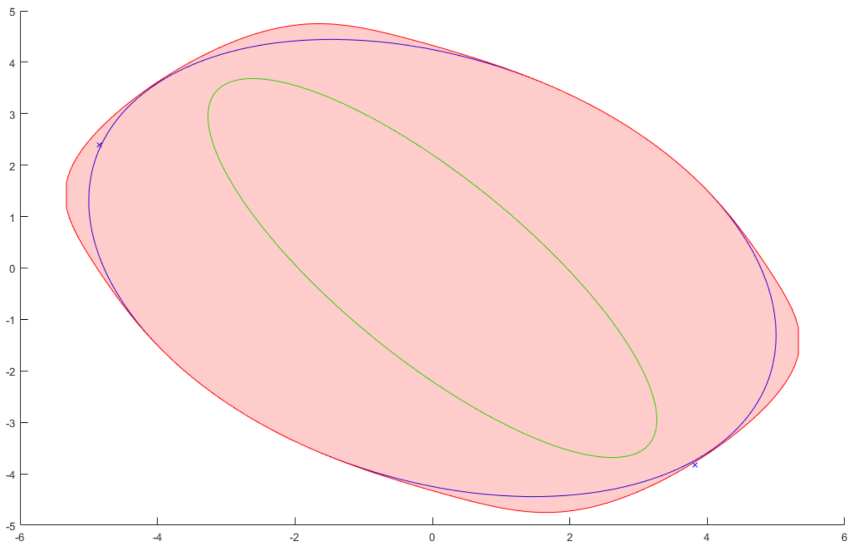

Figure 6.6: Shorter increment $incr = 0.01$

This new plot is interesting, in that it gives some insight in how the set is expanded.

Figure 6.7 shows the directions of expansion of the level set. This trend of how the set is expanded is present in all the other figures, but in Figure 6.7 it is even clearer.
The set is expanded in directions perpendicular to the state constraints, i.e. in direct pos-

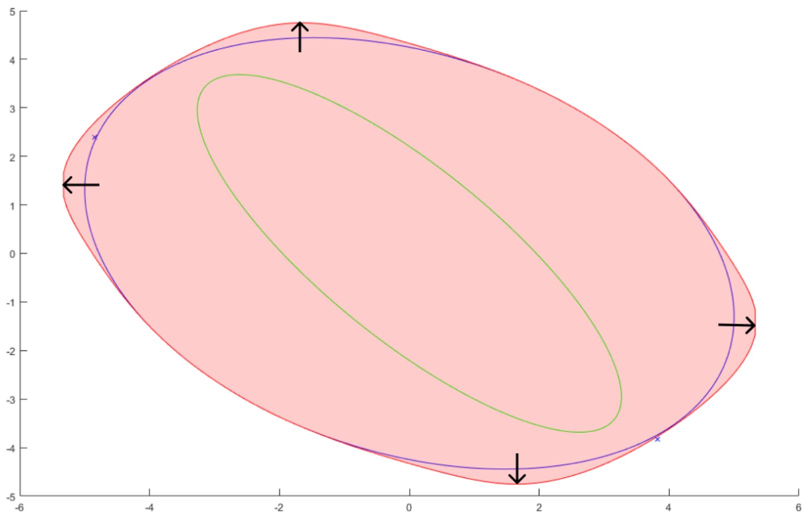

Figure 6.7: The directions in which the Lyapunov function is expanded

itive and negative directions along the state axes. Seemingly, the extreme points of Ω are pushed towards the state boundaries (or over them), while the rest of the set follows most of the curvature of Ω. Comparing this to the results of [1] represented by Figure 6.5, where the set is expanded more in the direction of the corners of the constrained state space. Ultimately, the set S should expand in these directions as well, but it doesn't. In fact, the set actually seems constrained in those directions, as it is tangential to Ω in approximately those points.

The thought that S perhaps was constrained in a way, sparked a new thought: could relaxing any of the constraints provide better results? Another question is which constraint to relax. SOS-optimization will, as discussed in Chapter 2, try to find a PSD matrix Q. Looking at the output from the optimization, and using the validation technique in Section 2.2.2, the matrix Q_{λ_u} associated with the constraint (4.5f) is found to have a negative eigenvalue.
Weirdly enough, it is a constraint associated with the input that is violated. In theory, this should be unproblematic in the first iteration, as a Lagrange multiplier for the set Ω is defined. The poor conditioning on the multiplier leads to another attempt at solving the state constraint violation problem: relaxing the input constraint.

6.3.3 Relaxing the Input Constraint

Figure 6.8 shows the sets obtained after two iterations when the input constraint is changed to $-3 \leq u_k \leq 3$. The effect of the relaxation is quite noticeable, and at first glance the state constraint violation seems to be solved. However, upon closer inspection the level set still violates the state constraints. Still, this counts as progress. The level set is now increasing more in the desired directions, with noticeably lesser constraint violations.

Note that, as a result of increasing the allowed range of u_k, the sets Ω and Ω_{uc} have

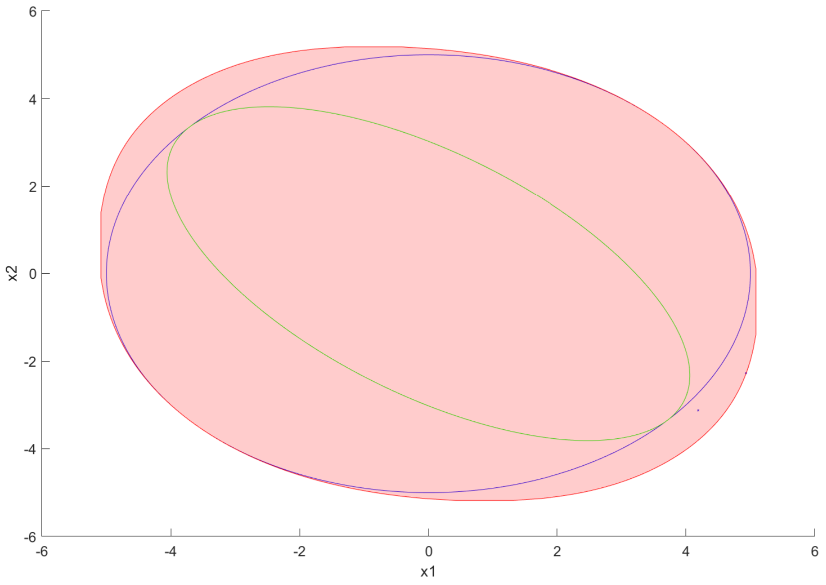

Figure 6.8: Increased input constraint $-3 \leq u_k \leq 3$

increased as well. In fact, Ω now touches all the state constraints, which means that S now violates all state constraints. Still, the shape of the set is closer to the shape shown in Figure 6.5, indicating that something is done right.

In Combination with Hard Constraint on State Violation

Combining the method discussed in Section 6.3.1 with the increased area for u_k yields the set S in Figure 6.9. The combination actually seems to solve the state constraint violation problem. The level set is now fully contained within the state constraints, and there is a noticeable increase in the volume of the level set. The program stops after one iteration, meaning it is not able to find a larger set within the given constraints without increasing the degree of the Lyapunov function or input.

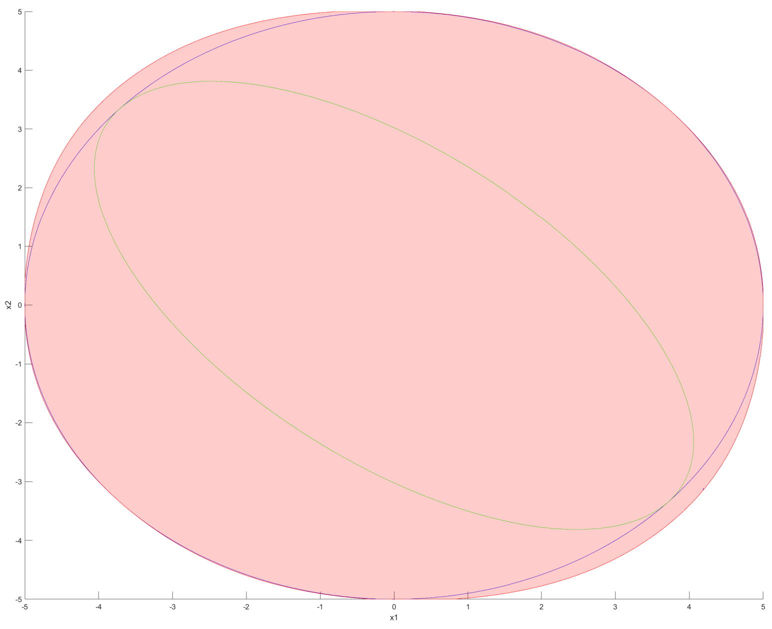

Figure 6.9: Increased input constraint $-3 \leq u_k \leq 3$, hard constraint on state violation

6.3.4 Discussion

The problem of the state constraint violations was, in a way, solved, in that a larger set within the state constraints was found. This, however, came at the cost of a larger allowed input and ad hoc constraints on the Lyapunov function. The application of the final version is now limited to systems that are basically unconstrained in the input, not for all constrained linear systems. In addition, the increase in the set, even though it is present, is not on par with the levels shown in [1], and this for a less constrained system.

6.4 Closing Thoughts

The results show that the convexity constraint was needed, as the earlier formulation could produce non convex level sets. The next step was to expand the formulation to include input affine rational nonlinear dynamics, but the complications of the level set violating constraints halted this progress. Although this seems like an issue with either the solver or a software bug, the root of the problem was not found even after thorough investigation. A different SDP solver, SDPT3[18], was tried. The solver unfortunately provided no better results than MOSEK.

Several different alterations have been attempted, all with moderate success. There are clearly some underlying faults in the implementation. Reasons other than the ones discussed could be simple typos in the supplied software, underlying problems with the problem formulation, or a poor configuration in the settings for MOSEK. The MATLAB code supplied was quite unorganized, so finding a mistake is difficult. Through several explicit modifications, and a small, but significant, change in the example, a valid set S was eventually found. This will be considered a result, even though it is a weak result.

Chapter 7

Conclusion and Further Work

7.1 Conclusion

The result of this thesis is that the addition of the convexity constraint on the Lyapunov function was necessary. The formulation without the constraint was shown to be able to produce a non-convex level set, which would not sufficiently guarantee the distance from the optimum. With the added constraint, the Lyapunov function is convex, which in turn makes the contractivity constraint and the rest of the problem convex. Even if the older formulation had never resulted in a non-convex function, the distance from optimum would never be guaranteed.

As discussed in Section 6.4, there were large problems with state constraint violations when the method was implemented. Although an increased level set was discovered, this came at the cost of explicit value constraints on the Lyapunov function, and an increase in the allowed input, essentially changing the system description to attain a valid result. These modifications were made in an effort to achieve a convex level set within the state constraints with an increased volume from Ω. These modifications, however, should only serve as guidelines for further research, and should not be implemented on physical systems. This is due to their ad hoc procedure and limited areas of application.

In conclusion, the problem formulation in [1] was completed with the addition of a convexity constraint on the Lyapunov function. The supplied software implementation suffered from clear state constraint violations, which were solved by modifying the input constraint and imposing strict constraints on the level set S. These modifications severely limit the application of the controlled contractive set method, and should be further developed and researched.

7.2 Contributions of this Thesis

This thesis provides the addition of the convexity of the contraction constraint to the formulation of [1]. The addition was suggested by one of the co-authors of [1], but was implemented independently. The additions to the problem formulation were implemented independently as well, with the final result of the convex, increased, level set within the state constraints reached independently.

7.3 Further Work

Due to the problems with the supplied software material, some of the points originally in the task description must be passed on to later research. The following problems should be considered by someone seeking to work further on this formulation.

7.3.1 Solving the State Constraint Violations

Although a solution to the problem of state constraint violations was found in Section 6.3.3, the solution is poor and far from general in application. Further research should focus on discovering the underlying reasons for the state constraint violations. A few different approaches are writing the software implementation from scratch, modifying the settings in the SDP solver, and thoroughly search the methodology for mistakes and flaws.

With this problem solved, the application with higher order Lyapunov functions and higher dimensional systems should be considered.

7.3.2 Update the Algorithm for Finding a New Point

The current way of finding a new point to include in the next level set is suboptimal, as discussed in Section 5.3. This development is not critical, but may drastically improve performance, so further research into this area should be considered.

7.3.3 Input Affine Rational Nonlinear Dynamics

A new area that should be further explored is the application of the procedure on nonlinear systems of the form

$$x_{k+1} = f(x_k) + g(x_k)u_k.$$

Where $f(x_k), g(x_k)$ are continuous functions. For a given timestep k, x_{k+1} is an affine, rational function of u_k. In theory, this implies that the problem formulation only needs some few, generalizing changes for application to a nonlinear formulation. Further research ought to include developing and implementing this.

Bibliography

[1] S. Munir, M. Hovd, and S. Olaru. Low complexity constrained control using higher degree lyapunov functions. *Automatica*, 98:215–222, 2018.

[2] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[3] E. D. Andersen and K. D. Andersen. The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. *High Performance Optimization*, pages 197–232, 2000.

[4] David Q. Mayne James B. Rawlings and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*, volume 2. Nob Hill Publishing, LLC, 2017. ISBN 978-0975937730.

[5] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38 (1):3 – 20, 2002. ISSN 0005-1098. doi: https://doi.org/10.1016/S0005-1098(01) 00174-1. URL `http://www.sciencedirect.com/science/article/pii/S0005109801001741`.

[6] Alessandro Alessio and Alberto Bemporad. *A Survey on Explicit Model Predictive Control*, pages 345–369. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-01094-1. doi: 10.1007/978-3-642-01094-1_29. URL `https://doi.org/10.1007/978-3-642-01094-1_29`.

[7] C. E. T. Dórea and J. C. Hennet. (a, b)-invariant polyhedral sets of linear discrete-time systems. *Journal of Optimization Theory and Applications*, 103(3):521–542, Dec 1999. ISSN 1573-2878. doi: 10.1023/A:1021727806358. URL `https://doi.org/10.1023/A:1021727806358`.

[8] Morten Hovd, Sorin Olaru, and Georges Bitsoris. Low complexity constraint control using contractive sets. 2014.

[9] Pablo A. Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry*

Methods in Robustness and Optimization. PhD thesis, California Institute of Technology, Pasadena, California, 5 2000.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

[11] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

[12] J. Löfberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions on Automatic Control*, 54(5):1007–1011, May 2009. ISSN 0018-9286. doi: 10.1109/TAC.2009.2017144.

[13] G. Valmorbida S. Prajna P. Seiler A. Papachristodoulou, J. Anderson and P. A. Parrilo. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*. `http://arxiv.org/abs/1310.4716`, 2013. Available from `http://www.eng.ox.ac.uk/control/sostools`, `http://www.cds.caltech.edu/sostools` and `http://www.mit.edu/~parrilo/sostools`.

[14] Hassan K Khalil. *Nonlinear systems*, volume 3. 2002.

[15] H. . Nguyen. *Constrained control of uncertain, time-varying, discrete-time systems: An interpolation-based approach*, volume 451 of *Lecture Notes in Control and Information Sciences*. 2014. URL `www.scopus.com`. Cited By :11.

[16] Yasuaki Oishi. Direct design of a polynomial model predictive controller. volume 7, pages 633–638, 06 2012. ISBN 9783902823038. doi: 10.3182/20120620-3-DK-2025.00174.

[17] Georgina Hall. *OPTIMIZATION OVER NONNEGATIVE AND CONVEX POLYNOMIALS WITH AND WITHOUT SEMIDEFINITE PROGRAMMING*. PhD thesis, Princeton University, 6 2018.

[18] M.J. Todd K.C. Toh and R.H. Tutuncu. Sdpt3 — a matlab software package for semidefinite programming. *Optimization Methods and Software*, 11:545–581, 1999.

NTNU
Norwegian University of
Science and Technology