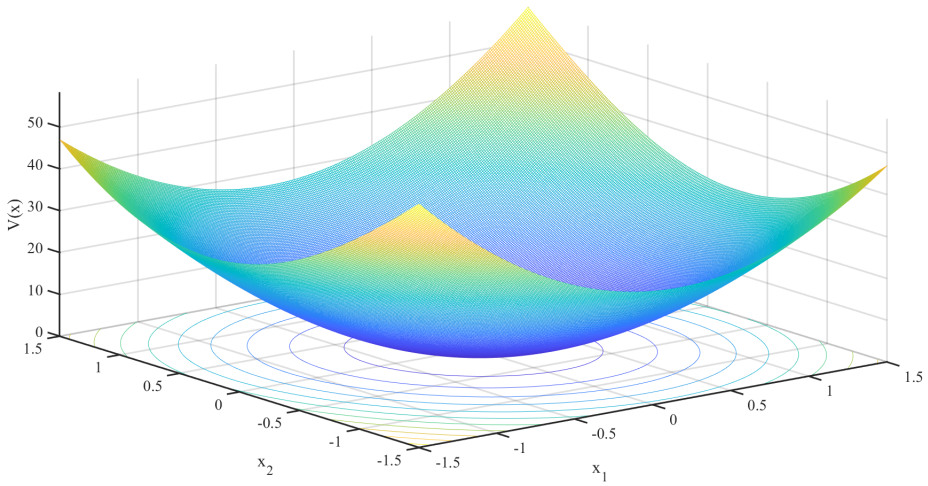
Synthesis of Non-linear Control

Using Sum of Squares and Semidefinite Programming



Master's Thesis

Rahul Misra

Aalborg University
Control and Automation

AALBORG UNIVERSITY
STUDENT REPORT

Preface

This Master thesis documents the work done by me at the Control and Automation department at Aalborg University during the time period September, 2018 - June, 2019. The theme for the project was Control of Complex systems. I would like to thank my supervisors Prof. Rafael Wisniewski and Postdoc Özkan Karabacak for their guidance and weekly meetings throughout the project duration. I would also like to thank Postdoc Ayesgul for reviewing my project and presentation. Finally, I would like to thanks my friends and my family for their support.

This report has been typeset using LaTeX. Since, a variety of dynamic models have been used in this work, maintaining a consistent notation is difficult, despite the author's best attempt and the reader should consider the context when in doubt. Further, a nomenclature is provided for the reader's reference. Vectors and quaternions are indicated by boldface ex. $\mathbf{q}$. The citations are indicated by square brackets [] which encloses an abbreviation of the author(s) and the year of publication ex. [Las10]. Relevant details about the citation can be found in the bibliography. Reference to equations and figures are indicated by round brackets () which encloses the equation number or the figure number. Figures can be diagrams, flowcharts and graphs.Figures have been generated using MATLAB, Microsoft Office or TikZ package in LaTeX. Each chapter has a summary at the beginning which summarizes the chapter with conclusions.

The thesis was completed on June 6, 2019. The total number of pages are 79. The content of this report is freely available but publication (with reference) may only be pursued due to agreement with the author

Rahul Misra
<rmisra17@student.aau.dk>

Abstract

This report documents synthesis of control law using Lyapunov functions for a range of nonlinear systems. Finding Lyapunov functions for a nonlinear system is a non-trivial task and to overcome this challenge, we have considered the Lyapunov function to be a Sum of Squares (SOS) polynomial. Using this approach, the problem of finding a suitable Lyapunov function is posed as an Semidefinite Program (SDP) which can be solved using a suitable solver. In this project, we have used results from Algebraic Geometry specifically Putinar's Positivstellensatz so as to restrict the search of Lyapunov function to a semialgebraic set.

A major drawback in using this approach is scalability to bigger problems. As the number of states increase, the size of the SDP increases and the computational time is polynomial in number of states n i.e. $\mathcal{O}(n^d)$ [Las06] (with d being the maximum degree of Lyapunov function). Thus, practical implementation of this approach becomes difficult and we have used a sparse version of Putinar's Positivstellensatz so as to overcome the aforementioned challenge in practical implementation. We begin by finding Lyapunov function for Van der Pol's model of a nonlinear oscillator. Thereafter, we have considered complex systems such as a wind turbine and Ørsted Satellite and found Lyapunov function using this approach. Finally, based on the obtained Lyapunov function we have attempted, methods for Nonlinear control design such as Sontag's formula and Lyapunov Redesign.

Keywords *Nonlinear systems, Lyapunov function, Sum of Squares, Semidefinite Programming, Semialgebraic sets, Putinar's Positivstellensatz, Sparse Putinar's Positivstellensatz, computational complexity*

Contents

Nomenclature

Acronyms

LEO	Low Earth Orbit Satellite
LMI	Linear Matrix Inequality
LP	Linear Programming
MATLAB	Matrix Laboratory by Mathworks Inc.
MOSEK	Optimization toolbox by MOSEK Aps for MATLAB
ROA	Region of Attraction
SDP	Semidefinite Programming
SeDuMi	Self Dual Minimization Solver
SoS	Sum of Squares
YALMIP	Yet Another LMI Parser (sets up SDP for the solvers)

Symbols

β	Pitch angle
Γ	Tuning parameter for ensuring feasibility of SoS program
Λ	Eigenvalue of a matrix
λ	Lagrange multiplier associated with inequality constraints
	Quaternion
$\mathbb{R}$	Field of Real numbers
$\mathbb{R}[x]$	Ring of real polynomials in the variables $x \in \mathbb{R}^n$
$\mathbb{S}$	Space of Real symmetric matrices

$\mathbf{0}_{n\times n}$	Zero matrix having the dimensions $n \times n$
$\mathbf{\Omega}$	Satellite angular velocity vector
$\mathbf{I}$	Diagonal Inertia Matrix
$\mathbf{I}_{n\times n}$	Identity matrix having the dimensions $n \times n$
$\mathbf{q}$	Satellite Quaternion
$\mathcal{D}$	Domain of candidate Lyapunov function
$\mathcal{K}$	Cliques
$\mathcal{O}$	Time complexity expressed as an asymptotic upper bound of a function
$\mathcal{P}$	Newton Polytope
μ	Nonlinearity in Van der pol equations
ν	Lagrange multiplier associated with equality constraints
ω_g	Generator speed
ω_r	Rotor speed
$\omega_{r,f}$	Rotor speed obtained by filtering from the wind
ρ	Air density
$\Sigma[x]$	Space of SoS polynomials in $\mathbb{R}[x]$
θ_Δ	Torsion angle of the drive train
B_r	Viscous friction of the low-speed shaft
B_t	Tower damping coefficient
C	Capacitance
$d*$	Optimal value of dual objective function
I_k	Index set
J_g	Moment of inertia of the high-speed shaft
J_k	Index set
J_r	Moment of inertia of the low-speed shaft
K	Semialgebraic set

k_r	Torsion stiffness of the low-speed shaft
k_t	Tower torsion coefficient
L	Inductance
M_t	Mass of the tower
N_g	Drive train gear ratio
$p*$	Optimal value of primal objective function
$Q(g)$	Quadratic module of polynomial g
$s(d)$	Number of possible monomials with n variables and a maximum degree of d
V	Candidate Lyapunov function
v_r	Velocity of the rotor
v_t	velocity of the tower
v_w	velocity of the wind

List of Tables

List of Figures

Chapter 1

Introduction

The study of nonlinear systems is a fundamental study in control theory as virtually all physical systems are nonlinear in nature [Vid02]. The well-known, Lyapunov stability theorems form the basis for verifying stability of such systems. Specifically, the Lyapunov stability theorem is used to prove stability of an equilibrium point of the system under consideration. We find a candidate Lyapunov function $V(x)$ which should be positive i.e. $V(x) > 0$ (but zero at the equilibrium point i.e. $V(0) = 0$), while having a negative Lie derivative along the system trajectories i.e. $\dot{V}(x) < 0$ (see Theorem 3).

However, the Lyapunov stability theorem only gives the conditions which a candidate Lyapunov function must satisfy and does not provide any formal method for finding them. Intuitively, for a physical system, the candidate Lyapunov function can be considered as the total energy of the system as the energy of a system is always positive and if the energy of the considered system, is decreasing over time, then we can say that such a system is stable. However, finding the total energy of a complex real world system may not be straightforward. Further, if we are not able to find a suitable candidate Lyapunov function by trial and error, it may not mean that the considered equilibrium point for the system is unstable and the stability analysis is thus inconclusive.

1.1 Literature review

In recent years, many control theorists have proposed Sum of Squares (SoS) functions as a possible candidate Lyapunov function [PP05], [Par00], [Löf04], [AM19], [Slo16], [JW03], [Tan06] and [SPW12]. This is particularly, due to the fact that checking global non-negativity of an arbitrary function is an NP-hard problem [MK87] and one way to avoid this is to find an SoS function (instead of an arbitrary non-negative function) which is guaranteed to be non-negative. Furthermore,

the problem of finding an SoS function can be posed as an Semidefinite program (SDP) [Par00] which can then be solved using a solver like MOSEK [MOS16] or SeDuMi [Stu99]. Thus, we can relax the strict condition of non-negativity with an SoS condition which is referred to as an 'SoS relaxation' in the literature [Las10], [Wak+06] and [Las06]. Suitable MATLAB based toolbox have been developed for converting SoS problems into a standard SDP for the solvers such as SOSTOOLS [PP05] and YALMIP [Löf04], [Löf09]. In this project, we have exclusively used YALMIP for SOS programming.

It is important to note, that a multi-dimension function can be non-negative without being SoS. Further, as the number of variables grows, then the gap between non-negative and SoS polynomials increases and is unbounded [Las10]. As practical real-world systems are multidimensional in nature and may not be stable globally, we need a specific representation for finding SoS candidate Lyapunov functions which are positive within a local region. These representations are thus, valid only in a local region and are referred to as Positivstellensatz (which means "positive-locus-theorem") in the literature. In [Par00], [JW03] and [Tan06], Stengle's Positivstellensatz has been used to find numerical certificates of non-negativity which can be considered as candidate Lyapunov functions. However, Stengle's Positivstellensatz is computationally demanding nature and using it for a real-world problem is not practical. To circumvent this issue, Linear Programming and Second order Cone Programming based relaxations have been proposed in [AM19]. Further, Putinar's Positivstellensatz (and its sparse variant) has been used for computation of Lyapunov functions in [Slo16].

1.2 Motivation

In this project, we have focused on the algorithms and methods instead of a particular system. The methods have then been tried on the dynamical models which have been obtained from various sources and the focus is on algorithms for finding candidate Lyapunov functions instead of the system itself.

The motivation behind this project is to find Lyapunov functions and develop control law for nonlinear systems in an algorithmic way. Unlike nonlinear systems, for linear systems, synthesis of a control law is comparatively straightforward and can be done by using the well known Lyapunov Inequality.

Consider the LTI system $\dot{x} = Ax$. The system is stable if and only if $\exists\ P > 0$ such that

$$PA + A^T P < 0 \tag{1.1}$$

with $x^T Px$ being the candidate Lyapunov function.

A feedback $u = Kx$ stabalizes the LTI system $\dot{x} = Ax + Bu$ if and only if $\exists\ P > 0$ such that

$$P(A + BK) + (A + BK)^T P < 0 \tag{1.2}$$

We can write (1.2) as

$$(A + BK)Q + Q(A + BK)^T < 0 \tag{1.3}$$

with $Q = P^{-1}$, and define $Y = KQ$ such that

$$AQ + QA^T + BY + Y^T B^T < 0 \tag{1.4}$$

Thus we conclude, a feedback control $u = Kx$ stabalizes the LTI system $\dot{x} = Ax + Bu$ if and only if the LMI's

$$Q > 0 \quad \text{and} \quad AQ + QA^T + BY + Y^T B^T < 0 \tag{1.5}$$

with $Y = KQ$ are feasible.

Is it possible to find a Lyapunov function and do synthesis of a control law for a class of nonlinear systems? We attempt to do that by considering the candidate Lyapunov function as SoS polynomial. Further, we wish to address the issue of scalability (due to increasing computational complexity) of this approach to real world problems.

Based on the above discussions, we can broadly summarize the aims of the project as follows:

- Find candidate Lyapunov functions using SoS and Putinar's Positivstellensatz
- Explore scalability of this approach to real world problems and exploit sparsity in the considered system structure for reducing computational demand

1.3 Contribution

The following contributions were made in this work.

- In Chapter 4, we have suggested ideas for the construction of Index sets I and J while exploiting system sparsity for finding a candidate Lyapunov function. The proposed, algorithm (Algorithm 1) is useful for dynamical systems which are composed of cascaded subsystems.
- We have also proposed a generalized version of Algorithm 1 (Algorithm 2), which is based on ideas from [Wak+06] and an algorithm for construction of Index set J, once we have obtained Index sets I (Algorithm 3).

- A candidate Lyapunov function has been found for the Wind Turbine model (based on CART3 wind turbine model from [SPW12]) while a shutdown has been initiated.

1.4 Report Outline

The rest of the report is organized as follows:

- **Chapter 2** introduces the reader to the concepts of Sum of Squares (SoS) polynomials, Semidefinite Programming (SDP) and Lyapunov's stability theorem.
- **Chapter 3** introduces the reader to the concepts of Semialgebraic sets and Putinar's Positivstellensatz. Putinar's Positivstellensatz is a representation theorem, used for representing a SoS polynomial on a given semialgebraic set. This chapter also demonstrates the application of previously introduced concepts while finding a candidate Lyapunov function for Van der Pol's model of an nonlinear oscillator.
- **Chapter 4** is arguably the most important chapter of this thesis where, we have focused on exploiting sparsity in the system dynamics while finding the candidate Lyapunov functions. We have discussed Newton Polytopes and Sparse version of Putinar's Positivstellensatz and how to apply them.
- **Chapter 5** focuses on the application of the concepts discussed in Chapter 4 for finding candidate Lyapunov functions for complex systems such as Wind turbine, Ørsted satellite and Adaptive control.
- **Chapter 6** focuses on the computational considerations and verification of obtained candidate Lyapunov functions.
- **Chapter 7** focuses on the synthesis of control law based on the obtained candidate Lyapunov functions. We have explored nonlinear design techniques such as Lyapunov Redesign and Sontag's formula. We have also presented some intermediate results on the same and this chapter serves as the motivation for future work
- **Chapter 8** serves as the conclusion of this report.
- **Appendix A** shows the maximum constraint violations obtained after solving SDP 2 while finding the candidate Lyapunov function for the wind turbine model.

Chapter 2

Preliminaries

Summary *This chapter provides a background to the reader for concepts such as Sum of Squares (SoS) polynomials, Semidefinite Programming (SDP) and Lyapunov Stability which will be used heavily in the later chapters of this thesis. We begin by reviewing monomials and SoS polynomials. SoS polynomials allow us to relax the strict condition of non-negativity of a polynomial to the existence of an SoS decomposition which is a tractable problem compared to proving the non-negativity of a polynomial [Las10]. The problem of finding a SoS decomposition of a polynomial is converted into a SDP which can be solved by solvers like MOSEK or SeDuMi. Thereafter, we review SDP and state its general form as well as the concept of duality. Finally, we review the well known concepts of Lyapunov stability and Region of Attraction. This chapter is based on [Kha02], [Ber99], [BV04], [Las15], [Las10] and [Par00].*

2.1 Polynomial background and SoS polynomials

Consider the state variables $X \in \mathbb{R}^n$. A vector of monomials $v_d(x)$ is a set of all possible monomials $x^\alpha = (1, x_1, x_2, .., x_n, x_1^2, x_1x_2, ...x_n^d)$ where α and d are positive real numbers. $v_d(x)$ can be formed upto a degree d ($\alpha \leqslant d$) which is user-defined. The dimension of this vector is $s(d) := \binom{n+d}{d}$

Let $\mathbb{R}[x]$ be a set of polynomials in the variables X. A polynomial $p \in \mathbb{R}[x]$ is said to be Sum of Squares (SoS) if it can be written as

$$p(x) = \Sigma_j p_j(x)^2 \tag{2.1}$$

Proposition 1 : A polynomial p has a sum of squares decomposition if and only if there exists a real symmetric and positive semi-definite matrix $Q \in \mathbb{R}^{s(d)\times s(d)}$ (also referred to as Gram Matrix) such that

$$p(x) = v_d(x)^T Q v_d(x) \quad \forall x \in \mathbb{R}^n \tag{2.2}$$

A proof for the above proposition can be found in [Las10]. In the case of univariate polynomials i.e. polynomials having only a single variable x, any polynomial of even degree is nonnegative if and only if its SoS decomposition exists whereas, this is not true for multivariate polynomials [BPT12].

2.2 Problem of finding SoS decomposition as a Primal form of SDP

Given a SoS polynomial $p \in \mathbb{R}[x]^{2d}$, the coefficients of the matrix Q must satisfy the eq. (2.2). These conditions give affine constraints on Q. The vector of monomials v_d defined above forms the basis of SoS polynomial and we can write

$$v_d(x)v_d(x)^T = \sum_{\alpha \in \mathbb{N}^n} B_\alpha x^\alpha \tag{2.3}$$

for some $B_\alpha \in \mathbb{R}^{s(d)\times s(d)}$
Equation (2.2) can be rewritten as

$$\sum_\alpha p_\alpha x^\alpha = trace(v_d v_d^T Q) \tag{2.4}$$

by using the cyclic property of the trace operator. p_α is the coefficient associated with x^α and recall that $trace(A, B) = \sum_{i,j} A_{ij}B_{ij}$. Substituting, (2.3) in (2.4) we get,

$$\sum_\alpha p_\alpha x^\alpha = \sum_\alpha trace(B_\alpha Q)x^\alpha \tag{2.5}$$

Equating the coefficients of RHS and LHS of (2.5) and recalling that the trace operator is commutative, we get,

$$trace(Q, B_\alpha) = p_\alpha \tag{2.6}$$

We can now state the problem of finding an SoS decomposition for a polynomial $p(x)$ as a Primal form of SDP (explained in the next section) as stated in [Las10] and [Par00].

Theorem 1 *The problem of finding an SoS decomposition can be posed as the feasibility of SDP*

$$\begin{aligned}
&\textit{Find} && Q \in \mathbb{R}^{s(d)\times s(d)} && && (2.7a)\\
&\textit{such that} && Q = Q^T, && && (2.7b)\\
& && Q \succcurlyeq 0, && && (2.7c)\\
& && trace(Q, B_\alpha) = p_\alpha, && \forall \alpha \in \mathbb{N}^n && (2.7d)
\end{aligned}$$

2.3 Semidefinite Programming

In the previous section, we have concluded that, the problem of finding SoS decomposition for a polynomial can be posed as an SDP. In this section, we give a background on *mathematical optimization* and SDP.

2.3.1 General mathematical optimization problem and LP

A *mathematical optimization problem* can be stated as follows.

$$\begin{aligned} &\text{minimize} && f_0(x) && \text{(2.8a)}\\ &\text{subject to} && f_i(x) \leq b_i, \quad i = 1, \cdots, m && \text{(2.8b)} \end{aligned}$$

where, the vector $x \in \mathbb{R}^n$ is the *optimization variable* of the problem, the function $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, the functions $f_i : \mathbb{R}^n \to \mathbb{R}, \quad i = 1, \cdots, m$ are the *constraints* and the constants $b_{1,\cdots,m}$ are the *bounds*.

A vector x^* is called *optimal*, or a solution of the problem (2.8), if it has the smallest objective value among all vectors that satisfy the constraints i.e. for any z with $f_1(z) \leq b_1, \cdots, f_m(z) \leq b_m$, we have $f_0(z) \geq f_0(x^*)$.

Mathematical optimization problems are classified on the basis of the objective function and the constraints. Suppose, if the optimization problem (2.8) has a linear objective function f_0 and linear constraints f_i, then such an optimization problem is called Linear Programming (LP).

2.3.2 General form of SDP

A SDP is a generalization of an LP over convex cones of positive semidefinite matrices.

Let $\mathbb{S}_+^k$ denote cone of positive semidefinite $k \times k$ matrices. A general SDP can be stated as follows:

$$\begin{aligned} &\text{minimize} && c^T x && \text{(2.9a)}\\ &\text{subject to} && x_1 F_1 + \cdots + x_n F_n + G \preccurlyeq 0, && \text{(2.9b)}\\ & && Ax = b && \text{(2.9c)} \end{aligned}$$

where, $G, F_1, \cdots, F_n \in \mathbb{S}_+^k$, $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$, $c \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$.

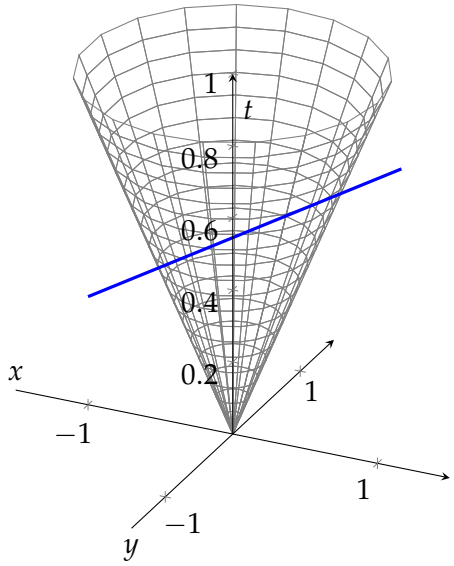


Figure 2.1: Feasible region of an SDP is the intersection between the Positive Semidefinite Cone $\mathbb{S}^2_+$ due to 2.9b and the blue line due to 2.9c.

The inequality constraints are expressed as an Linear Matrix Inequality (LMI) and it provides two advantages. Firstly, multiple inequality constraints can be clubbed together as a single LMI. Secondly, nonlinear (convex) inequalities can be converted to an LMI using Schur complement [Boy+94], [BV04]. If the matrices $G, F_1, \cdots, F_n$ are diagonal matrices, than the LMI reduces to a set of inequalities and correspondingly the SDP reduces to an LP.

The problem of finding SoS decomposition is stated as a primal form of SDP (2.7) and it is only a feasibility problem which implies that the set composed of semidefinite cone and equality constraints should be non-empty or feasible (see fig. (2.1)). Thus, the objective function in this case is void as we are only checking the feasibility of the SDP. The constraint (2.7c) is an inequality constraint represented by an LMI of size $s(d) \times s(d)$ and the other two constraints (2.7b) and (2.7d) are equality constraints.

2.3.3 Duality

We now discuss the concept of duality for an SDP. Solving a dual problem for an optimization problem gives the best lower bound on the primal objective function. The dual problem for (2.9) is formed by using the Lagrangian dual function $L(x, \lambda, \nu) : \mathbb{R}^n \times \mathbb{S}^k \times \mathbb{R}^p \mapsto \mathbb{R}$ which is defined as follows:

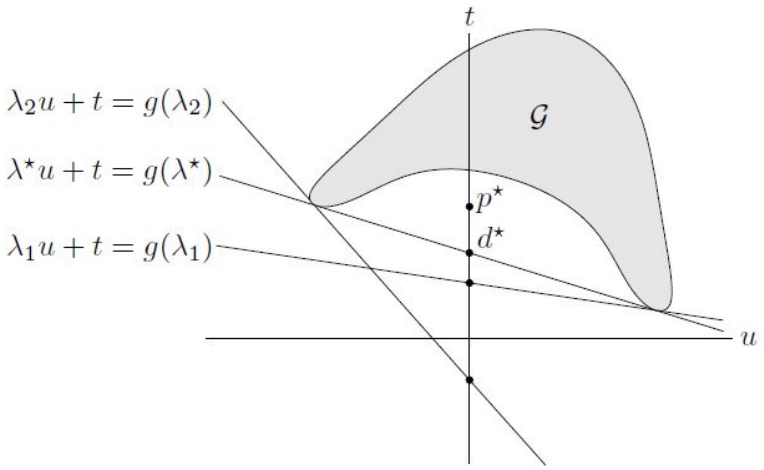


Figure 2.2: Geometric interpretation of duality from [BV04]. Strong duality does not hold in this case as duality gap is positive.

$$L(x, \lambda, \nu) = c^T x + trace(\lambda, x_1 F_1 + \cdots + x_n F_n - G) + \sum_{i=1}^{p} \nu_i (Ax - b) \tag{2.10}$$

where, λ is a matrix $\in \mathbb{S}^k$ of Lagrange multipliers associated with the LMI constraint (2.9b) and $\nu \in \mathbb{R}^p$ are Lagrange multipliers associated with the p equality constraints (2.9c). λ and ν are also refered to as the dual variables.

The lower bound (parametrized by λ and ν) of the primal SDP is found by taking the infimum of the Lagrangian (2.10).

$$g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) \tag{2.11}$$

$$g(\lambda, \nu) = \begin{cases} -trace(G, \lambda) - b^T \nu, & \text{if } trace(F_i, \lambda) + c_i + A^T \nu, \quad i = 1, \cdots, n \\ -\infty, & \text{otherwise} \end{cases} \tag{2.12}$$

Solving (2.12) gives us the optimal value $d*$ which is the lower bound for optimal value $p*$ of primal objective function obtained by solving (2.9). If the LMI constraint (2.9b) is strictly feasible i.e. $x_1 F_1 + \cdots + x_n F_n + G \prec 0$, than the SDP has a strong duality, which implies $p* = d*$. A geometric interpretation of duality is given in fig. (2.2). Further information on duality such as Slater's conditions can be found in [Ber99] and [BV04].

2.4 Lyapunov Stability

We now give a brief introduction to the well known Lyapunov stability theorem. Primarily, we shall be using Lyapunov stability theorems for proving stability of equilibrium points for a dynamical system. There are other kinds of stability theorems as well such as input-output stability and stability of periodic orbits. This section is based on [Kha02] and [SL+91].

2.4.1 Equilibrium point

Consider the following autonomous system:

$$\dot{x} = f(x) \tag{2.13}$$

where, $f : \mathcal{D} \to \mathbb{R}^n$ is a locally Lipschitz map from a domain $\mathcal{D} \subset \mathbb{R}^n$ into $\mathbb{R}^n$ and $f(0) = 0$ is the *equilibrium point* for (2.13).

Theorem 2 *The equilibrium point $x = 0$ of (2.13) is*

- *stable, if for each $\epsilon > 0$, there is $\delta = \delta(\epsilon) > 0$ such that*

$$\|x(0)\| < \delta \Rightarrow \|x(0)\| < \epsilon, \quad \forall t \geq 0 \tag{2.14}$$

- *unstable, if it is not stable*
- *asymptotically stable, if it is stable and δ can be chosen such that*

$$\|x(0)\| < \delta \Rightarrow \lim_{t \to \infty} x(t) = 0 \tag{2.15}$$

If the equilibrium point is not at the origin, then it can be shifted to the origin via a change of variables (without any loss of generality).

2.4.2 Lyapunov Stability theorem

We now state, the well-known Lyapunov's stability theorem for proving stability of equilibrium point for the system (2.13). By using, Lyapunov's stability theorem, we can prove stability of the equilibrium point (origin) of the system, without solving the differential equation (2.13) for all possible trajectories [Kha02].

Theorem 3 *Let $x = 0$ be an equilibrium point for (2.13) and $\mathcal{D} \subset \mathbb{R}^n$ be a domain containing $x = 0$. Let $V : \mathcal{D} \to \mathbb{R}$ be a continuously differentiable function such that:*

$$V(0) = 0 \tag{2.16a}$$
$$V(x) > 0 \; \forall x \neq 0 \tag{2.16b}$$
$$\dot{V}(x) < 0 \tag{2.16c}$$

then $x = 0$ is asymptotically stable.

If we relax the strict inequality $\dot{V}(x) < 0$ to $\dot{V}(x) \leq 0$, then the equilibrium point is stable but we cannot conclude whether it is asymptotically stable or not. LaSalle's Invariance Principle [Kha02] can be used in such a case.

2.4.3 Region of Attraction (ROA)

In this subsection, we define Region of Attraction (ROA) as given in [Kha02]. Let the origin $x = 0$ be an asymptotically stable equilibrium point for the system (2.13). Let $\phi(t; x)$ be the solution of (2.13) that starts at initial state x at time $t = 0$. ROA of the origin can be defined as follows:

$$ROA = \{x \in \mathcal{D} \mid \phi(t; x) \text{ is defined } \forall t \geq 0 \text{ and } \phi(t; x) \to 0 \text{ as } t \to \infty\} \tag{2.17}$$

Estimation of ROA is of great practical significance. It can be used in estimating critical fault clearing time for a nonlinear system such that if the fault is cleared within this time interval, the system does not become unstable [Kha02]. The following Lemma from [Kha02] gives some properties of ROA.

Lemma If $x = 0$ is an asymptotically stable equilibrium point for the system (2.13), then its ROA is an open, connected, invariant set. Moreover, the boundary of ROA is formmed by trajectories.

This Lemma suggests that ROA can be determined by characterizing the trajectories lying on its boundary. It is important to note that the ROA is not always equivalent to domain $\mathcal{D}$ of the Lyapunov function. An example demonstrating this can be found in [Kha02].

Chapter 3

Putinar's Positivstellensatz

Summary *In this chapter, we discuss two key concepts from algebraic geometry which we have used extensively in this work. We begin by defining a semialgebraic set as per [Las10], which is a subset of $\mathbb{R}^n$. This set can be considered as the domain $\mathcal{D}$ of a Candidate Lyapunov function. Thereafter, we have stated Putinar's Positivstellensatz which is used to certify a polynomials positivity on a given semialgebraic set. Finally, we demonstrate the application of the previously stated concepts of SoS, semialgebraic sets and Putinar's Positivstellensatz for proving stability of Van der Pol's equation which are stated as a 2^{nd} order autonomous state space model. This chapter is based on [Kha02], [Lau09], [Las06] and [Las10], [Tan06].*

3.1 Semialgebraic sets

A semialgebraic set is a compact subset of $\mathbb{R}^n$ and can be defined as follows:

Let $(g_j)_{j=1}^m \in \mathbb{R}[x]$ be such that the basic semialgebraic set

$$K := \{x \in \mathbb{R}^n : g_j(x) \geq 0, j = 1, \cdots, m\} \tag{3.1}$$

Using Putinar's Positivstellensatz, we can obtain polynomial certificates of positivity on the compact semialgebraic set K defined above (3.1).

We now define Quadratic modules $Q(g)$ associated with the polynomials $g_j \subset \mathbb{R}[x]$ (that define K).

$$Q(g) = Q(g_1, \cdots, g_m) := \left\{ q_0 + \sum_{j=1}^{m} q_j g_j : \quad (q_j)_{j=0}^m \subset \Sigma[x] \right\} \tag{3.2}$$

A quadratic module $Q(g)$ is called Archimedean if $M - \sum_{i=1}^n x_i^2 \in Q(g)$ for some $M \in \mathbb{N}$

3.2 Putinar's Positivstellensatz

Earlier in Chapter 1 and 2, we had discussed that for a non-negative polynomial $p \in \mathbb{R}[x]$ does not necessarily needs to have an SoS representation in the multivariate case. Further, we are interested in constrained polynomial optimization as the considered system may not be globally stable. Therefore, we need representation theorems for constrained polynomial optimization of multivariate polynomials. Putinar's Positivstellensatz is one such representation theorem which focuses on the positivity of a polynomial on a compact basic semialgebraic set [Put93]. Besides Putinar's Positivstellensatz, there are other representation theorems such as Stengle's Positivstellensatz which characterizes when a semialgebraic set described by polynomial inequalities, equalities and non-equalities is empty (see Theorem 2.11 in [Las10] and [Las15]), Pólya provided a certificate of positivity for homogeneous polynomials that are positive on a simplex and Schmüdgen's Positivstellensatz which does not require Assumption 2.1 but is computationally more demanding as the number of terms is exponentional in number of polynomials that define K [Las10]. We have therefore, focused on Putinar's Positivstellensatz in this thesis due to its simplicity and less computational demand as compared to Stengle's, Pólya's and Schmüdgen's Positivstellensatz.

Putinar's Positivstellensatz can be stated as follows:

Assumption 2.1 Let K be a semialgebraic set as defined in (3.1) and let there exist $u \in Q(g)$ such that the level set $\{x \in \mathbb{R}^n : u(x) \geq 0\}$ is compact.

Theorem 4 *If $F \in \mathbb{R}[x]$ is strictly positive on K than $F \in Q(g)$ that is,*

$$F = F_0 + \sum_{j=1}^{m} F_j g_j \tag{3.3}$$

for some SoS polynomials $F_j \in \Sigma[x]$, $j = 0, 1, \cdots, m$.

Compared to Stengle's Positivstellensatz or Schmüdgen's Positivstellensatz, Putinar's Positivstellensatz is computationally less expensive as the number of terms in (3.3) is linear in the number of terms that define K [Las10]. Computational complexity is further discussed later in chapter 6.

The Assumption 2.1 required for application of Putinar's Positivstellensatz is not very restrictive [Las10] and is always satisfied in the following cases:

- All the g_i's are affine and K is a polytope
- The set $\{x \in \mathbb{R}^n : g_j(x) \geq 0\}$ is compact for some $j \in 1, \cdots, m$

Further this assumption implies that the Quadratic module $Q(g)$ generated by the semialgebraic set K is Archimedean.

3.3 Finding Lyapunov functions using Putinar's Positivstellensatz

Consider the autonomous system given in (2.13) which has an equilibrium point at the origin. The problem of finding an Lyapunov function for such a system using SoS and Putinar's Positivstellensatz can be stated as follows.

We begin by defining an semialgebraic set which includes the equilibrium point. Any trajectory starting within this set will asymptotically converge to the equilibrium point (Locally Asymptotically Stable). Thus, a semialgebraic set can be considered as an ROA for the equilibrium point. Using Putinar's Positivstellensatz, we can find a suitable candidate Lyapunov function which is required to be positive only within the predefined semialgebraic set. In a similar way, we constrain the candidate Lyapunov function's Lie derivative such that it is negative over the predefined semialgebraic set. Thus, the following SDP is posed for finding Lyapunov function.

Semidefinite Program 1

$$\begin{aligned}
&\textit{Find} && V(x) && \text{(3.4a)}\\
&\textit{such that} && V(0) = 0, && \text{(3.4b)}\\
& && V(x) - \sum_{j=1}^{m} F_j g_j - \Gamma \left\|x\right\|_2 \in \Sigma[x]\ ,\ j = 1, \cdots, m, && \text{(3.4c)}\\
& && -\frac{\partial V(x)}{\partial x} f(x) - \sum_{j=1}^{m} F_j g_j - \Gamma \left\|x\right\|_2 \in \Sigma[x]\ ,\ j = 1, \cdots, m && \text{(3.4d)}
\end{aligned}$$

Since SDP solvers can violate the constraints slightly, while solving the optimization problem ([Löf09] and [Löf11]), we use Γ (which is a small constant of order 10^{-3}) to make $V(x)$ and $-\frac{\partial V(x)}{\partial x} f(x)$ greater than $\left\|x\right\|_2$ which ensures that they are greater than 0.

3.4 Van Der Pol equations

The Van Der Pol equations are a specific case of Lienard's equations which were used to model oscillating circuits during the development of vacuum tube circuits [Kha02]. In this section, we apply the previously defined SDP 1 for proving the stability of a simple physical system. For an electronic circuit having an inductor

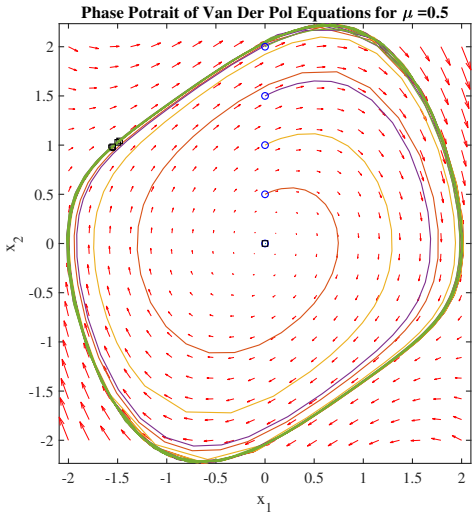


Figure 3.1: Van Der Pol Equations Phase portrait revealing limit cycle

$L > 0$, capacitor $C > 0$ and a voltege v across the resistive element, the Van Der Pol equations can be stated as follows:

$$\ddot{v} - \mu(1 - v^2)\dot{v} + v = 0 \tag{3.5}$$

where $\mu = \sqrt{\frac{L}{C}}$

In the above equation it can be seen that the term μ affects the nonlinearity in the system. When $\mu = 0$, the Van Der Pol equations are reduced to a linear harmonic oscillator which has an exact analytical solution.

We begin analyzing the system by writing the model in state space. Let us choose $x_1 = v$ and $x_2 = \dot{v}$ as state variables to obtain

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 + \mu(1 - x_1^2)x_2 \end{aligned} \tag{3.6}$$

The phase portrait of eq. (3.6) reveals that the system has a stable limit cycle fig. (3.1) which implies that all trajectories starting near the limit cycle asymptotically tends towards the limit cycle [Kha02].

Proving stability of a limit cycle is harder than proving the stability of an equilibrium point and is beyond the scope of this project. We therefore consider the Van Der Pol equations in reverse time which has an unstable limit cycle (i.e. all trajectories starting near the limit cycle asymptotically tends away from the limit

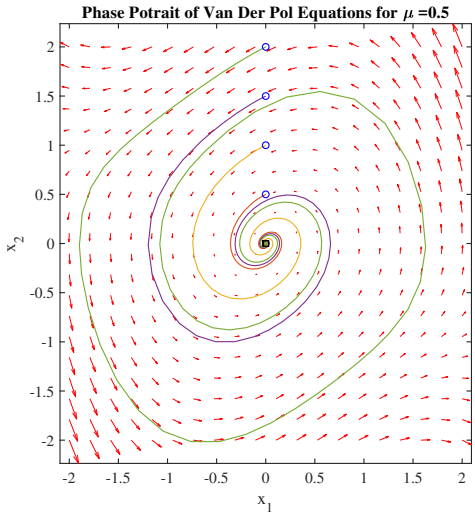


Figure 3.2: Phase portrait for Van Der Pol Equations in reverse time

cycle) but an stable equilibrium point at the origin and a ROA around it. Any trajectory starting within the ROA will asymptotically converge towards the origin while any trajectory starting outside the ROA will go away from the origin. The state space model of Van Der Pol equations in reverse time can be stated as follows:

$$\begin{aligned} \dot{x}_1 &= -x_2 \\ \dot{x}_1 &= x_1 - \mu(1 - x_1^2)x_2 \end{aligned} \tag{3.7}$$

The phase portrait corresponding to eq. (3.7) can be seen in the corresponding fig. (3.2).

The SoS problems are posed as an SDP (Semidefinite Program 1) by using the YALMIP environment in MATLAB [Löf09]. For the Van Der Pol equations considered in this section, the following Lyapunov function candidate was obtained (see fig. (3.3))

$$\begin{aligned} &6.62440344927 * x_1^2 + 0.0260363377173 * x_1^2 * x_2 \\ &-2.40737613573 * x_1 * x_2 + 2.28459682059 * x_2^2 \\ &-0.0517333345533 * x_1^3 - 1.23565122248 * x_1^4 \\ &-0.822713464636 * x_1^2 * x_2^2 - 0.00690539002097 * x_1 * x_2^3 \\ &-0.0894895760021 * x_2^4 \end{aligned} \tag{3.8}$$

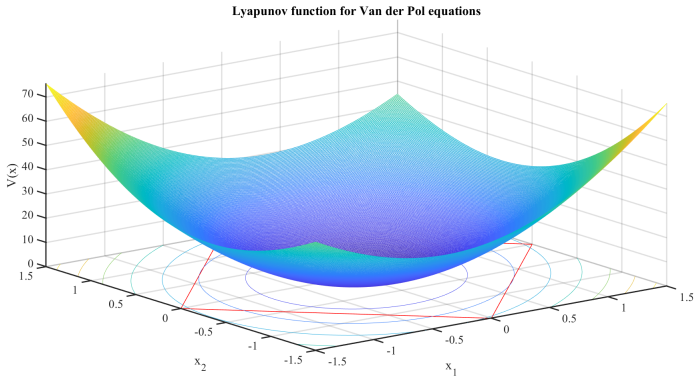


Figure 3.3: Candidate Lyapunov function V. x_1 and x_2 are system states. The red box is the semialgebraic set.

with the following Lie derivative (see fig. (3.4))

$$\begin{aligned}
&-2.40737613573 * x_1^2 + 0.0250183150736 * x_1^2 * x_2 \\
&+3.35726742126 * x_1 * x_2 - 20.4385920702 * x_2^2 \\
&+0.0260363377173 * x_1^3 - 0.0520726754346 * x_1 * x_2^2 \\
&-8.73970271797 * x_1^3 * x_2 + 31.0523866822 * x_1^2 * x_2^2 \\
&+1.39104947558 * x_1 * x_2^3 + 1.79669691006 * x_2^4 \\
&+0.130181688586 * x_1^4 * x_2 - 8.22713464636 * x_1^4 * x_2^2 \\
&-0.103580850314 * x_1^3 * x_2^3 - 1.78979152004 * x_1^2 * x_2^4
\end{aligned} \tag{3.9}$$

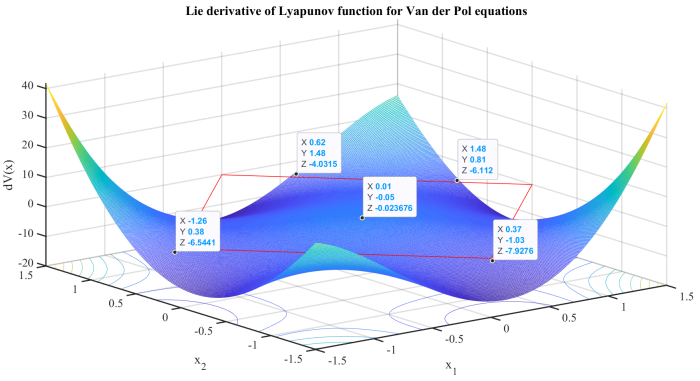


Figure 3.4: Lie derivative dV of the candidate Lyapunov function. x_1 and x_2 are system states. The red box is the semialgebraic set.

It is important to noted that SDP 1 finds a candidate Lyapunov function which only needs to satisfy (3.4b), (3.4c) and (3.4d) within the semialgebraic set. There

is no guarantee that the constraints will also be satisfied outside the semialgebraic set. It can be seen in fig. 3.4) that the constraint (3.4d) is not satisfied outside the semialgebraic set. For clarity, we have added a scaled version of fig. (3.4) which shows only the region of the Lie derivative within the semialgebraic set (see fig. (3.5)).

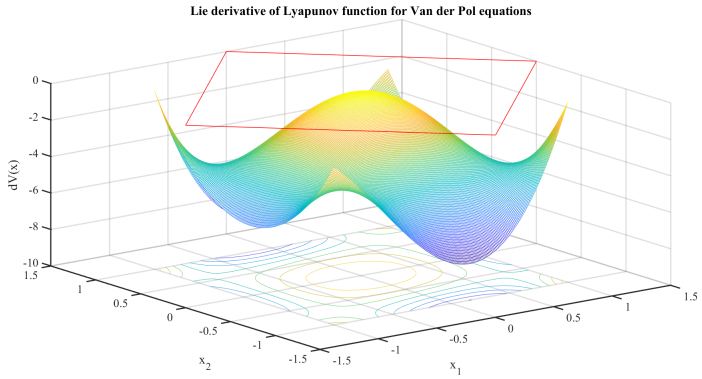


Figure 3.5: Lie derivative dV of the candidate Lyapunov function. x_1 and x_2 are system states. The red box is the semialgebraic set. The figure has been scaled to show only the region within the semialgebraic set.

The degree of SoS polynomial was chosen to be 4. The semialgebraic set considered for this was a box in $\mathbb{R}^2$ with edges at $(0, 1.5), (0, -1.5), (1.5, 0), (-1.5, 0)$ and is shown in fig (3.6).

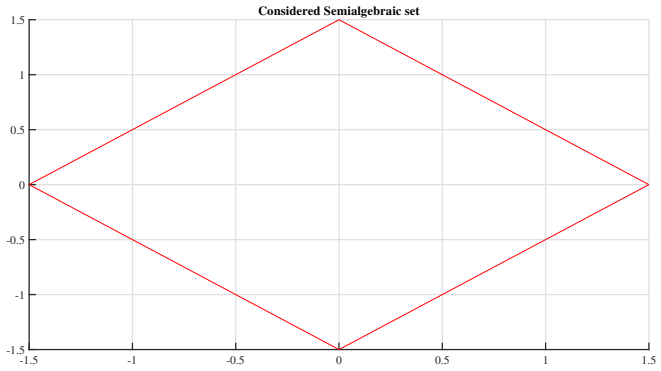


Figure 3.6: The semialgebraic set considered for finding Lyapunov function (3.8)

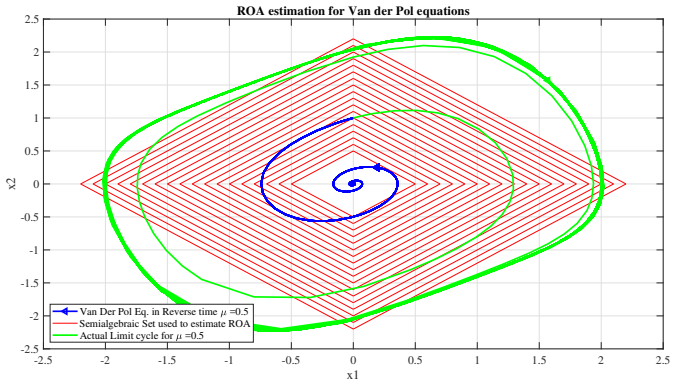


Figure 3.7: Estimation of ROA for Van Der Pol equations with $\mu = 0.5$

3.4.1 Estimation of ROA using semialgebraic sets

The basic tool for estimating ROA is Zubov's theorem which is given in [Kha02]. One drawback in using Zubov's theorem is that, it requires the solution of a partial differential equation. [Kha02] also suggests that the ROA can be estimated as the set Ω which is a compact positively invariant subset of $\mathcal{D}$ i.e. every trajectory starting in Ω stays in Ω for all future time. The semialgebraic set K defined in (3.1) can be considered as such a subset of $\mathcal{D}$. We start with a small K which is close to the equilibrium point and we keep increasing the boundaries of K till it becomes equivalent to the ROA. As long as the semialgebraic set considered in the analysis is a subset of the ROA, all the constraints specified by (2.16), (3.1) and (3.3) are satisfied and the SDP is feasible. If the edges of the considered semialgebraic set exceeds the ROA, the SDP becomes infeasible. The ROA can thus, be estimated by increasing the bounds of the semialgebraic set until the SDP becomes infeasible as shown in fig. (3.7) and fig. (3.8).

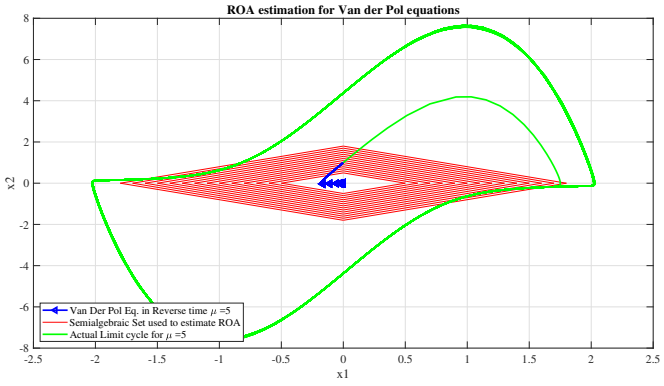


Figure 3.8: Estimation of ROA for Van Der Pol equations with $\mu = 5$

The estimate of ROA using this method is conservative when μ is increased as can be seen in the fig. (3.8) where only a small subset of ROA is correctly estimated. The estimate can be improved if we consider a circular or an elliptical semialgebraic set instead of a rectangular set. The ROA estimate is affected by the numerical accuracy and precision of the solver package and it can be observed in fig (3.7) that the estimated ROA exceeds the actual ROA by a single step. This can be improved by reducing the step size at the cost of extra computing time by the solver.

3.5 Conclusion

We can conclude that by applying Semidefinite program 1, we have an algorithmic way of finding candidate Lyapunov functions and finding an estimate of the ROA. This is true for systems having few variables. However, when dealing with systems involving large number of variables, the Semidefinite Program 1 quickly becomes too large for obtaining a solution. Fortunately, most of the dynamical systems are sparse in nature and it is possible to exploit this sparsity while searching for candidate Lyapunov function. This is the main study topic for the following chapters and this project.

Chapter 4

Sparsity in system dynamics

Summary *This chapter focuses on exploiting sparsity in the dynamical system while finding a candidate Lyapunov function. As discussed previously, SDP 1 works only for dynamical systems having few state variables. We begin by reviewing the concepts of Netwon Polytope and Sparse Putinar's Positivstellensatz as given in [Löf09] and [Las10] respectively. The sparsity patterns of the dynamical system are represented via Index sets and we present a couple of algorithms for constructing Index sets such that they satisfy Assumption 4.1, Assumption 4.2 and the Running Intersection Property 4.12 simultaneously. Algorithm 2 is based on [Wak+06] and uses concepts from graph theory which we have briefly reviewed for the sake of completeness. Finally, we have applied Sparse Putinar's Positivstellensatz for reformulating SDP 1 such that it exploits inherent sparsity in dynamical systems and is thus, computationally more efficient. This chapter is based on [Las10], [Löf09], [Slo16], [Ant13], [SPW12] and [Wak+06].*

4.1 Motivation

Recall from Theorem 1 in Chapter 2, the elements of the Q matrix impose affine constraints on the SDP. Let us consider a state space model, having $n = 7$ states and if we search for a candidate Lyapunov function having degree $d = 2$, the size of matrix Q in (2.7) is $s(d) = \binom{7+2}{2} = 36$. Thus the dimensions of Q will be 36×36 in this case. This is equivalent to an SDP with 1296 affine constraints. As the number of constraints increases, the SDP becomes practically unsolvable. Thus, we wish to exploit sparsity in the dynamical system.

4.2 Sparse and Dense Polynomials

We begin by defining a dense polynomial and a sparse polynomial.

Definition 4.2.1 *Consider the state variables $X = (x_1, x_2, .., x_n) \in \mathbb{R}^n$ and the vector $v_d(x)$ consisting of all possible monomials $x^\alpha = (1, x_1, x_2, .., x_n, x_1^2, x_1x_2, ...x_n^d)$ which can be formed upto a degree d ($\alpha \leqslant d$).*

- *If the polynomial is composed of all possible monomials, then the polynomial is considered to be a dense polynomial.*
- *If the polynomial does not contain all possible monomials, then the polynomial is considered to be a sparse polynomial.*

4.3 Newton Polytope

The Newton polytope of polynomial is one of the earliest results in exploiting sparsity while finding SoS decomposition for a polynomial [Lau09]. It has already been implemented in YALMIP SoS module and its implementation is discussed in [Löf09]. We begin by finding the support function of a polynomial.

Consider the SoS polynomial $p = \sum_{\alpha \in \mathbb{N}^n} c_\alpha x^\alpha \in \mathbb{R}[x]$, the support function of p is

$$supp(p) := \{\alpha \in \mathbb{N}^n \quad | \quad c_\alpha \neq 0\} \tag{4.1}$$

The Newton Polytope is the convex hull of the support function [Lau09], [Slo16] and can be formally defined as follows:

$$New(p) = co(supp(p)) \tag{4.2}$$

Some authors [Löf09] define the Newton Polytope in an equivalent way as follows: The Newton polytope is the convex hull of exponents of the polynomial in $\mathbb{R}^n$, where n is the number of variables in the polynomial. We illustrate the Newton Polytope by a simple example.

Consider the polynomial $p = x_1^2 + x_2^2 + 2x_1$ with variables $x \in \mathbb{R}^2$. The Newton Polytope for p is shown in fig. (4.1).

Newton polytope is used in order to reduce the considered monomials based on the sparsity in the considered polynomial considered for SoS decomposition [Löf09]. In fact, [Wak+06] defines a polynomial p as *sparse*, if the number of elements in its support $supp(p)$ is much smaller then the number of elements that

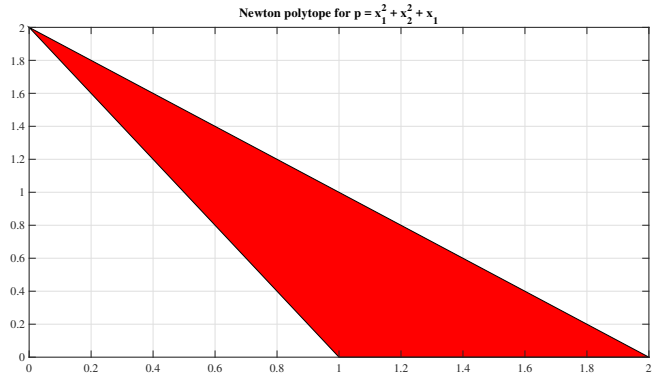


Figure 4.1: Newton Polytope for polynomial $p = x_1^2 + x_2^2 + 2x_1$

form a support of a fully *dense* polynomial. We can illustrate this by following example.

Consider the polynomial $p = x_1^4 + x_2^4 + x_1x_2$ with variables $x \in \mathbb{R}^2$. The exponents of the polynomial p are

$$P = \begin{bmatrix} 4 & 0 & 1 \\ 0 & 4 & 1 \end{bmatrix} \tag{4.3}$$

and the exponents of all possible monomials in $\mathbb{R}[x_1, x_2]$ are

$$S = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{bmatrix} \tag{4.4}$$

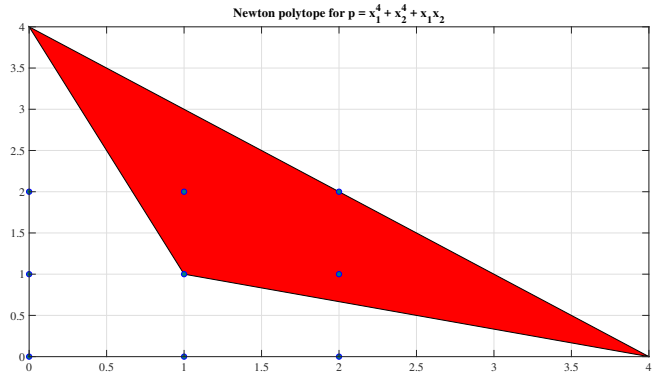


Figure 4.2: Newton Polytope for polynomial $p = x_1^4 + x_2^4 + x_1x_2$

The corresponding Newton polytope for p is shown in fig. (4.2) in red shaded area and the blue dots represent the candidate monomials for SoS decomposition.

The monomials lying outside the polytope are rejected as they are not present in the original polynomial itself due to its sparse nature. The resulting set of candidate monomials after applying Newton polytope is

$$S = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{4.5}$$

The monomial $x_1^2 x_2^2$ is rejected (despite appearing on the boundary of Newton polytope in fig. (4.2)) because we only consider polynomials having a maximum degree of $1/2\ p$.

4.3.1 Implementation of Newton Polytope in YALMIP

The Newton polytope $\mathcal{P}$ is implemented in YALMIP by checking if we can find a separating hyperplane between a candidate monomial s_i and all the vertices of $1/2\mathcal{P}$. If we can find a such a separating hyperplane, than the candidate monomial is not in $1/2\mathcal{P}$ and it can be rejected. The problem of finding an separating hyperplane is an Linear program (LP) [BV04] and it can be stated as follows:

$$\begin{aligned} \text{maximize} \quad & a^T s_i - b && \text{(4.6a)} \\ \text{subject to} \quad & \frac{1}{2} a^T p_k - b \leq 0 \ \forall k = 1, \cdots, N && \text{(4.6b)} \\ & a^T s_i - b \geq 0 && \text{(4.6c)} \end{aligned}$$

where, a and b define the hyperplane (which is a generalization of a line in higher dimensions), s_i are the candidate monomials and p_k are the exponents of the polynomial $p(x)$ whose SoS decomposition we are interested in finding. If the LP (4.6) is infeasible, then no strictly feasible hyperplane exists for that particular s_i and therefore, the candidate monomial s_i cannot be rejected. On the other hand, if the objective function (4.6a) is unbounded then the candidate monomial s_i does not exist inside the Newton Polytope and can thus be rejected [Löf09].

4.4 Sparse Putinar's Positivstellensatz

We now review a 'sparse' version of Putinar's Positivstellensatz as given in [Las10]. This representation is the most computationally efficient and the computational savings obtained from it are discussed in Chapter 6. Prior to applying Sparse Putinar's Positivstellensatz, we have assumed that a a sparse dynamical system may have a Lyapunov function that has the same (or finer) sparsity pattern as the dynamical system itself.

Consider $\mathbb{R}[x] = \mathbb{R}[x_1, \cdots, x_n]$. Let $K \subset \mathbb{R}[x]$ be as defined in (3.1). Let $I \subset \mathbb{N}$ denote a finite index set. An Index set I_k ensures that $\mathbb{R}[x(I_k)]$ consists only of variables stated in the index set. Formally, an Index set can be defined as follows:

$$I_k := \{x(I_k) \quad | \quad x_i \quad : \quad i \in I_k\} \tag{4.7}$$

Let $I_0 := \{1, \cdots, n\}$ be the union $\bigcup_{k=1}^{p} I_k$ of p subsets $I_k, \quad k = 1, \cdots, p$ (with possible overlaps). Let $n_k := |I_k|$. Thus, $\mathbb{R}[x(I_k)]$ is a ring of polynomials in the n_k variables $x(I_k)$ and $\mathbb{R}[x(I_0)] = \mathbb{R}[x]$ as per (4.7). The Index sets characterize sparsity as weak couplings between the variables x in the polynomial. We now present a simple example to clarify the concept of weak couplings and Index sets.

Consider the following polynomial.

$$F(x_1, x_2, x_3) = f_1(x_1, x_2) + f_2(x_2, x_3) \tag{4.8}$$

As can be seen from (4.8), the variables x_1 and x_3 are not directly coupled or correlated. However, they are coupled via variable x_2. Thus, the variables are weakly coupled and we can characterize sparsity via Index sets. In this case, we can make 2 subsets (thus, $p = 2$), I_1 and I_2 with $I_0 = \{1, 2, 3\}$. $I_1 = \{1, 2\}$ with $n_1 = 2$ and $I_2 = \{2, 3\}$ with $n_2 = 2$.

We now state two important assumptions related to the semialgebraic set K which must be satisfied in order to exploit sparsity while representing a polynomial F as SoS polynomial.

Assumption 4.1 Let $K \subset \mathbb{R}^n$ be as in (3.1). A scalar $M > 0$ is know such that $\|x\|_\infty < M$ for all $x \in K$. Under this assumption, we have $\sum_{i \in I_k} x_i^2 \leq n_k M^2, \quad k = 1, \cdots, p$ and we are adding p redundant polynomial inequalities $g_k(x) \geq 0$ in the definition of K (3.1).

$$g_{m+k}(x) := n_k M^2 - \sum_{i \in I_k} x_i^2 \qquad k = 1, \cdots, p \tag{4.9}$$

and set $m' = m + p$ so that K is now defined as

$$K := \{x \in \mathbb{R}^n : \quad g_j(x) \geq 0, \quad j = 1, \cdots, m'\} \tag{4.10}$$

Note that $g_{m+k} \in \mathbb{R}[x(I_k)]$, for all $k = 1, \cdots, p$. These additional polynomial inequalities are redundant as in order to apply Putinar's Positivstellensatz, the quadratic modules $Q(g)$ generated by K are compact which means that they are Archimedean as stated in theorem 4. We now state the second assumption required for sparse Putinar's representation.

Assumption 4.2 Let $K \subset \mathbb{R}^n$ be as in (4.10). The index set $J = \{1, \cdots, m'\}$ is partitioned into p disjoint sets J_k, $k = 1, \cdots, p$. Further, $\{I_k\}$ and $\{J_k\}$ satisfy:

- For every $j \in J_k$, $g_j \in \mathbb{R}[x(I_k)]$, that is, for every $j \in J_k$, $g_j(x) \geq 0$ only involves the variables $x(I_k) = \{x_i : i \in I_k\}$
- The objective function $F \in \mathbb{R}[x]$ can be written as

$$F = \sum_{k=1}^{p} F_k, \quad \text{with } F_k \in \mathbb{R}[x], \quad k = 1, \cdots, p \tag{4.11}$$

We now state sparse version of Putinar's Positivstellensatz.

Theorem 5 *Let $K \subset \mathbb{R}^n$ be as in (4.10) with the above assumptions and in addition, assume that for every $k = 1, \cdots, p-1$,*

$$(I_{k+1} \cap (\bigcup_{j=1}^{k} I_j)) \subseteq I_s \quad \textit{for some } s \leq k \tag{4.12}$$

If $F \in \mathbb{R}[x]$ is strictly positive on K, then

$$F = \sum_{k=1}^{p} \left(q_k + \sum_{j \in J_k} q_{jk} g_j \right) \tag{4.13}$$

for some SoS polynomials $(q_k, q_{jk}) \subset \mathbb{R}[x(I_k)]$, $k = 1, \cdots, p$.

Eq. (4.12) is referred as Running Intersection Property. Running Intersection Property captures the absence of extra coupling variables and ensures sparsity in SoS representation (4.13). A system can thus be considered to be a sparse system if the Running Intersection property is satisfied. From Assumption 4.1, Assumption 4.2 and (4.12), it is evident that if we add a new Index set I, we will get a new redundant polynomial inequality $g_K(x)$ and we will also need to construct a new Index set J which should satisfy Assumption 4.1, 4.2 and (4.12). Note that, the index set J are disjoint from each other and they have to include all the polynomial inequalities including the redundant polynomial inequalities.

4.4.1 Construction of Index sets

Theorem 5 only gives the conditions which should be satisfied while constructing Index sets and it does not give us a clear procedure on how to do so. The focus of this subsection is to construct an algorithm which exploits sparsity. Further, we will now consider dynamical systems and we will be exploiting the system dynamics to obtain a sparse polynomial as the candidate Lyapunov function.

Firstly, we divide the dynamical model into relevant subsystems. We then, find out the states which are correlated with other states and the states which connect different subsystems together. We begin by illustrating the possible construction(s) of Index set I using Venn diagrams and small examples. This is for the sake of better intuition into the problem.

Suppose, we have a dynamical system which is composed of many subsystems which are correlated with each other via certain states. Consider the Index sets I_1 and I_2 based on any two subsystems which are correlated via some common states (represented by the intersection between sets). The first figure shows some of the possibilities while constructing a third Index set I_3 such that the Assumptions 4.1, 4.2 and (4.12) are satisfied.

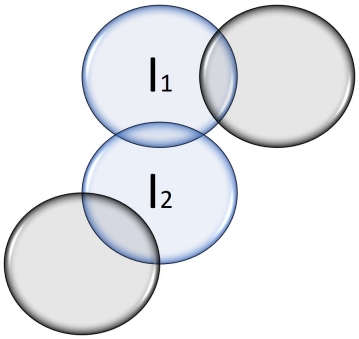


Figure 4.3: Illustration of Running Intersection Property using Venn Diagram. I_1 and I_2 are Index sets with some common elements between them. The grey sets show the possibilities for I_3.

The next figure (4.4) shows possible Index sets which can be formed while intersecting with two preexisting Index sets. From this figure, it is clear that a new index set should not intersect with more than one preexisting index sets unless, it intersects with the common space between the preexisting index sets.

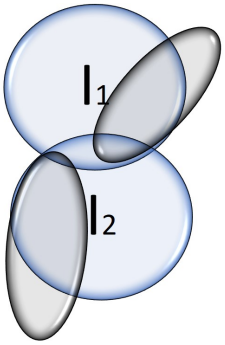


Figure 4.4: Illustration of Running Intersection Property using Venn Diagram. I_1 and I_2 are Index sets with some common elements between them. The grey sets show the possibilities for I_3.

It is important to note that, if the new Index set does intersect with more than one preexisting Index set, including elements of the other index set which are not in the common intersection space, then we need to reject it as it will violate (4.12). This idea can be illustrated by the following example and the subsequent Venn diagram fig. 4.5.

Consider state variables $X \in \mathbb{R}^4$ and let, $I_1 = \{1, 2\}$ and $I_2 = \{2, 3\}$. Now, if we want to add I_3 then, it cannot be chosen as $I_3 = \{1, 3, 4\}$, because $I_3 \cap (I_1 \cup I_2) \not\subseteq I_1$ or I_2 and hence (4.12) is not satisfied.

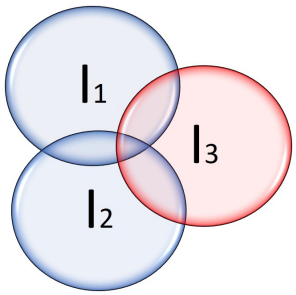


Figure 4.5: Illustration of Running Intersection Property using Venn Diagram. I_1 and I_2 are Index sets with some common elements between them. The red set is not a correct construction for I_3 as it violates (4.12).

The next figure (fig. 4.6) shows the correct propagation of Index sets when we add a new Index set.

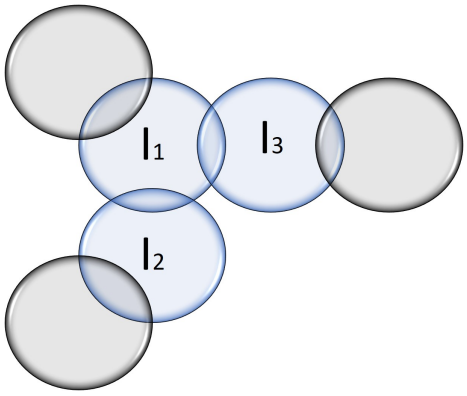


Figure 4.6: Illustration of Running Intersection Property using Venn Diagram. I_1, I_2 and I_3 are Index sets with some common elements between them. The grey sets show the possibilities for I_4. The possible Index sets involving intersection spaces between preexisting Index sets have been not shown for the sake of clarity.

Finally, it should be noted that the empty set is always a subset of all other sets and the Running Intersection Property (4.12) is always satisfied if the index sets

are disjoint.

We now present a couple of algorithms for construction of index sets. Algorithm 1 is based on the intuition that an Index set always satisfies Running Intersection Property if it has only two elements and there is only one elements which is common between two adjoining index sets. Later, it was found out that this algorithm suffers from a drawback and it works only in a specific case. Therefore, we have used the second Algorithm which looks at the problem of constructing index sets visually and is based on Graph theory. These algorithms will be used in later applications as well.

Algorithm 4.1: Construction of Index set I with at most two elements

Input : States $(x_a, \cdots, x_{last})$ and K defined by $g_{1,\cdots,m}$
Output: Index sets $I_{1,\cdots,p}$, $J_{1,\cdots,p}$ and K' defined by $g_{1,\cdots,m+p}$

Find the state x_a which has the least couplings in the dynamic model
Check whether x_a is coupled with at least one other state in the dynamic model
while x_a *is coupled with at least one more state* $(x_b, \cdots, x_{last})$ **do**
    Select x_b
    $I_k = \{a, b\}$
    $g_{m+k} = n_k M^2 - (x_a^2 + x_b^2)$
    $J_k = \{a, b, m+k\}$
    if x_b *is coupled with at least one more state* $(x_c, \cdots, x_{last})$ **then**
        $I_{k+1} = \{b, c\}$
        $J_{k+1} = \{c, m+k+1\}$
        if $I_{k+1} \cap I_k \subseteq I_s$ *for some* $s \leq k$ **then**
            repeat: until index of all states (including x_{last}) are included in Index sets
        else
            break: I_{k+1} does not satisfy Running Intersection Property.
    else
        goto: Step 3 and replace x_a with x_b
end

Thee above algorithm satisfies all the conditions and it exploits the sparsity to the maximum as each vector of monomials v_d has a dimension of only $s(d) = 6$, if we are searching for a candidate Lyapunov function having a maximum degree of 2. However, Algorithm 1 can only be used if the system has no couplings greater than 2 i.e. dynamical systems which are composed of cascaded subsystems (for example, the general dynamical system proposed in [Kha02] and [Kha15] for state

feedback stabilization using Backstepping) and thus this algorithm is applicable only for a few dynamical systems.

To overcome this drawback, we have used ideas from [Wak+06] for generalizing Algorithm 1 for dynamical systems which have couplings greater than 2 between the states. We begin by introducing some key concepts from Graph theory and then we present the Algorithm.

Graph theory

This subsection is based on [Slo13] and [BP93]. We review some basic definitions in graph theory and our aim is to define cliques which is then used in construction of Index sets.

Firstly, we define a Directed Graph.

Definition 4.4.1 *A Directed Graph (or digraph) is a pair $G = (V, E)$ where, V is the set of Vertices $\{v_1, \cdots, v_n\}$ of the digraph G and $E \subseteq V \times V$ is the set of Edges of the digraph.*

We now define an Undirected Graph which is a digraph with bidirectional edges.

Definition 4.4.2 *A graph $G = (V, E)$ is undirected if $(v_i, v_j) \in E \implies (v_j, v_i) \in E$)*

For any vertex set $S \subseteq V$, consider the edge set $E(S) \subseteq E$. E given by

$$E(S) := \{(v_i, v_j) \in E \mid (v_i, v_j) \in S\} \tag{4.14}$$

Let $G(S)$ denote the denotes the subgraph of G i.e. $G(S) = (S, E(S))$. We now define an Induced Subgraph.

Definition 4.4.3 *An Induced Subgraph is obtained by removing a set of vertices $S \subseteq V$ (and their associated edges) from the graph. Hence, we define $G \backslash S$ by*

$$G \backslash S := G(V - S) \tag{4.15}$$

Two vertices $v_i, v_j \in V$ are said to be *adjacent* if $(v_i, v_j) \in E$. An induced subgraph $G(S)$ is *complete* if the vertices in S are *pairwise adjacent* in G. In this case we also say that S is *complete* in G. We now define paths, cycles and chords as follows:

Definition 4.4.4 *A path of length k in the Graph G is denoted by $[v_0, v_1, \cdots, v_k]$ where, $v_i \neq v_j$ for $i \neq j$ and $(v_i, v_{i+1}) \in E$ for $0 \leq i \leq k-1$.*

Definition 4.4.5 *A cycle of length $k+1$ in the Graph G is denoted by $[v_0, v_1, \cdots, v_k, v_0]$ where, $v_i \neq v_j$ for $i \neq j$ and $(v_i, v_{i+1}) \in E$ for $0 \leq i \leq k$.*

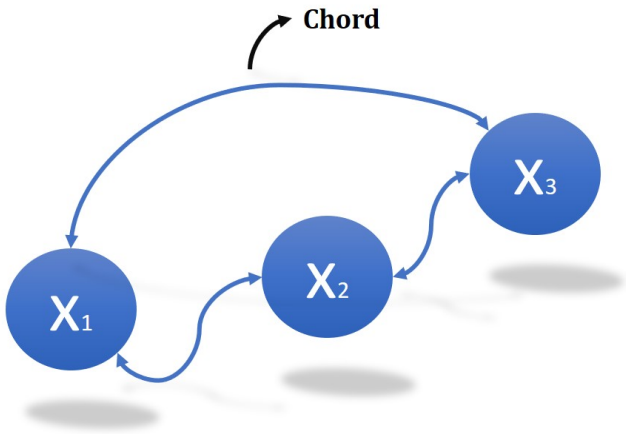


Figure 4.7: Chordal graph. A graph should atleast have 3 elements for it to have a chord.

Definition 4.4.6 *A chord of a path (cycle) is any edge joining two nonconsecutive vertices of the path (cycle). A chord is shown in fig. 4.7.*

Based on the above definitions, we now define chordal undirected graph.

Definition 4.4.7 *An undirected graph $G = (V, E)$ is chordal (triangulated, rigid circuit) if every cycle of length greater than three has a chord. A chord is shown in fig. 4.8.*

An induced subgraph of an chordal graph is also chordal.

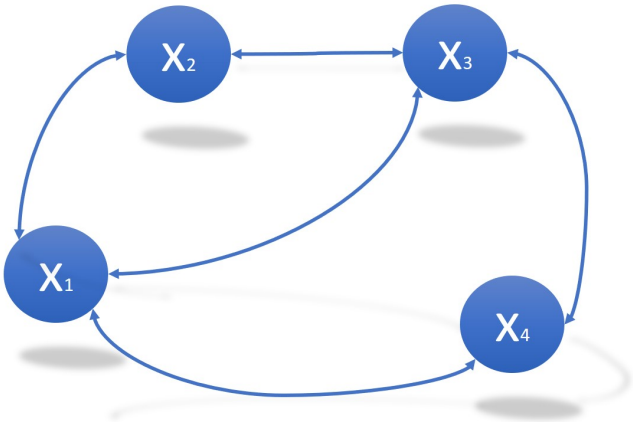


Figure 4.8: Chordal graph. A graph should atleast have 3 elements for it to have a chord.

Finally, we define cliques for a graph.

Definition 4.4.8 *A Clique $\mathcal{K}$ of a graph $G = (V, E)$ is a maximal set of vertices that is complete in G, and thus a clique is properly contained in no other clique. Fig. 4.8 shows consists of two cliques $\{x_1, x_2, x_3\}$ and $\{x_1, x_4, x_3\}$.*

We can now state a more generalized version of Algorithm 1. The idea behind this Algorithm is that for a dynamical system, the states $x \in \mathbb{R}^n$ represent the vertices of a Graph G and the Index set I can be constructed as the Maximal Cliques formed in G. Physically, the cliques may represent the subsystems in the overall dynamical system. In [Wak+06], this idea is stated in a more general form for polynomial optimization.

Algorithm 4.2: Construction of Index set I using Cliques

Input : States $(x_1, \cdots, x_{last})$ and K defined by $g_{1,\cdots,m}$
Output: Index sets $I_{1,\cdots,p}$ and K' defined by $g_{1,\cdots,m+p}$

Find the cliques $\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_{last}$
Find the the states (say $x_a, x_b, \cdots$) which connect $\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_{last}$ in the dynamical system
while *x_a is coupled with at least one more Clique $\mathcal{K}_k$* **do**
    $I_k = \{$All elements of Clique $\mathcal{K}_k\}$
    $g_{m+k} = n_k M^2 - (x_a^2 + x_b^2 + \cdots + x^2_{\text{last element of } \mathcal{K}})$
    if *$I_{k+1} \cap I_k \subseteq I_s$ for some $s \leq k$* **then**
        repeat: until index of all states (including x_{last}) are included in Index sets
    else
        break: I_{k+1} does not satisfy Running Intersection Property.
end

Compared to polynomial optimization problems in [Las10], [Las15] and [Wak+06], the dynamical systems are generally composed of a few state variables. It is unlikely to have dynamical systems with more then 15 variables and therefore, we have suggested Algorithm 2 instead of the procedure involved in polynomial optimization where, a correlation sparsity pattern matrix is constructed. Interested reader is referred to [Wak+06], [KM09] and the accompanying MATLAB toolbox SparsePOP [Wak+08].

Construction of Index set J

Suppose, we have constructed Index set I and have obtained the redundant polynomial inequalities as per Algorithm 2, we can easily construct the index set J as follows:

Algorithm 4.3: Construction of Index set J

Input : States x, Index sets I_k and semialgebraic set K' defined by $g_{1,\cdots,m+p}$
Output: Index sets $J_{1,\cdots,p}$

Start with I_1 and g_{m+1}
$J_1 = I_1 \cup \{m+1\}$
while $J_k \leq J_p$ **do**
  $J_{k+1} = (I_{k+1} \cap (\bigcup_{j=1}^{k} I_j)^{\complement}) \cup \{m+k\}$
  where $(.)^{\complement}$ denotes the compliment of the set $(.)$
end

4.5 Computationally efficient SDP for finding Lyapunov function

After applying Sparse version of Putinar's Positivstellensatz, we have obtained a more computationally efficient SDP which is listed below.

Semidefinite Program 2

$$\text{Find} \quad V(x) \tag{4.16a}$$

$$\text{such that} \quad V(0) = 0, \tag{4.16b}$$

$$V(x) - \sum_{k=1}^{p}(F_k + \sum_{j\in J_k}^{p} F_{jk}g_j) - \Gamma \|x\|_2 \in \Sigma[x]\ ,\ j = 1, \cdots, m+p \tag{4.16c}$$

$$-\frac{\partial V(x)}{\partial x} f(x) - \sum_{k=1}^{p}(F_k + \sum_{j\in J_k}^{p} F_{jk}g_j) - \Gamma \|x\|_2 \in \Sigma[x]\ ,\ j = 1, \cdots, m+p \tag{4.16d}$$

Here p denotes the number of Index sets and rest of the terminology is as given Theorem 5. SDP 2 is more computationally efficient than SDP 1 because each $F \in \mathbb{R}[x(I_k)]$ in SDP 2 instead of $F \in \mathbb{R}[x]$ in SDP 1. Thus, it is evident that smaller the Index set, more computational savings. This is further discussed in Chapter 6. In the next Chapter, we focus on applying SDP 2 for finding candidate Lyapunov functions for complex systems.

Chapter 5

Applying Sparse Putinar Positivstellensatz

Summary *This chapter focuses on application of SDP 2 for finding a candidate Lyapunov function for autonomous models of systems. We have considered an Autonomous model of a wind turbine during shutdown from [SPW12]. Thereafter, we have considered stability of the dynamics of an Adaptive Controlled system based on [Kha02] and [ÅW13]. Finally, we have found Lyapunov function for an autonomous model of Ørsted Satellite based on [Wis96] and [Wer78]. Further, it should be noted that our goal is to simply apply SDP 2 and therefore, we will keep the discussions on the models as brief as possible.*

5.1 Wind Turbine

In this section, we consider a wind turbine model during shutdown phase. Thus, the blade pitch angle β will be constant and a constant generator torque $T_g = 3,580Nm$ is applied to the rotor shaft. Compared to a Van der Pol's model of harmonic oscillator, a wind turbine is a more complex system and the dynamics are described using a nonlinear state space model with 7 states. The considered model of wind turbine with subsystems is based on a CART3 model which will be stated now without derivation. Interested reader is referred to [PS12], [ES09] and [SPW12] for further information on the derivation of wind turbine model.

A wind turbine is an interconnected system primarily composed of the following three subsystems:

- Aerodynamics
- Tower
- Drive train

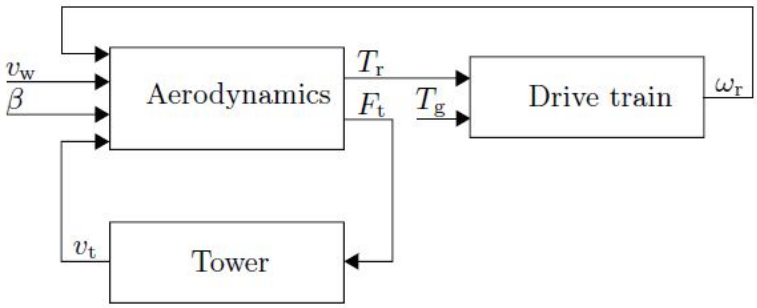


Figure 5.1: Wind turbine modeled as an interconnection of three subsystems

The interconnection of the three subsystems with coupling variables is shown in fig. (5.1). v_w represents the wind speed which is an external input to the system in the range of $15m/s$ to $25m/s$. In this case, we have assumed it to be constant at $20m/s$. β represents the pitching angle of the blades which during regular operation are aligned such that a high lift/drag ratio is achieved which gives high torque. During shutdown, the blades are pitched at an angle of $\beta = 90^\circ$, such that a negative torque is obtained which deaccelerates the rotor. Additionally, during shutdown a constant generator torque of T_g = 3580 N-m is also applied to speed up the deacceleration of the rotor until the rotor speed is below a threshold of 0.77 rad/s, at which point, it is not possible to apply a torque from the generator and hence, the wind turbine is left uncontrolled at that point of shutdown. T_r is the torque exerted by the wind via the aerodynamics model. F_t is the force exerted on the tower due to the wind. ω_r is the rotor speed. T_r and F_t depends on ω_r, β and the wind speed at the rotor given by $v_w - v_t$, where v_t is the velocity of the tower due to aerodynamic forces on it. These relations a given by C_p and C_t which are generally described by lookup tables. However, we have characterized C_p and C_t by polynomials p_1 and p_2 which are stated as follows.

$$p_1 = (c_{11} + c_{12}\omega_{r,f} + c_{13}v_r + c_{14}\omega_{r,f}^2 + c_{15}v_r^2 + c_{16}\omega_{r,f}v_r)v_r^3 \tag{5.1}$$

$$p_2 = (c_{21} + c_{22}\omega_{r,f} + c_{23}v_r + c_{24}v_r^2 + c_{25}\omega_{r,f}v_r)v_r^2 + c_{26} + c_{25}\omega_{r,f}^2 \tag{5.2}$$

where, $\omega_{r,f}$ is the rotor speed generated from filtering the wind speed. These polynomials have been obtained by a least squares approximation of C_p and C_t tables. Further information on the same can be obtained in [PS12].

We now state T_r and F_t based on the polynomials p_1 and p_2 respectively.

$$T_r = \frac{1}{2}\rho A R v_w^2 p_1 \tag{5.3}$$

$$F_t = \frac{1}{2}\rho A v_w^2 p_2 \tag{5.4}$$

where, ρ is the density of air, A is the area swept by the blades and R is the radius of the area swept by the blades.

We now state the dynamic model of the wind turbine

$$\begin{bmatrix} \dot{v}_r \\ \dot{\omega}_{r,f} \end{bmatrix} = \begin{bmatrix} -c_{v_r} v_r + (v_w - v_t) \\ -c_{\omega_{r,f}} \omega_{r,f} + \omega_r \end{bmatrix} \tag{5.5a}$$

$$h_1 = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \tag{5.5b}$$

$$\begin{bmatrix} \dot{v}_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} \frac{1}{M_t}(F_t - B_t v_t - k_t x_t) \\ v_t \end{bmatrix} \tag{5.5c}$$

$$h_2 = v_t \tag{5.5d}$$

$$\begin{bmatrix} \dot{\omega}_r \\ \dot{\theta}_\Delta \\ \dot{\omega}_g \end{bmatrix} = \begin{bmatrix} \frac{1}{J_r}(T_r - k_r\theta_\Delta - B_r(\omega_r - \frac{1}{N_g}\omega_g)) \\ \omega_r - \frac{1}{N_g}\omega_g \\ \frac{1}{J_g}(\frac{1}{N_g}(k_r\theta_\Delta - B_r(\omega_r - \frac{1}{N_g}\omega_g)) - T_g) \end{bmatrix} \tag{5.5e}$$

$$h_3 = \omega_r \tag{5.5f}$$

where, h_1, h_2 and h_3 represent the interconnecting outputs of subsystems 1, 2 and 3 respectively. We will now state the parameters used in the wind turbine model (5.5) along with their physical description. A complete list of descriptions of all the state used in the model (along with the considered regions) can be found in Table 5.3.

Symbols	**Description**	**Value**	**Unit**
M_t	Mass of the tower	7.76×10^3	kg
B_t	Tower damping coefficient	18.6	$kN/(m/s)$
k_t	Tower torsion coefficient	2.7	MN/m
N_g	Drive train gear ratio	43	$-$
J_g	Moment of inertia of the high-speed shaft	534.12	$kg\text{-}m^2$
J_r	Moment of inertia of the low-speed shaft	611.1×10^3	$kg\text{-}m^2$
B_r	Viscous friction of the low-speed shaft	24	$kN\text{-}m/(rad/s)$
k_r	Torsion stiffness of the low-speed shaft	24.7×10^6	$N\text{-}m/rad$

Table 5.1: Parameters (with dimensions) for the wind turbine model (5.5)

Besides the parameters mentioned in table 5.1, the dynamic model also consists of dimensionless constants c_* which have no physical significance. They have been stated in table 5.2 along with their suitable values.

Symbols	Value
c_{v_r}	11.65
$c_{\omega_{r,f}}$	21
c_{11}	-32.42×10^6
c_{12}	-746×10^6
c_{13}	53.03×10^6
c_{14}	-1.128×10^9
c_{15}	-18.63×10^6
c_{16}	-384.6×10^6
c_{21}	8492.6
c_{22}	300.88×10^3
c_{23}	-11.85×10^3
c_{24}	3584
c_{25}	-90.32×10^3
c_{26}	318.3
c_{27}	1.692×10^6

Table 5.2: Dimensionless parameters for the wind turbine model (5.5)

We now consider a region of state space which can be defined by a semialgebraic set. Our aim is to find a Lyapunov function which satisfies the conditions (2.16) in the semialgebraic set.

The considered region of state space is given as follows:

Description	Symbol	Region
Rotor velocity	v_r	$[2, 28]$
Rotor angular velocity (filtered)	$\omega_{r,f}$	$[0.77, 44]$
Tower velocity	v_t	$[-0.01, 0.07]$
Tower displacement	x_t	$[-0.05, 0.05]$
Rotor angular velocity	ω_r	$[0.77, 44]$
Drive train torsion angle	θ_Δ	$[-25, 25] \times 10^{-3}$
Generator angular velocity	ω_g	$[33.2, 172.7]$

Table 5.3: Description of States and considered State space region

Based on (5.3), the semialgebraic set K is constructed as follows:

$$K = \{x \in \mathbb{R}^n : g_j(x) \geq 0, j = 1, \cdots, 7\} \tag{5.6a}$$

where,

$$g_1 = (x_1 - 2) \times (28 - x_1) \tag{5.6b}$$
$$g_2 = (x_2 - 0.77) \times (4 - x_2) \tag{5.6c}$$
$$g_3 = (x_3 + 0.01) \times (0.07 - x_3) \tag{5.6d}$$
$$g_4 = (x_4 + 0.05) \times (0.05 - x_4) \tag{5.6e}$$
$$g_5 = (x_5 - 0.77) \times (4 - x_5) \tag{5.6f}$$
$$g_6 = (x_6 + 25 \times 10^{-3}) \times (25 \times 10^{-3} - x_6) \tag{5.6g}$$
$$g_7 = (x_7 - 33.2) \times (172.7 - x_7) \tag{5.6h}$$

We begin by stating the dynamics (5.5) of wind turbine in a simpler state space form which highlights the couplings between the states.

$$\dot{x}_1 = f_1(x_1, x_3) \tag{5.7a}$$
$$\dot{x}_2 = f_2(x_2, x_5) \tag{5.7b}$$
$$\dot{x}_3 = f_3(x_1, x_2, x_3, x_4) \tag{5.7c}$$
$$\dot{x}_4 = f_4(x_3) \tag{5.7d}$$
$$\dot{x}_5 = f_5(x_1, x_2, x_5, x_6, x_7) \tag{5.7e}$$
$$\dot{x}_6 = f_6(x_5, x_7) \tag{5.7f}$$
$$\dot{x}_7 = f_7(x_6, x_7) \tag{5.7g}$$

We can now construct Index sets for (5.7) using Algorithms 2 and 3. We cannot use Algorithm 1 in this case, as the wind turbine model is composed of cliques (see fig. 5.2), instead of being composed of cascaded subsystems.

From fig (5.2), it is evident that the states x_5 and x_3 connect the subsystems, Drive Train (x_5, x_6 and x_7) and the Tower (x_3 and x_4) via the Aerodynamics subsystem (x_1 and x_2) respectively.

The following index sets were obtained for the wind turbine model (5.7):

$$I_1 = \{1, 2, 3, 5\}, \quad g_8 = 4M^2 - (x_1^2 + x_2^2 + x_3^2 + x_5^2), \quad J_1 = \{1, 2, 3, 5, 8\} \tag{5.8a}$$
$$I_2 = \{3, 4\} \quad , \quad g_9 = 2M^2 - (x_3^2 + x_4^2) \quad , \quad J_2 = \{4, 9\} \tag{5.8b}$$
$$I_3 = \{5, 6, 7\} \quad , \quad g_{10} = 3M^2 - (x_5^2 + x_6^2 + x_7^2) \quad , \quad J_3 = \{6, 7, 10\} \tag{5.8c}$$

Note that, M in g_8, g_9 and g_{10} is the maximum value attainable by any of the state. Looking at g_7 in 5.6h, it is evident that the maximum possible value attainable by any state is 172.7. We therefore, select $M = 172.7$. This ensures compactness

of the semialgebraic set as per Theorem 5.

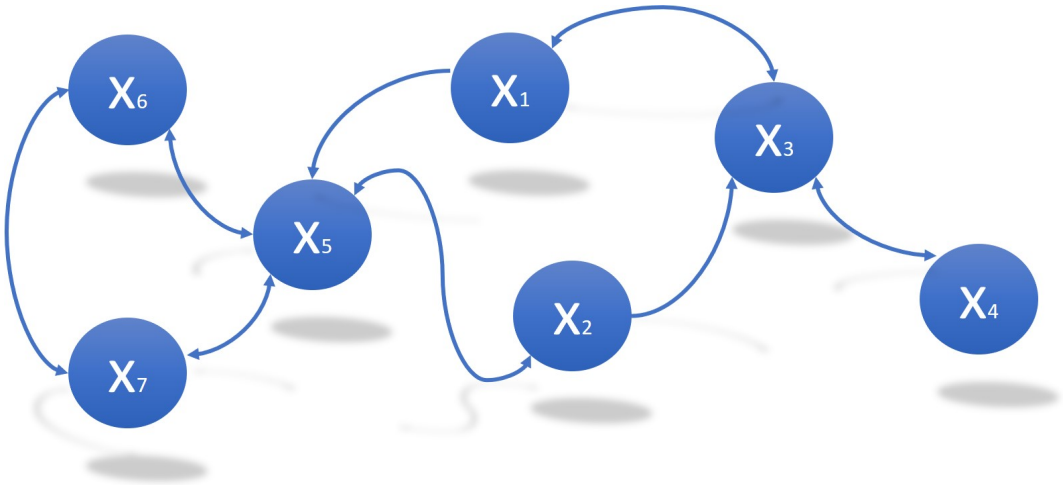


Figure 5.2: Graphical representation of the dynamics for the wind turbine model (5.5)

Using (5.8) and (4.13), we can now write an representation for finding candidate Lyapunov function V which is guaranteed to be positive in K.

$$\begin{aligned} V &= \sum_{k=1}^{p} \left(q_k + \sum_{j \in J_k} q_{jk} g_j \right) \\ &= q_1 + q_{11} g_1 + q_{21} g_2 + q_{31} g_3 + q_{51} g_5 + q_{81} g_1 + \\ &\quad q_2 + q_{42} g_4 + q_{92} g_9 + q_3 + q_{63} g_6 + q_{73} g_7 + q_{103} g_{10} \end{aligned} \tag{5.9}$$

Similarly we obtain a representation for the negative Lie derivative $-\frac{\partial V(x)}{\partial x} f(x)$ which ensures that the condition (2.16c) is satisfied. Substituting V obtained from (5.9) into SDP 2, we have obtained the following candidate Lyapunov function:

$$\begin{aligned} &- 364625036.89 * x_1 + 6736099.77835 * x_2 - 0.0349697674069 * x_3 \\ &+ 6638022.48774 * x_4 + 968605484.225 * x_6 + 162937.980549 * x_7 \\ &+ 1845916378.31 * x_2^2 + 295850773.632 * x_1^2 - 143907819.635 * x_1 * x_2 \\ &+ 22491497.3928 * x_4^2 + 1595109060.44 * x_6^2 + 1367.3279238 * x_7^2 \\ &- 2613108.88038 * x_6 * x_7 + 5830855.18081 * x_1 * x_4 - 62652.9344278 * x_2 * x_4 \\ &- 94518551.544 * x_1 * x_6 - 189586625.553 * x_2 * x_6 + 4174790.86017 * x_4 * x_6 \\ &- 246539.560875 * x_1 * x_7 - 207318.002999 * x_2 * x_7 - 5771.09408269 * x_4 * x_7 \end{aligned} \tag{5.10}$$

After obtaining the Lyapunov function in (5.10), we would like to verify whether the obtained function satisfies all the constraints and we will discuss that in the section 6.1 in Chapter 6.

5.2 Adaptive Control

In this section we apply SoS and Sparse Putinar's Positivstellensatz to prove stability for an autonomous adaptive control system. We begin by defining the dynamic model for the model reference adaptive control. We have considered an Adaptive Control system for this project as the non-autonomous model (5.17) is a 3^{rd} order nonlinear system which is the minimum order required for application of Sparse Putinar's Positivstellensatz. Later, we have discussed an autonomous model of Adaptive control which is a higher order system. The model is based on [ÅW13] and [Kha02]. Consider a 1^{st} order linear system

$$\dot{y}_p = a_p y_p + K_p u \tag{5.11}$$

where, subscript $_p$ indicates plant, u is the control input to the plant, y_p is the measured output from the plant, a_p and k_p are unknown plant parameters. Suppose that it is desirable to obtain a closed loop system where input-output behavior is described by the following reference model.

$$\dot{y}_m = a_m y_m + K_m r \tag{5.12}$$

where, r is the reference input and $y_m(t)$ represents the desired output of the closed loop system. This can be achieved by a linear feedback control law

$$u(t) = \theta_1(t) r(t) + \theta_2(t) y_p(t) \tag{5.13}$$

where the time varying gains $\theta_1(t)$ and $\theta_2(t)$ represent adjustable controller parameters which are adjusted online.

From (5.11), (5.12) and (5.13), it can be seen that the ideal values of $\theta_1(t)$ and $\theta_2(t)$ should be $\theta_1^*(t)$ and $\theta_2^*(t)$ which are given as follows:

$$\theta_1^*(t) = \frac{k_m}{K_p} \quad \text{and} \quad \theta_2^*(t) = \frac{a_m - a_p}{K_p} \tag{5.14}$$

where $k_p \neq 0$.

We now choose an adaptation rule such that $\theta_1(t)$ and $\theta_2(t)$ converge to $\theta_1^*(t)$ and $\theta_2^*(t)$. One such rule is the gradient algorithm which is stated as follows:

$$\dot{\theta}_1 = -\gamma(y_p - y_m) r \tag{5.15a}$$

$$\dot{\theta}_2 = -\gamma(y_p - y_m) y_p \tag{5.15b}$$

where γ is a positive constant which determines the speed of adaptation. This adaptive control law assumes that k_p is positive.

We now define output error and parameter errors e_0, ϕ_1 and ϕ_2 as follows:

$$e_0 = y_p - y_m \quad , \quad \phi_1 = \theta_1 - \theta_1^* \quad \text{and} \quad \phi_2 = \theta_2 - \theta_2^* \tag{5.16}$$

The closed loop system can now be described by a nonlinear, 3^{rd} order system

$$\dot{e}_0 = a_m e_0 + k_p \phi_1 r(t) + k_p \phi_2 [e_0 + y_m(t)] \tag{5.17a}$$
$$\dot{\phi}_1 = -\gamma e_0 r(t) \tag{5.17b}$$
$$\dot{\phi}_2 = -\gamma e_0 [e_0 + y_m(t)] \tag{5.17c}$$

This appears to be a simpler system compared to (5.5). However, it is non-autonomous in nature because the reference $r(t)$ is user-defined. The reference input needs to be *persistently exciting* of the order equal to the model order, so as to ensure convergence of the parameters ϕ_1 and ϕ_2 to the origin [ÅW13].

Since, the considered plant model (5.11) has an order of 1, a single sinusoidal reference will be sufficiently persistently exciting. We have obtained a candidate Lyapunov function for the non-autonomous system with a sinusoidal input and later we will discuss, the autonomous variant of adaptive controlled system.

Firstly, we consider the semialgebraic set K for this system. We have obtained the state space region for the Adaptive control system by simulating it with a sinusoidal reference.

$$K = \{x \in \mathbb{R}^n : g_j(x) \geq 0, j = 1, \cdots, 3\} \tag{5.18a}$$
$$\text{where,} \quad g_1 = x_1^2 - 0.2771297 \tag{5.18b}$$
$$g_2 = x_2^2 - 16 \tag{5.18c}$$
$$g_3 = x_3^2 - 16 \tag{5.18d}$$

As can be seen from fig. (5.3), there are no cliques in the graph and thus, we can apply Algorithm 1 for constructing Index sets as follows:

$$I_1 = \{1, 2\}, \quad g_4 = 2M^2 - (x_1^2 + x_2^2), \quad J_1 = \{1, 2, 4\} \tag{5.19a}$$
$$I_2 = \{1, 3\}, \quad g_5 = 2M^2 - (x_1^2 + x_3^2), \quad J_2 = \{3, 5\} \tag{5.19b}$$

We can write a sparse representation of the candidate Lyapunov function which is guaranteed to be positive on the semialgebraic set K.

$$\begin{aligned} V &= \sum_{k=1}^{p} \left(q_k + \sum_{j \in J_k} q_{jk} g_j \right) \\ &= q_1 + q_{11} g_1 + q_{21} g_2 + q_{41} g_4 + q_2 + q_{32} g_3 + q_{52} g_5 \end{aligned} \tag{5.20}$$

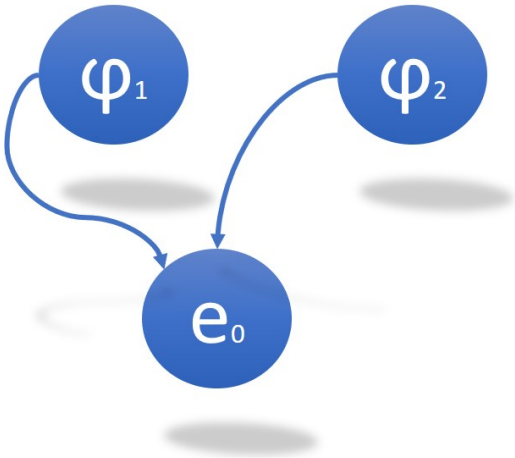


Figure 5.3: Graphical representation of the nonlinear dynamics for adaptive control (5.17)

Similarly we can write a sparse representation for the negative Lie derivative $-\frac{\partial V(x)}{\partial x} f(x)$ which ensures that the condition (2.16c) is satisfied. Substituting V obtained from (5.9) into SDP 2, we have obtained the following candidate Lyapunov function:

$$\begin{aligned} &- 0.0197193466133 * x_1 - 16.0313864599 * x_2 + 21.3694746707 * x_3 \\ &- 0.00741910396354 * x_1 * x_3 + 489.643177153 * x_1^2 + 244.064576396 * x_2^2 \\ &+ 244.541999491 * x_3^2 + 0.00596493998992 * x_1 * x_2 + 0.776667515418 * x_2 * x_3 \end{aligned} \tag{5.21}$$

For the sake of generalization, we would like to make the system autonomous. As stated earlier, this system can be made autonomous at the cost of adding extra states to the system which define the reference $r(t)$ via some persistently exciting function of the order n. For a sinusoidal reference, we need to add two more states as a model of harmonic oscillator (5.22) is a 2^{nd} order model.

$$\ddot{x} = x \tag{5.22}$$

As a result of adding, (5.22) to (5.17), we got a 5^{nd} order model which does not have a single equilibrium point but instead a periodic orbit due to (5.22). Analysis of such a system is outside the scope of this project, as it requires specialized theorems because Lyapunov's Stability theorem 3 is valid only for equilibrium point.

5.3 Ørsted Satellite

In this section, we apply sparse version of Putinar's Positivstellensatz and SoS to obtain a candidate Lyapunov function for a satellite stabilized via a PD controller. The motivation behind considering a satellite in this project is that, the motion of a satellite is defined by a nonlinear model with 7 states viz. 4 quaternions and 3 angular velocities. Further, quaternions have some special constraints which we will be discussing in this section. We begin by giving a brief overview of the satellite coordinate system and quaternions. This section is based on [Wer78], [TAM10], [Wis99] and [Wis96].

The satellite's body is a box shaped, 680 mm (length) $\times$ 450 mm (breadth) $\times$ 340 mm (height). Further, during the normal operation of the satellite, an $8m$ long scientific boom is deployed (see fig. 5.4). For simplification, we are only considering the control of satellite during normal operation and have ignored the detumbling and boom upside-down phase which have been studied in [Wis96].

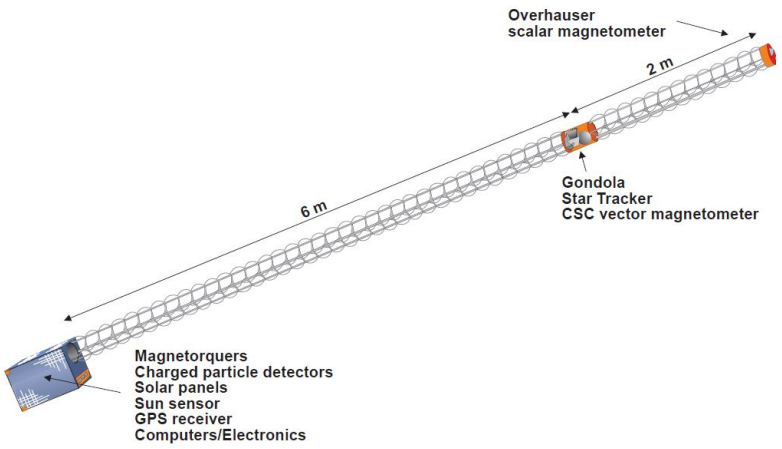


Figure 5.4: The Ørsted Satellite as described in [Wis96]

5.3.1 Coordinate systems

We have used four coordinate systems (CS) for defining satellite motion and control. They are defined as follows:

- **Control CS**
  The Control CS is a right orthogonal coordinate system with the origin at the center of mass of the spacecraft. Its a carefully chosen CS such that, the x axis has the maximum moment of inertia, and the z axis has the minimum moment of inertia.

- **Body CS**
  The Body CS is a right orthogonal coordinate system with the origin at the centre of gravity of the spacecraft body. The z axis is parallel to the boom direction and points towards the tip of the boom. The x axis is perpendicular to the breadth of the satellite, and points away from the boom. The y axis is perpendicular to the length of the satellite.

- **Orbit CS**
  The Orbit CS is a right orthogonal coordinate system with the origin at the centre of mass of the spacecraft body while in an orbit. The z axis points at the zenith i.e. it is aligned with the centre of the Earth and points away from the Earth towards the satellite. The x axis points in the orbital plane normal direction and its direction coincides with the direction of the orbital angular velocity vector. The Orbit CS is the reference for the attitude control system.

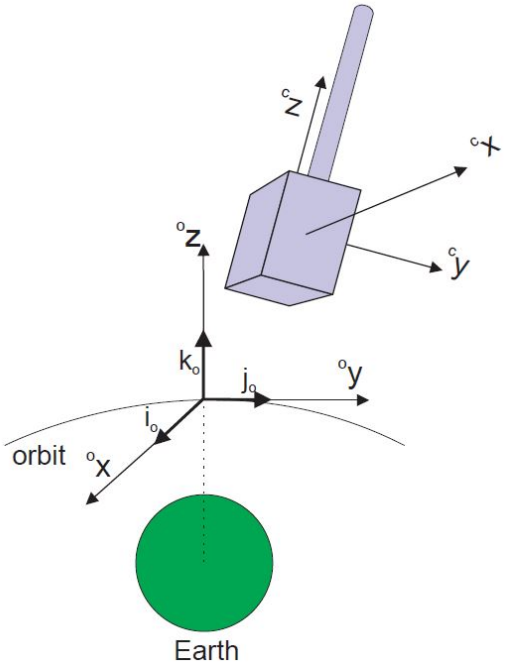


Figure 5.5: Control CS and Orbital CS for the satellite as described in [Wis96]

- **World CS**
  The World CS is an earth centered inertial right orthogonal coordinate system. The z axis is parallel to the rotation axis of the Earth and points towards the North Pole. The x axis is parallel to the line connecting the centre of the Earth with Vernal Equinox and points towards Vernal Equinox.

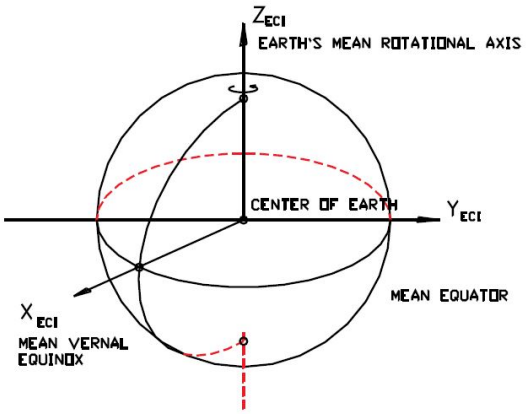


Figure 5.6: World CS which is an Earth centered Inertial CS (abbreviated as ECI) for the satellite as described in [Wis99]

5.3.2 Modeling of the Satellite with Quaternions

Rotation of a coordinate system can be defined via quaternions which provide a convenient product rule for describing successive rotations. Intuitively, rotation can also be described via the three Euler angles viz. Pitch, Roll and Yaw. However, the minimum number of parameters required for global attitude representation is four [Wis99] and the problem of gimbal lock associated with Euler angles is also avoided.

A quaternion $\mathbf{q}$ consists of four parameters $[q_1\ q_2\ q_3\ q_4]^T$ of which the first three $[q_1\ q_2\ q_3]^T$ components, form the vector part of the quaternion and the last component q_4, forms the scalar part of the quaternion. Thus, $\mathbf{q}$ can be written as follows:

$$\mathbf{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k} + q_4 \tag{5.23}$$

where, $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are hyper imaginary numbers which satisfy the following condition

$$\begin{aligned} \mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 &= -1 \\ \mathbf{ij} = -\mathbf{ji} &= \mathbf{k} \\ \mathbf{jk} = -\mathbf{kj} &= \mathbf{i} \\ \mathbf{ki} = -\mathbf{ik} &= \mathbf{j} \end{aligned} \tag{5.24}$$

The inverse of quaternion $\mathbf{q}$ is defined as

$$\mathbf{q}^* = -q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k} \tag{5.25}$$

The norm of quaternion $\mathbf{q}$ is defined as

$$|\mathbf{q}| = \sqrt{\mathbf{q}\mathbf{q}^*} = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} \tag{5.26}$$

We now construct a unit quaternion having a norm 1 which gives the rotation of an angle Φ around a unit vector $\epsilon = [\epsilon_1 \ \epsilon_2 \ \epsilon_3]^T$. These parameters are also known as Euler symmetric parameters. The four parameters of the quaternion $\mathbf{q}$ can be defined as follows:

$$\begin{aligned} \mathbf{q_1} &:= \epsilon_1 \sin\frac{\Phi}{2} \\ \mathbf{q_2} &:= \epsilon_2 \sin\frac{\Phi}{2} \\ \mathbf{q_3} &:= \epsilon_3 \sin\frac{\Phi}{2} \\ q_4 &:= \cos\frac{\Phi}{2} \end{aligned} \tag{5.27}$$

Clearly, the Euler symmetric parameters satisfy the constraint,

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1 \tag{5.28}$$

Suppose, we want to find the rotation of the satellite in the World CS and the quaternions, ${}^c_o\mathbf{q}$ (describing the transformation Control CS to the Orbit CS) and ${}^o_w\mathbf{q}$ (describing the transformation Orbit CS to the World CS) are given then,

$${}^c_w\mathbf{q} = \mathbf{R}({}^o_w\mathbf{q}){}^c_o\mathbf{q} \tag{5.29}$$

where,

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \tag{5.30}$$

The matrix $\mathbf{R}(\mathbf{q})$ is referred to as the quaternion product matrix and is used for finding successive rotations. Further, we can write

$$\mathbf{R}(\mathbf{q})\mathbf{R}^T(\mathbf{q}) = \mathbf{R}^T(\mathbf{q})\mathbf{R}(\mathbf{q}) = \mathbf{q}^T\mathbf{q}\mathbf{I}_{4\times 4} \tag{5.31}$$

We now define the transformation matrix $\mathbf{A}$ (also referred to as the Attitude Matrix or the direction cosine matrix) for the transformation between the Orbit CS and the Control CS.

$${}^c_o\mathbf{A} = [{}^c_o\mathbf{i} \quad {}^c_o\mathbf{j} \quad {}^c_o\mathbf{k}] \tag{5.32}$$

where, ${}^c_o\mathbf{i}$, ${}^c_o\mathbf{j}$ and ${}^c_o\mathbf{k}$ are the unit vectors representing the transformation of x, y and z axis of the Orbit CS into the Control CS. We can now parametrize, the unit vectors in terms of the attitude quaternion.

$$\begin{aligned} {}^c_o\mathbf{i} &= [\ q_1^2 - q_2^2 - q_3^2 + q_4^2 \quad 2(q_1q_2 - q_3q_4) \quad 2(q_1q_3 + q_2q_4)\ ]^T \\ {}^c_o\mathbf{j} &= [\ 2(q_1q_2 + q_3q_4) \quad -q_1^2 + q_2^2 - q_3^2 + q_4^2 \quad 2(q_2q_3 - q_1q_4)\ ]^T \\ {}^c_o\mathbf{k} &= [\ 2(q_1q_3 - q_2q_4) \quad 2(q_2q_3 + q_1q_4) \quad -q_1^2 - q_2^2 + q_3^2 + q_4^2\ ]^T \end{aligned} \tag{5.33}$$

Transformation of a vector $\mathbf{v}$ observed in Orbit CS (${}^o\mathbf{v}$) to Control CS (${}^c\mathbf{v}$) can now be done as

$${}^c\mathbf{v} = {}^c_o\mathbf{A}\,{}^o\mathbf{v} \tag{5.34}$$

Kinematics Model

The Kinematics can be represented in terms of the Quaternion product matrix (5.30) as follows

$${}^c_o\dot{\mathbf{q}} = \frac{1}{2}\mathbf{R}({}^c\tilde{\mathbf{\Omega}}_{co})\,{}^c_o\mathbf{q} \tag{5.35}$$

where, $\mathbf{R}(\cdot)$ is Quaternion Product as given in (5.30) and ${}^c\tilde{\mathbf{\Omega}}_{co} = [{}^c\mathbf{\Omega}_{co}^T 0]^T$. ${}^c\mathbf{\Omega}_{co}$ is the angular velocity in the Control CS with respect to Orbit CS and it can be resolved in the World CS as follows:

$${}^c\mathbf{\Omega}_{co} = {}^c\mathbf{\Omega}_{cw} - \omega_o\,{}^c_o\mathbf{i} \tag{5.36}$$

where, ω_o is the orbital rate which we have assumed to be constant as the eccentricity of the spacecraft orbit is negligible.

Dynamics Model

The dynamics describe the relation between the satellite's angular momentum with the torques acting on the spacecraft. This is known as the Newtonian approach since we are essentially writing Newton's 2^{nd} law of motion which relates the force with the change of momentum. We obtain a set of three differential equations which are known as Euler's equations of motion and they can be written as follows:

$$\mathbf{I}^c\dot{\mathbf{\Omega}}_{cw}(t) = -{}^c\mathbf{\Omega}_{cw}(t) \times \mathbf{I}^c\mathbf{\Omega}_{cw}(t) + {}^c\mathbf{N}_{ctrl}(t) + {}^c\mathbf{N}_{gg}(t) + {}^c\mathbf{N}_{dist}(t) \tag{5.37}$$

where, ${}^c\mathbf{N}_{ctrl}(t)$ is the control torque, ${}^c\mathbf{N}_{gg}(t)$ is the torque due to gravity gradient and ${}^c\mathbf{N}_{dist}(t)$ is the torque due to external disturbances.

We begin, by defining ${}^c\mathbf{N}_{ctrl}(t)$ which is generated due to the interaction between the geomagnetic field and the magnetic moment $m(t)$, generated due to the magnetorquer current $i(t)$ as follows.

$$ {}^c\mathbf{N}_{ctrl}(t) = {}^c\mathbf{m}(t) \times {}^c\mathbf{B}(t) \tag{5.38} $$

where, ${}^c\mathbf{B}(t)$ is the geomagnetic field and we have used the dipole model with IGRF coefficients given in [Wer78] for this simulation. It should be noted, that ${}^c\mathbf{B}(t)$ is dependent upon the position of the satellite and thus should be transformed into Control CS using the Attitude matrix, ${}^c_o\mathbf{A}$ given in (5.32)

${}^c\mathbf{m}(t)$ is the magnetic moment generated by the satellite and it serves as the control signal for this system. It is generated via electromagnetic coils housed inside the satellite body as follows.

$$ \mathbf{m}(t) = n_{coil} i_{coil}(t) A_{coil} \tag{5.39} $$

where, n_{coil}, $i_{coil}(t)$ and A_{coil} are the number of winding's of the coil, current in the coil and area of the coil respectively.

Since, the winding's are housed inside the body of the satellite, we have to transform (5.39) from body CS to Control CS as follows.

$$ {}^c\mathbf{m} = {}^c_b\mathbf{A}\,{}^b\mathbf{m} \tag{5.40} $$

Unlike, other LEO satellites, the Ørsted Satellite is actuated only via Magnetorquers and therefore, there are no additional actuators, such as reaction wheels described in [Wis99] and [Wer78].

The gravity gradient torque ${}^c\mathbf{N}_{gg}(t)$ is given as follows.

$$ {}^c\mathbf{N}_{gg}(t) = 3\omega_o^2({}^c\mathbf{k}_o \times \mathbf{I}\,{}^c\mathbf{k}_o) \tag{5.41} $$

Finally, we have the disturbance torque which is primarily due to the aerodynamic drag since its a LEO satellite. Disturbance due to residual magnetic field is already compensated while estimating states and disturbance due to Solar radiation is negligible enough in comparison to the aerodynamic drag and so it can be neglected. Based on [BW97], we can write ${}^c\mathbf{N}_{dist}(t)$ as follows.

$$ {}^c\mathbf{N}_{dist}(t) = \frac{1}{2} C_D \rho v_0^2 \sum_{i=1}^{k} A_i(\hat{\mathbf{n}}_i \cdot \hat{\mathbf{v}}_b)\hat{\mathbf{v}}_b \times \mathbf{r}_i \tag{5.42} $$

where A_i is the surface area of one of the faces of the satellite (assuming the satellite's main body to be a box, there are three surfaces so $k = 3$), $\hat{\mathbf{n}}_i$ is the outward normal to the considered surface, $\hat{\mathbf{v}}_b$ is the unit vector in the direction of the

translational velocity of the considered surface element, C_D is the drag coefficient and ρ is air density.

5.3.3 Finding Candidate Lyapunov function

In the previous subsection, we obtained the kinematics and the dynamical model of the Ørsted Satellite, Now, we can find the candidate Lyapunov function for the same. We begin by defining the semialgebraic set as follows.

$$K = \{x \in \mathbb{R}^n : g_j(x) \geq 0, j = 1, \cdots, 7\} \tag{5.43a}$$

where,

$$g_1 = x_1^2 - 1 \tag{5.43b}$$
$$g_2 = x_2^2 - 1 \tag{5.43c}$$
$$g_3 = x_3^2 - 1 \tag{5.43d}$$
$$g_4 = x_4^2 - 1 \tag{5.43e}$$
$$g_5 = x_5^2 - 0.1 \tag{5.43f}$$
$$g_6 = x_6^2 - 0.1 \tag{5.43g}$$
$$g_6 = x_7^2 - 0.1 \tag{5.43h}$$

Since the quaternions are bound by the constraint (5.28), g_1, g_2, g_3 and g_4 are constrained by 1 in (5.43). Further, angular speed $\mathbf{\Omega}$ is of the magnitude of 0.001 m/s and conservatively, we have constrained it by 0.1. We now, construct Index sets based on Algorithm 2 and 3 as the dynamics reveal two cliques $\mathcal{K}_1$ consisting of the four quaternion $\mathbf{q}$ states (see (5.35)) and $\mathcal{K}_2$ consisting of the remaining three angular velocity $\mathbf{\Omega}$ states (see (5.37)). Thus, we have constructed the following Index sets.

$$I_1 = \{1, 2, 3, 4\}, \quad g_8 = 4M^2 - (x_1^2 + x_2^2 + x_3^2 + x_4^2), \quad J_1 = \{1, 2, 3, 4, 8\} \tag{5.44a}$$
$$I_2 = \{5, 6, 7\} \quad , \quad g_9 = 3M^2 - (x_5^2 + x_6^2 + x_7^2) \quad , \quad J_2 = \{5, 6, 7, 9\} \tag{5.44b}$$

Based on the Index sets (5.44), we can now write the Sparse Putinar's Positivstellensatz based representation of the candidate Lyapunov function as follows.

$$\begin{aligned} V &= \sum_{k=1}^{p} \left(q_k + \sum_{j \in J_k} q_{jk} g_j \right) \\ &= q_1 + q_{11} g_1 + q_{21} g_2 + q_{31} g3 + q_{41} g_4 + q_{81} g_8 + \\ &\quad q_2 + q_{52} g_5 + q_{62} g_6 + q_{72} g_7 + q_{92} g_9 \end{aligned} \tag{5.45}$$

Similarly we can write a sparse representation for the negative Lie derivative $-\frac{\partial V(x)}{\partial x} f(x)$ which ensures that the condition (2.16c) is satisfied. Substituting V obtained from (5.45) into SDP 2, we have obtained the following candidate Lyapunov function:

$$2.01726347735 * q_1 + 1.8769208317 * q_2 + 1.97655932521 * q_3 + 3.1690686905 * q_4 + 0.0277894915149 * w_1 + 0.00240131425285 * w_2 - 0.0138054230665 * w_3 + 2.74577385031 * q_1^2 + 3.40998675508 * q_2^2 + 2.87167813837 * q_3^2 + 3.20115868719 * q_4^2 + 0.554720673841 * q_1 * q_2 + 1.36313094243 * q_3 * q_4 + 1.42805892686 * q_1 * q_3 + 1.20305142048 * q_2 * q_4 + 0.338689316864 * q_2 * q_3 + 1.39663370578 * q_1 * q_4 \tag{5.46}$$

5.4 Conclusion

In this chapter, we have demonstrated how a candidate Lyapunov function can be found for complex dynamic systems. However, we still need to verify these candidate Lyapunov functions and that has been done in Chapter 6. Further, it is important to note, that Lyapunov's Satbility theorem works only for equilibrium point and it has to be the origin of the system. If a system has an equilibrium point at some other point instead of the origin, we can do change of variables by subtracting the equilibrium point coordinates from the states (i.e. shift the coordinate axis). If a system doesn't have an equilibrium point but instead a limit cycle or a periodic equilibrium point then the above methods cannot be applied.

Chapter 6

Computational considerations

Summary *In this chapter, we focus on computational considerations and practical issues while solving SDP's 1 and 2. We firstly introduce a method for verification of candidate Lyapunov functions based on [Löf09] and [Löf11] for YALMIP SoS module. Another YALMIP SoS module feature under consideration is symmetry reduction. Thereafter, we introduce some well-known metrics used to quantify computational complexity and state the theoretical time complexity for SoS, SDP, Putinar's Positivstellensatz and its Sparse version. We have also compared the computational savings achieved by applying SDP 2 versus application of SDP 1 based on number of parametric variables, number of constraints, Solver execution time and the size of SoS multipliers. As expected theoretically, we have obtained computational savings, when sparsity in system dynamics is exploited while finding the candidate Lyapunov function. In the last section, we have given a breakdown of the total number of parametric variables used and have shown the reduction in the number of variables, if we have a small Index set. This implies that whenever possible, we should use Algorithm 1 in Chapter 4 while constructing Index sets. This chapter is based on [Las06], [Las10], [Löf09], [Par00] and [MOS16].*

6.1 Verification of Candidate Lyapunov function

Once a Candidate Lyapunov function has been found, it is necessary to verify whether it satisfies the conditions (2.16) stated in Lyapunov's Stability Theorem (3). This verification is required (even though we specify (2.16) as constraints while solving the SDP (2.7)) due to the finite machine precision of the solvers used while solving an SDP. The floating point arithmetic of numerical solvers is prone to errors such as rounding off errors. Thus, the solution obtained after solving an SDP may violate the constraints slightly [Löf09]. We have primarily used MOSEK as the solver for solving the SDP's and MOSEK lists the constraint violation in its basic solution summary [MOS16].

We can solve the SDP (2.7) either in its primal form or in dual form by using YALMIP [Löf04]. If we solve the SDP in its primal form, the equality constraint (2.7d) may be violated in the final solution whereas, if we solve the SDP in its dual form (also referred to as the image form in some literature) the semidefinite constraint (2.7c) may be violated. Violation of the equality constraint means that we have not found the exact certificate for proving non-negativity of p but instead we have proven non-negativity of a slightly perturbed polynomial and thus the original polynomial is very close to being an SoS polynomial based on accuracy upto finite machine precision. However, if we are solving the dual problem and the semidefinite constraint is violated than the obtained certificate of non-negativity is practically useless as we can't use it to prove non-negativity of the obtained polynomial (or the Lyapunov function) over the considered semialgebraic set. Further discussion on the feasibility of SoS can be found in [Löf09], [Löf11] and [LP04].

For the verification of the candidate Lyapunov function we have used two approaches. Firstly, we have used the following theorem from [Löf09] for validating whether the solution is strictly feasible.

Theorem 6 *Let $Q \in \mathbb{R}^{s(d)\times s(d)}$ and consider the SDP constraints, (2.7c) and (2.7d). Suppose we are solving the SoS decomposition problem, $p(x) = v_d(x)^T Q v_d(x)$.*

If the constraints are violated and we get $p'(x)$ instead of $p(x)$, then then the obtained polynomial is non-negative (over the considered semialgebraic set), if

$$\Lambda_{min}(Q) \geq \left\| p'(x) - v_d(x)^T Q v_d(x) \right\|_\infty \tag{6.1}$$

where, $\Lambda_{min}(Q)$ is the minimum eigenvalue of the gram matrix Q.

This approach is based on the fact that SDP solvers using Interior point methods, typically find solutions in the analytic center of the feasible region. Thus, if a strict interior exists, we will obtain a strictly feasible solution. A feasibility problem will however not return an optimal solution. To overcome this, we add a new constraint, which forces the solver to return a feasible solution which is very close to the optimal value. Hence, we resolve the problem with the requirement that we should be within $\frac{1}{10000}$ from the previously computed optimal objective value [Löf11].

It can be concluded from Theorem 6, that if we can make the Gram matrix Q sufficiently positive definite, than the polynomial is non-negative on the semialgebraic set. The next approach is also based on this idea.

We constrain the candidate Lyapunov function to not just be positive but to be greater than a quadratic function $\Gamma(x^T x)$, where $x \in \mathbb{R}^n$ is the state of the system and Γ is a tuning parameter. The tuning parameter Γ is set based on Theorem 6.

6.2 Symmetry reduction and Post-processing by YAMIP

YALMIP automatically detects sign symmetries in polynomials i.e. $p(x) = p(-x)$ and based on it, the SDP can be reduced to half of its original size as some of the elements of Q matrix will cancel out each other.

Post-processing is a practical feature in YALMIP by which we can eliminate further candidate monomials after solving the SDP (2.7). Due to the limitations posed by numerical solvers (discussed earlier in section 6.1), we may obtain a Lyapunov function which has coefficients of some of its candidate monomials to be a very low value (say in the order of 10^{-5}), such monomials can also be rejected which reduces the computational complexity of the SDP [Löf09]. This feature was useful in this project (see table 6.1).

6.3 Computational complexity

The computational complexity (also referred to as time complexity in some literature) of an algorithm is defined by the order of growth $\mathcal{O}$ of its running time [Cor+09]. The order of growth $\mathcal{O}$ of running time for an algorithm gives the asymptotic upper bound for the running time of the algorithm. For a function $f(n)$, it can be defined as follows:

$$\mathcal{O}(g(n)) = \{f(n) \mid \exists c, n_0 \in \mathbb{R}_+ \quad \text{such that} \quad 0 \leq f(n) \leq cg(n) \quad \forall n \geq n_0\} \tag{6.2}$$

$\mathcal{O}$-notation gives an upper bound on a function, within a constant factor.

6.3.1 SoS and SDP

As mentioned earlier, the size of the Q matrix while finding an SoS decomposition in (1) determines the computational complexity. The size of the corresponding SDP is determined by the number of constraints which is $s(d)$ and the size of $s(d)$ is bounded by n^d [Las10]. Therefore, as per (6.2), the computational complexity for finding SoS is $\mathcal{O}(n^d)$ where n is the number of variables and d is the maximum degree of the same.

For a general SDP solver such as MOSEK or SeDuMi which works with Interior-point method, we can find an approximate solution (within prescribed accuracy

$\epsilon > 0$), in a time that is polynomial in the input size of the problem [Las10]. Specifically, the *worst-case time complexity* for an SDP is $\mathcal{O}(n^{\frac{1}{2}})$ [VB96] where n is the input size or the number of variables. It is also mentioned in [VB96], that practically, solving a SDP is faster then $\mathcal{O}(n^{\frac{1}{2}})$ and can be of the order of $\mathcal{O}(n^{\frac{1}{4}})$ or $\mathcal{O}(\log n)$.

6.3.2 Putinar's Positivstellensatz

The worst-case computational complexity of solving SDP 1 obtained via Putinar's Positivstellensatz is $\mathcal{O}(n^d)$ where, n is the number of variables and d is the maximum degree of the SoS multipliers. The size of the LMI constraint (3.4c) in 1 is equivalent to solving $m+1$ LMI's growing at the rate of $\mathcal{O}(n^{\frac{d}{2}})$, where m is the number of semialgebraic sets. Finally, the constraint (3.4d) is also equivalent to solving $m+1$ LMI's growing at the rate of $\mathcal{O}(n^{\frac{d}{2}})$ (see [Las06], [Las10] and [Las15]).

Practically, we have observed that the size grows rapidly as we increase the degree of Lyapunov function candidate. Fortunately, in all the considered problems, we have been able to find candidate Lyapunov functions having a degree of 2 or 4.

6.3.3 Sparse Putinar's Positivstellensatz

If we can apply sparse version of Putinar's Positivstellensatz, than computational savings are guaranteed in the sparse variant because the SoS multiplier associated with polynomial inequality g_j consists of monomials associated with g_j only. We can demonstrate this comparison by following example.

Consider the following polynomial,

$$F(x_1, x_2, x_3) = f_1(x_1, x_2) + f_2(x_2, x_3) \tag{6.3}$$

Let the semialgebraic sets for (6.3) be as follows:

$$K_{x_1,x_2} = \{(x_1, x_2) \in \mathbb{R}[x_1, x_2]^{n_1+n_2} : \quad g_j(x_1, x_2) \geq 0, \quad j \in I_{x_1,x_2}\} \tag{6.4a}$$
$$K_{x_2,x_3} = \{(x_2, x_3) \in \mathbb{R}[x_2, x_3]^{n_2+n_3} : \quad g_j(x_2, x_3) \geq 0, \quad j \in I_{x_2,x_3}\} \tag{6.4b}$$
$$\tag{6.4c}$$

By applying Putinar's Positivstellensatz we get the representation,

$$F = P_0 + \sum_j P_j g_j + \sum_k Q_k h_k \tag{6.5}$$

for some SoS polynomials $P_0, P_j, Q_j \, \epsilon \, \Sigma[x_1, x_2, x_3]$.

Whereas, if we apply sparse version of Putinar's Positivstellensatz we get the representation,

$$F = P_0 + \sum_j P_j g_j + Q_0 + \sum_k Q_k h_k \tag{6.6}$$

for some SoS polynomials $P_0, P_j \, \epsilon \, \Sigma[x_1, x_2]$ and $Q_0, Q_j \, \epsilon \, \Sigma[x_2, x_3]$.

Note that, even if we apply Putinar's Positivstellensatz despite the presence of sparsity in the system dynamics, the resulting SoS decomposition will have 0 coefficients for the monomials which could have been rejected before. However, the size of SDP will be large (it may be practically to large to solve) and therefore, it is advisable to consider sparsity while searching for SoS polynomials.

As expected in (6.6), SDP 2 obtained via Sparse Putinar's Positivstellensatz is less computationally complex as compared to SDP 1 obtained via Putinar's Positivstellensatz.

Let p denote the maximum number of Index sets constructed for SDP 2 and let κ denote the maximum number of elements among all the Index sets and can be formally defined as follows.

$$\kappa := max_k |I_k| \quad k = 1, \cdots, p \tag{6.7}$$

Then, the worst-case computational complexity of solving SDP's 2 is $p\mathcal{O}(\kappa^d)$ where, d is the maximum degree of the SoS multipliers. The size of the LMI constraint (4.16c) in 2 is equivalent to solving $m + 2p$ LMI's growing at the rate of $\mathcal{O}(\kappa^{\frac{d}{2}})$, where m is the number of semialgebraic sets. The constraint (4.16d) is also similarly, equivalent to solving $m + 2p$ LMI's growing at the rate of $\mathcal{O}(\kappa^{\frac{d}{2}})$ (see [Las06], [Wak+06], [Las10] and [Las15]). Finally, based on (6.7), we can conclude that, *smaller the size of Index set I_k, greater will be the computational savings achieved by applying Sparse Putinar's Positivstellensatz*. In the next section, we have practically compared the saving's obtained by exploiting sparsity.

6.4 Practical computational comparison between SDP 1 and SDP 2

As mentioned earlier, practically computation might be faster as compared to the theoretical worst-case time complexity bounds discussed in the previous section. We present the computational savings for the complex systems discussed in Chapter 5, viz. the Wind turbine and Ørsted Satellite.

Wind Turbine

To demonstrate computational savings, we have generated candidate Lyapunov functions for previous example of wind turbine (5.5) using both the approaches and the following table 6.1 shows the comparison between the same.

No.	**Description**	**Putinar Positivstellensatz**	**Sparse Putinar Positivstellensatz**
1.	Number of parametric variables	541	303
2.	Number of constraints prior to post-processing	1236	998
3.	Number of constraints after post-processing	1213	975
4.	YALMIP execution time	0.8732 sec	0.6632 sec
5.	Solver execution time	1.2718 sec	0.9418 sec
6.	Size of SoS multipliers	14×1 92244 bytes	24×1 138408 bytes
7.	Total size of constraints	18×1 165884 bytes	28×1 193328 bytes

Table 6.1: Computational Comparison between Putinar's Positivstellensatz and Sparse Putinar's Positivstellensatz for the wind turbine

As can be seen from table 6.1, the number of constraints and parametric variables is significantly reduced if we exploit sparsity while constructing the Lyapunov function using SoS polynomials. However, this comes at the cost of creation of more SoS 'multipliers'. By SoS 'multipliers' over here refer to the SoS polynomials in the representations given by Theorem 4 and Theorem 5. The next fig. (6.1) shows the comparison between the SDP's 1, 2 and the theoretical bounds discussed in the previous section.

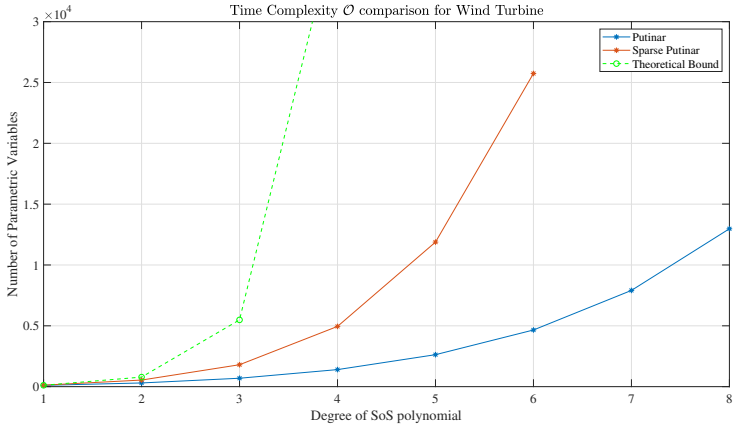


Figure 6.1: Time complexity comparison between SDP 1 and SDP 2, while finding candidate Lyapunov function for the Wind Turbine

Ørsted Satellite

As done previously, we will now demonstrate computational savings for Ørsted Satellite (5.35), (5.37). The following table 6.2 shows the comparison between the same.

No.	Description	Putinar Positivstellensatz	Sparse Putinar Positivstellensatz
1.	Number of parametric variables	540	286
2.	Number of constraints prior to post-processing	1223	969
3.	Number of constraints after post-processing	1223	969
4.	YALMIP execution time	0.7707 sec	0.7573 sec
5.	Solver execution time	0.6783 sec	0.5912 sec
6.	Size of SoS multipliers	14×1 92244 bytes	20×1 116384 bytes
7.	Total size of constraints	17×1 193008 bytes	23×1 194044 bytes

Table 6.2: Computational Comparison between Putinar's Positivstellensatz and Sparse Putinar's Positivstellensatz for the Ørsted Satellite

As can be seen from table 6.2, finding candidate Lyapunov function for the satellite has a similar computational load as compared to finding candidate Lyapunov function for the wind turbine problem. This is because, the number of states in both the systems is 7 and recall that, the computational complexity of (2.7) is dependent on the number of variables and the maximum degree of the SoS polynomials only. Fig. (6.2) shows the comparison between the SDP's 1, 2 and the theoretical bounds discussed in the previous section.

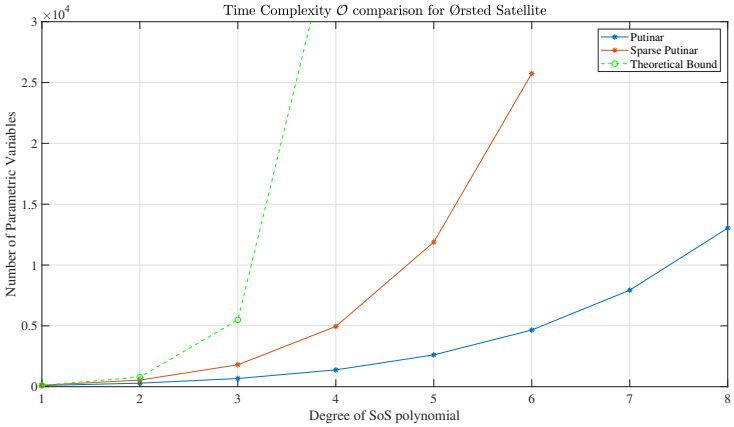


Figure 6.2: Time complexity comparison between SDP 1 and SDP 2, while finding candidate Lyapunov function for the Ørsted Satellite

6.5 Computational Savings obtained by using small Index sets

Consider a 7^{th} order nonlinear model and suppose we want to find a candidate Lyapunov function having degree 2. The semialgebraic set K is constructed as follows:

$$K = \{x \in \mathbb{R}^7 : g_j(x) \geq 0, j = 1, \cdots, 7\} \tag{6.8a}$$

where, $g_{1,\cdots,7}(x)$ is a compact function of $x \in \mathbb{R}^7$. Further consider, the following two sets of Index sets.

$$I_1 = \{1,2,3,5\}, \quad g_8 = 4M^2 - (x_1^2 + x_2^2 + x_3^2 + x_5^2), \quad J_1 = \{1,2,3,5,8\} \tag{6.9a}$$
$$I_2 = \{3,4\} \quad , \quad g_9 = 2M^2 - (x_3^2 + x_4^2) \quad , \quad J_2 = \{4,9\} \tag{6.9b}$$
$$I_3 = \{5,6,7\} \quad , \quad g_{10} = 3M^2 - (x_5^2 + x_6^2 + x_7^2) \quad , \quad J_3 = \{6,7,10\} \tag{6.9c}$$

In (6.9), we have $p = 3$ Index sets I with corresponding Index sets J.

$$I_1 = \{1,3\}, \quad g_8 = 2M^2 - (x_1^2 + x_3^2) \ , \quad J_1 = \{1,3,8\} \tag{6.10a}$$
$$I_2 = \{3,4\}, \quad g_9 = 2M^2 - (x_3^2 + x_4^2) \ , \quad J_2 = \{4,9\} \tag{6.10b}$$
$$I_3 = \{4,2\}, \quad g_{10} = 2M^2 - (x_2^2 + x_4^2), \quad J_3 = \{2,10\} \tag{6.10c}$$
$$I_4 = \{2,5\}, \quad g_{11} = 2M^2 - (x_2^2 + x_5^2), \quad J_4 = \{5,11\} \tag{6.10d}$$
$$I_5 = \{5,6\}, \quad g_{12} = 2M^2 - (x_5^2 + x_6^2), \quad J_5 = \{6,12\} \tag{6.10e}$$
$$I_6 = \{6,7\}, \quad g_{13} = 2M^2 - (x_6^2 + x_7^2), \quad J_6 = \{7,13\} \tag{6.10f}$$

In (6.10), we have $p = 6$ Index sets I with corresponding Index sets J.

On the next page, we have presented a tabular comparison between (6.9) and (6.10) based on the number of parametric variables when we pose the problem as an SDP. It can be concluded from table 6.3 that, having $p = 6$ gives us a saving of 50 variables.

Description	$p = 3$	$p = 6$
SoS Representation	$V = \sum_{k=1}^{3} \left(q_k + \sum_{j \in J_k} q_{jk} g_j \right)$ $= q_1 + q_{11}g_1 + q_{21}g_2 + q_{31}g_3 + q_{51}g_5 + q_{81}g_1 + q_2 + q_{42}g_4 + q_{92}g_9 + q_3 + q_{63}g_6 + q_{73}g_7 + q_{103}g_{10}$	$V = \sum_{k=1}^{6} \left(q_k + \sum_{j \in J_k} q_{jk} g_j \right)$ $= q_1 + q_{11}g_1 + q_{31}g_3 + q_{81}g_8 + q_2 + q_{42}g_4 + q_{92}g_9 + q_3 + q_{23}g_2 + q_{103}g_{10} + q_4 + q_{54}g_5 + q_{114}g_{11} + q_5 + q_{65}g_6 + q_{125}g_{12} + q_6 + q_{76}g_7 + q_{136}g_{13}$
Number of variables per Index sets	$I_1 = \binom{4+2}{2} = 15$ $I_2 = \binom{2+2}{2} = 6$ $I_3 = \binom{3+2}{2} = 10$	$I_1 = \binom{2+2}{2} = 6$ $I_2 = \binom{2+2}{2} = 6$ $I_3 = \binom{2+2}{2} = 6$ $I_4 = \binom{2+2}{2} = 6$ $I_5 = \binom{2+2}{2} = 6$ $I_6 = \binom{2+2}{2} = 6$
Total number of variables due to all Index sets	$n_1 = 5 \times \binom{2+2}{2} = 75$ $n_2 = 3 \times \binom{2+2}{2} = 18$ $I_3 = 4 \times \binom{3+2}{2} = 40$	$n_1 = 3 \times \binom{2+2}{2} = 18$ $n_2 = 3 \times \binom{2+2}{2} = 18$ $n_3 = 3 \times \binom{2+2}{2} = 18$ $n_4 = 3 \times \binom{2+2}{2} = 18$ $n_5 = 3 \times \binom{2+2}{2} = 18$ $n_6 = 3 \times \binom{2+2}{2} = 18$
Total number of variables due to constraint (4.16c)	133	108
Total number of variables due to constraint (4.16d)	133	108
Variables declared due to candidate Lyapunov function	$\binom{7+2}{2} = 36$	$\binom{7+2}{2} = 36$
Total number of variables	302	252

Table 6.3: Breakdown of parametric variables and comparison based on number of Index sets constructed.

Chapter 7

Controller Synthesis

Summary *This chapter focuses on the design of a control law based on the candidate Lyapunov functions obtained in the previous chapters. We begin by reviewing Lyapunov redesign technique in which an existing controller is made robust to external disturbances, while requiring minimal information of the disturbance. Further, we review Control Lyapunov functions (CLF) and Sontag's formula for nonlinear stabilization. We also present some intermediate results obtained while trying Lyapunov Redesign for Ørsted Satellite. This chapter also serves as the motivation for future work. This chapter is based on [Kha15], [Kha02], [Son89], [Tan06] and [Vid02].*

7.1 Lyapunov Redesign

Consider the following nonlinear system

$$\dot{x} = f(x) + G(x)[u + \delta(t, x, u)] \tag{7.1}$$

where, $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ is the control input. The functions f, G, and δ are defined for $(t, x, u) \in [0, \infty) \times \mathcal{D} \times \mathbb{R}^m$, where $\mathcal{D} \subset \mathbb{R}^n$ is a domain that contains the origin. We assume that f, G are locally Lipschitz and δ is piecewise continuous in t and locally Lipschitz in x and u. The functions f and G are known, while δ is unknown. A nominal model of the system is

$$\dot{x} = f(x) + G(x)u \tag{7.2}$$

Suppose, we have a state feedback controller $u = \Phi(x)$ which stabilizes the origin of the nominal model (7.2) and suppose we have found a candidate Lyapunov function for the autonomous system (7.1) such that it is continuously differentiable and it satisfies (2.16a), (2.16b) and (2.16c) and can be stated as follows.

$$\frac{\partial V}{\partial x}[f(x) + G(x)\phi(x)] \leq -W(x) \tag{7.3}$$

where, $W(x)$ is some positive definite function. Recall that previously in SDP's 1 and 2, we have constrained the Lie derivative of the candidate Lyapunov function to be less than $-\Gamma \|x\|_2$. We now state an important assumption regarding the unknown disturbance δ.

Assumption 7.1 Assume that, with the modified control law $u = \Phi(x) + v$ where, v will be designed later based on the Lie derivative of the candidate Lyapunov function (hence, the name *Lyapunov Redesign*), then the uncertain term δ satisfies the inequality,

$$\|\delta(t, x, \Phi(x) + v)\| \leq \rho(x) + \kappa_0 \|v\|, \quad 0 \leq \kappa_0 < 1 \tag{7.4}$$

where, $\rho(x) \geq 0$ is an measure of the size of uncertainty and is locally Lipschitz. The bound (7.4) with the known ρ and κ_0 is the only information we need to know about $\delta(t, x, \Phi(x) + v)$. For the sake of simplicity, we will be relaxing the notation for $\delta(t, x, \Phi(x) + v)$ and stating it as simply, δ.

Under the control $u = \Phi(x) + v$, the closed-loop system, can be stated as follows.

$$\dot{x} = f(x) + G(x)\Phi(x) + G(x)[v + \delta] \tag{7.5}$$

The Lie derivative of candidate Lyapunov function $V(x)$ for (7.5) is given as follows.

$$\dot{V} \leq -W(x) + w^T v + w^T \delta \tag{7.6}$$

where, $w^T = \frac{\partial V}{\partial x} G$. The first term on the right-hand side is due to the nominal closed-loop system. The second and third terms represent, respectively, the effect of the control v and the uncertain term δ on $\dot{V}$. Due to the matching condition, the uncertain term δ appears at the same point where v appears. Consequently, it is possible to choose v to cancel the (possibly destabilizing) effect of δ on $\dot{V}$. We have

$$w^T v + w^T \delta \leq w^T v + \|w\| \|\delta\| \leq w^T v + \|w\| \|\rho(x) + \kappa_0 \|v\|\| \tag{7.7}$$

We design v as

$$v = -\beta(x) \cdot \frac{w}{\|w\|} \tag{7.8}$$

such that, with a nonnegative locally Lipschitz function β, we obtain

$$w^T v + w^T \delta \leq -\beta \|w\| + \rho \|w\| + \kappa_0 \beta \|w\| = -\beta(1 - \kappa_0) \|w\| + \rho \|w\| \tag{7.9}$$

Choosing $\beta(x) \geq \frac{\rho(x)}{1-\kappa_0} \quad \forall x \in \mathcal{D}$, yields

$$w^T v + w^T \delta \leq -\rho \|w\| + \rho \|w\| = 0 \tag{7.10}$$

Hence, with the control (7.8), the derivative of $V(x)$ along the trajectories of the closed-loop system (7.5) is negative definite. The controller (7.8) is a discontinuous function of the state x. The discontinuous function raises some practical (such as inability of the controller to oscillate with very high frequency due to actuator limitations) and theoretical issues (such as existence and uniqueness of solutions and validity of Lyapunov analysis) which have been discussed in [Kha15] We thereby, implement a continuous approximation of (7.8), given by

$$v = -\beta(x) Sat\left(\frac{\beta(x)w}{\mu}\right) = \begin{cases} -\beta(x)\frac{w}{\|w\|}, & \text{if } \beta(x)\|w\| > \mu \\ -\beta^2(x)\frac{w}{\mu}, & \text{if } \beta(x)\|w\| \leq \mu \end{cases} \tag{7.11}$$

where, μ is a real positive constant number which decides the continuity of the control law at the point of switching. Smaller the value of μ, more discontinuous will be the controller.

We have applied Lyapunov Redesign technique on the Ørsted Satellite (discussed previously in Chapter 5). Recall, that the Ørsted Satellite is stabilized via a PD control law. We have attempted to make the existing controller robust to disturbances by applying Lyapunov Redesign, using the candidate Lyapunov function obtained in (5.45). We present the simulation results as follows.

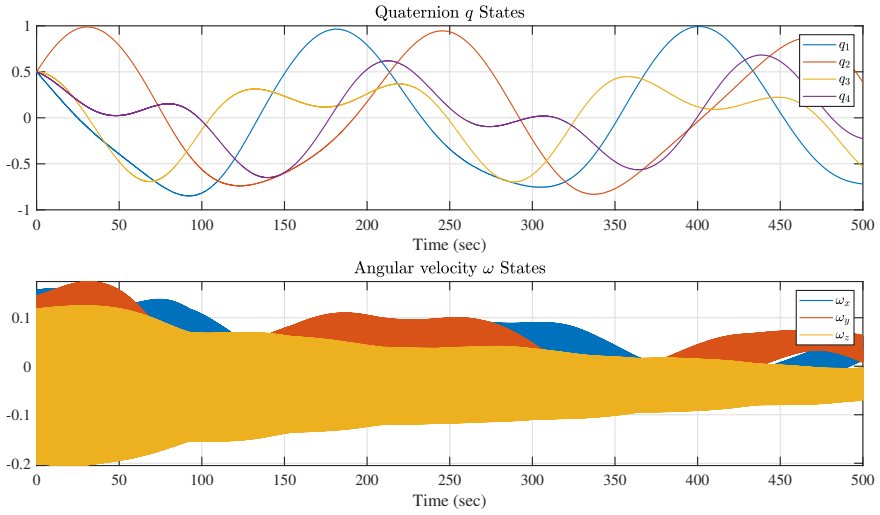


Figure 7.1: Simulation results for Lyapunov Redesign on Ørsted Satellite

It can be concluded from fig. (7.1) that since there is no integral action, the Quaternion state q is oscillating. The angular velocity is bounded due to the strong

derivative action. The high amount of chattering is expected due to the switching nature of the control law.

7.2 Sontag's formula for nonlinear stabilization

Recall that, the classical Lyapunov's stability Theorem (3) is defined for proving the stability of an equilibrium point for an autonomous system (2.13). Therefore, we need a generalization of Theorem 3 such that we can analyze the stability of a system with inputs. Artstein [Art83] and Sontag [Son89] came up with the concept of Control Lyapunov function (CLF) which can be stated as follows.

Theorem 7 *Consider the stabilization problem for the system*

$$\dot{x} = f(x) + g(x)u \tag{7.12}$$

where, $x \in \mathbb{R}^n$, $u \in \mathbb{R}$, $f(x)$ and $g(x)$ are locally Lipschitz and $f(0) = 0$. Suppose there is a locally Lipschitz stabilizing state feedback control $u = \chi(x)$ such that the origin of

$$\dot{x} = f(x) + g(x)\chi(x) \tag{7.13}$$

is asymptotically stable. Then, there is a smooth Lyapunov function $V(x)$ such that :

$$\frac{\partial V}{\partial x}[f(x) + g(x)\chi(x)] < 0, \quad \forall x \neq 0 \tag{7.14a}$$

$$\frac{\partial V}{\partial x}g(x) = 0 \text{ and } x \neq 0 \quad \Longrightarrow \quad \frac{\partial V}{\partial x}f(x) < 0 \tag{7.14b}$$

If the origin of (7.13) is globally asymptotically stable, then $V(x)$ is radially unbounded and the inequality (7.14b) holds globally.

Thus, the existence of a function V satisfying (7.14b) is a necessary condition for the existence of a stabilizing state feedback control. Existence of a CLF is also a sufficient condition for the existence of the following stabilizing state feedback control.

$$\Phi(x) = \begin{cases} -[\frac{\partial V}{\partial x}f + \sqrt{\frac{\partial V}{\partial x}f^2 + \frac{\partial V}{\partial x}g^4}], & \text{if } \frac{\partial V}{\partial x}g \neq 0 \\ 0, & \text{if } \frac{\partial V}{\partial x}g = 0 \end{cases} \tag{7.15}$$

(7.15) is referred to as Sontag's formula and the controller's design ensures that the Lie derivative is negative definite.

For, $x \neq 0$, if $\frac{\partial V}{\partial x}g = 0$, then

$$\dot{V} = \frac{\partial V}{\partial x} f < 0 \tag{7.16}$$

and if $\frac{\partial V}{\partial x} g \neq 0$, then

$$\dot{V} = \frac{\partial V}{\partial x} f - [\frac{\partial V}{\partial x} f + \sqrt{\frac{\partial V}{\partial x} f^2 + \frac{\partial V}{\partial x} g^4}] \tag{7.17a}$$

$$= -\sqrt{\frac{\partial V}{\partial x} f^2 + \frac{\partial V}{\partial x} g^4} < 0 \tag{7.17b}$$

We have attempted to construct an SoS program based on Sontag's formula such that, we can directly obtain the CLF as well as the control law as follows.

Semidefinite Program 3

$$\textit{Find} \quad V(x) \textit{ and } \Phi(x) \tag{7.18a}$$

$$\textit{such that} \quad V(0) = 0, \tag{7.18b}$$

$$V(x) - \sum_{k=1}^{p} (F_k + \sum_{j \in J_k}^{p} F_{jk} g_j) - \Gamma \left\| x \right\|_2 \in \Sigma[x] \ , \ j = 1, \cdots, m + p \tag{7.18c}$$

$$- \sqrt{\frac{\partial V}{\partial x} f^2 + \frac{\partial V}{\partial x} g^4} - \sum_{k=1}^{p} (F_k + \sum_{j \in J_k}^{p} F_{jk} g_j) - \Gamma \left\| x \right\|_2 \in \Sigma[x] \ , \ j = 1, \cdots, m + p \tag{7.18d}$$

However, the SDP 3 is non-convex in nature and solving it is not possible with available SDP solvers.

Chapter 8

Conclusion

The primary scope of this project was to find candidate Lyapunov functions and exploit sparsity in system dynamics such that this approach becomes applicable to complex systems. We can conclude that by applying Putinar's Positivstellensatz and SoS, we can find candidate Lyapunov functions using SDP 1. Fortunately, it is unlikely for a practical dynamical system to have more than 15 states and thus, the approach is scalable to bigger problems as well using SDP 2. However, it is important to note that the results of these SDP's are dependent on the accuracy of solvers. Compared to LP, SDP is still not a mature technology and while, we can verify the results using Theorem 6, we should still be careful while designing controllers based on the same. Thus, the topic considered in the project requires further investigation and we have compiled a list of possible future works.

8.1 Future Works

- Verify the candidate Lyapunov function for the Wind turbine model by simulating extreme wind conditions
- Find candidate Lyapunov function for different regions of a wind turbine operation and verify them by designing controllers for the same
- Formulate a convex SDP for Sontag's formula
- Apply Algorithm 1 for designing a control law for systems composed of cascaded subsystems using backstepping
- Improve the Lyapunov Redesign control law for Ørsted Satellite.
- Find optimal adaptive gain for adaptive controllers using SoS and Putinar's Positivstellensatz

Bibliography

[AM19] Amir Ali Ahmadi and Anirudha Majumdar. "DSOS and SDSOS optimization: more tractable alternatives to sum of squares and semidefinite optimization". In: *SIAM Journal on Applied Algebra and Geometry* 3.2 (2019), pp. 193–230.

[Ant13] Gianluca Antonelli. "Interconnected dynamic systems: An overview on distributed control". In: *IEEE Control Systems Magazine* 33.1 (2013), pp. 76–88.

[Art83] Zvi Artstein. "Stabilization with relaxed controls". In: *Nonlinear Analysis: Theory, Methods & Applications* 7.11 (1983), pp. 1163–1173.

[Ber99] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[Boy+94] Stephen Boyd et al. *Linear matrix inequalities in system and control theory*. Vol. 15. Siam, 1994.

[BP93] Jean RS Blair and Barry Peyton. "An introduction to chordal graphs and clique trees". In: *Graph theory and sparse matrix computation*. Springer, 1993, pp. 1–29.

[BPT12] Grigoriy Blekherman, Pablo A Parrilo, and Rekha R Thomas. *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[BW97] Thomas Bak and Rafal Wisniewski. "Passive Aerodynamic Stabilisation of Low Earth Orbit Satellite". In: *Spacecraft Guidance, Navigation and Control Systems*. Vol. 381. 1997, p. 469.

[Cor+09] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2009.

[ES09] Thomas Esbensen and Christoffer Sloth. "Fault diagnosis and fault-tolerant control of wind turbines". In: *Aalborg University* (2009).

[JW03] Zachary William Jarvis-Wloszek. "Lyapunov based analysis and controller synthesis for polynomial systems using sum-of-squares optimization". PhD thesis. University of California, Berkeley Berkeley, CA, 2003.

[Kha02] Hassan K Khalil. *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002.

[Kha15] Hassan K Khalil. *Nonlinear control*. Pearson New York, 2015.

[KM09] Masakazu Kojima and Masakazu Muramatsu. "A note on sparse SOS and SDP relaxations for polynomial optimization problems over symmetric cones". In: *Computational Optimization and Applications* 42.1 (2009), pp. 31–41.

[Las06] Jean B Lasserre. "Convergent SDP-relaxations in polynomial optimization with sparsity". In: *SIAM Journal on Optimization* 17.3 (2006), pp. 822–843.

[Las10] Jean-Bernard Lasserre. *Moments, positive polynomials and their applications*. Vol. 1. World Scientific, 2010.

[Las15] Jean Bernard Lasserre. *An introduction to polynomial and semi-algebraic optimization*. Vol. 52. Cambridge University Press, 2015.

[Lau09] Monique Laurent. "Sums of squares, moment matrices and optimization over polynomials". In: *Emerging applications of algebraic geometry*. Springer, 2009, pp. 157–270.

[LP04] Johan Lofberg and Pablo A Parrilo. "From coefficients to samples: a new approach to SOS optimization". In: *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*. Vol. 3. IEEE. 2004, pp. 3154–3159.

[Löf04] Johan Löfberg. "YALMIP: A toolbox for modeling and optimization in MATLAB". In: *Proceedings of the CACSD Conference*. Vol. 3. Taipei, Taiwan. 2004.

[Löf11] Johan Löfberg. *Strictly feasible sum-of-squares solutions*. Last visited on 2019-17-03, `https://yalmip.github.io/Strictly-feasible-sum-of-squares/`. 2011.

[Löf09] Johan Löfberg. "Pre- and post-processing sum-of-squares programs in practice". In: *IEEE Transactions on Automatic Control* 54.5 (2009), pp. 1007–1011.

[MK87] Katta G Murty and Santosh N Kabadi. "Some NP-complete problems in quadratic and nonlinear programming". In: *Mathematical programming* 39.2 (1987), pp. 117–129.

[MOS16] ApS MOSEK. *Modeling cookbook*. 2016.

[Par00] Pablo A Parrilo. "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization". PhD thesis. California Institute of Technology, 2000.

[PP05] Antonis Papachristodoulou and Stephen Prajna. "A tutorial on sum of squares techniques for systems analysis". In: *Proceedings of the 2005, American Control Conference, 2005.* IEEE. 2005, pp. 2686–2700.

[PS12] Andreas Søndergaard Pedersen and Christian Sigge Steiniche. "Safe operation and emergency shutdown of wind turbines". In: *Aalborg University* (2012).

[Put93] Mihai Putinar. "Positive polynomials on compact semi-algebraic sets". In: *Indiana University Mathematics Journal* 42.3 (1993), pp. 969–984.

[SL+91] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. Vol. 199. 1. Prentice hall Englewood Cliffs, NJ, 1991.

[Slo13] Christoffer Sloth. *Lecture notes on Consensus in Dynamic Networks*. Vol. 1. Aalborg University, 2013.

[Slo16] Christoffer Sloth. "On the computation of Lyapunov functions for interconnected systems". In: *2016 IEEE Conference on Computer Aided Control System Design (CACSD)*. IEEE. 2016, pp. 850–855.

[Son89] Eduardo D Sontag. "A 'universal'construction of Artstein's theorem on nonlinear stabilization". In: *Systems & control letters* 13.2 (1989), pp. 117–123.

[SPW12] Christoffer Sloth, George J Pappas, and Rafael Wisniewski. "Compositional safety analysis using barrier certificates". In: *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*. Citeseer. 2012, pp. 15–24.

[Stu99] Jos F Sturm. "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones". In: *Optimization methods and software* 11.1-4 (1999), pp. 625–653.

[TAM10] Daniel Torczynski, Rouzbeh Amini, and Paolo Massioni. "Magnetorquer based attitude control for a nanosatellite testplatform". In: *AIAA Infotech@ Aerospace 2010*. 2010, p. 3511.

[Tan06] Weehong Tan. "Nonlinear Control Analysis and Synthesis using Sum-of-Square Programming". In: (2006).

[VB96] Lieven Vandenberghe and Stephen Boyd. "Semidefinite programming". In: *SIAM review* 38.1 (1996), pp. 49–95.

[Vid02] Mathukumalli Vidyasagar. *Nonlinear systems analysis*. Vol. 42. Siam, 2002.

[Wak+06] Hayato Waki et al. “Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity”. In: *SIAM Journal on Optimization* 17.1 (2006), pp. 218–242.

[Wak+08] Hayato Waki et al. “Algorithm 883: SparsePOP—A Sparse Semidefinite Programming Relaxation of Polynomial Optimization Problems”. In: *ACM Transactions on Mathematical Software (TOMS)* 35.2 (2008), p. 15.

[Wer78] James R Wertz. *Spacecraft Attitude Determination and Control*. Vol. 73. Springer Science & Business Media, 1978.

[Wis96] Rafal Wisniewski. “Satellite attitude control using only electromagnetic actuation”. PhD thesis. Aalborg University. Department of Control Engineering, 1996.

[Wis99] Rafal Wisniewski. *Lecture notes on Modeling of a Spacecraft*. Dansk. Research report. Department of Control Engineering, 1999. ISBN: xxxxxxxxxx.

[ÅW13] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

Appendix A

Appendix A : Feasibility of constraints

A.1 Putinar Positivstellensatz representation

In this section of the Appendix, we have stated the constraint violation while obtaining the Lyapunov function for the wind turbine model using Putinar's Positivstellensatz representation. The first 14 constraints represent the SoS constraint on the SoS 'multipliers'. Constraint 15 represents the SoS constraint (2.16b) on the Lyapunov function, Constraint 16 represents the SoS constraint (2.16c) on on the Lie derivative of the Lyapunov function and finally, Constraint 17 represents the SoS constraint (2.16a) on Lyapunov function.

ID	Constraint	Primal residual
1	SOS constraint (polynomial)	1.9303×10^{-13}
2	SOS constraint (polynomial)	6.0063×10^{-13}
3	SOS constraint (polynomial)	7.3896×10^{-13}
4	SOS constraint (polynomial)	5.914×10^{-13}
5	SOS constraint (polynomial)	6.8212×10^{-13}
6	SOS constraint (polynomial)	3.7836×10^{-13}
7	SOS constraint (polynomial)	1.9966×10^{-14}
8	SOS constraint (polynomial)	2.8422×10^{-13}
9	SOS constraint (polynomial)	8.5265×10^{-13}
10	SOS constraint (polynomial)	5.6843×10^{-13}
11	SOS constraint (polynomial)	3.979×10^{-13}
12	SOS constraint (polynomial)	2.9382×10^{-13}
13	SOS constraint (polynomial)	4.2633×10^{-13}
14	SOS constraint (polynomial)	3.6948×10^{-12}
15	SOS constraint (polynomial)	4.0745×10^{-10}

ID	Constraint	Primal residual
16	SOS constraint (polynomial)	8.3144×10^{-5}
17	Equality constraint	-1.4822×10^{-21}

Table A.1: Constraint violations for Wind turbine model while using Putinar's Positivstellensatz representation

Minimum eigenvalue $\Lambda_{min}(Q)$ of the obtained SoS Lyapunov function was 0.0002351. Thus, as per Theorem 6, the obtained SoS certificate is nonegative over the considered semialgebraic set.

A.2 Sparse Putinar Positivstellensatz representation

In this section of the Appendix, we have stated constraint violations while obtaining the Lyapunov function for the wind turbine model using Sparse version of Putinar's Positivstellensatz representation. The first 36 constraints represent the SoS constraint on the SoS 'multipliers'. Constraint 37 represents the SoS constraint (2.16b) on the Lyapunov function, Constraint 38 represents the SoS constraint (2.16c) on the Lie derivative of the Lyapunov function and finally, Constraint 39 represents the SoS constraint (2.16a) on Lyapunov function.

ID	Constraint	Primal residual
1	SOS constraint (polynomial)	2.7285×10^{-12}
2	SOS constraint (polynomial)	9.0949×10^{-13}
3	SOS constraint (polynomial)	8.8818×10^{-16}
4	SOS constraint (polynomial)	1.4779×10^{-12}
5	SOS constraint (polynomial)	9.4363×10^{-13}
6	SOS constraint (polynomial)	5.5511×10^{-17}
7	SOS constraint (polynomial)	1.1369×10^{-12}
8	SOS constraint (polynomial)	1.5916×10^{-12}
9	SOS constraint (polynomial)	2.6645×10^{-15}
10	SOS constraint (polynomial)	2.8955×10^{-13}
11	SOS constraint (polynomial)	1.6769×10^{-12}
12	SOS constraint (polynomial)	4.4409×10^{-15}
13	SOS constraint (polynomial)	1.1369×10^{-13}
14	SOS constraint (polynomial)	9.3969×10^{-13}
15	SOS constraint (polynomial)	3.5527×10^{-15}
16	SOS constraint (polynomial)	9.0949×10^{-13}
17	SOS constraint (polynomial)	2.8422×10^{-14}

ID	Constraint	Primal residual
18	SOS constraint (polynomial)	2.2204×10^{-15}
19	SOS constraint (polynomial)	1.0914×10^{-11}
20	SOS constraint (polynomial)	1.819×10^{-12}
21	SOS constraint (polynomial)	8.8818×10^{-15}
22	SOS constraint (polynomial)	3.0695×10^{-12}
23	SOS constraint (polynomial)	1.5916×10^{-12}
24	SOS constraint (polynomial)	1.3878×10^{-16}
25	SOS constraint (polynomial)	4.5475×10^{-13}
26	SOS constraint (polynomial)	1.819×10^{-12}
27	SOS constraint (polynomial)	5.6843×10^{-14}
28	SOS constraint (polynomial)	1.4655×10^{-13}
29	SOS constraint (polynomial)	1.3642×10^{-12}
30	SOS constraint (polynomial)	4.8352×10^{-12}
31	SOS constraint (polynomial)	4.5475×10^{-13}
32	SOS constraint (polynomial)	1.819×10^{-12}
33	SOS constraint (polynomial)	3.5527×10^{-14}
34	SOS constraint (polynomial)	4.5475×10^{-13}
35	SOS constraint (polynomial)	3.4106×10^{-13}
36	SOS constraint (polynomial)	2.6645×10^{-15}
37	SOS constraint (polynomial)	9.3132×10^{-10}
38	SOS constraint (polynomial)	2.1854×10^{-7}
39	Equality constraint	-5.3482×10^{-23}

Table A.2: Constraint violations for Wind turbine model while using Sparse Putinar's Positivstellensatz representation

Minimum eigenvalue $\Lambda_{min}(Q)$ of the obtained SoS Lyapunov function was 0.032592. Thus, as per Theorem 6, the obtained SoS certificate is nonegative over the considered semialgebraic set.