PARALLEL OPTIMIZATION OF POLYNOMIALS FOR LARGE-SCALE PROBLEMS IN STABILITY AND CONTROL

by

Reza Kamyar

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

Approved January 2016 by the
Graduate Supervisory Committee:

Matthew Peet, Chair
Daniel Rivera
Georgios Fainekos
Panagiotis Artemiadis
Spring Berman

ARIZONA STATE UNIVERSITY

January 2016

ABSTRACT

In today's world, optimal operation of ever-growing industries and markets often requires solving optimization problems with unprecedented sizes. Economic dispatch of generating units in power companies, frequency assignment in large mobile communication networks, profit maximization in competitive markets, and optimal operation of smart grids are few examples of many real-world problems which can be closely modeled as optimization over a large number (tens of thousands) of integer- and real-valued decision variables. Unfortunately, majority of the existing commercial off-the-shelf software are not designed to scale to optimization problems of this size. Moreover, in theory, these optimization problems often fall into the class NP-hard - meaning that despite the tremendous effort towards modernization of optimization algorithms, it is widely suspected that no algorithm can find exact solutions to these problems in a reasonable amount of time.

In this thesis, we focus on some of the NP-hard problems in control theory. Thanks to the converse Lyapunov theory, these problems can often be modeled as optimization over polynomials. To avoid the problem of intractability, we establish a trade off between accuracy and complexity. In particular, we develop a sequence of tractable optimization problems - in the form of Linear Programs (LPs) and/or Semi-Definite Programs (SDPs) - whose solutions converge to the exact solution of the NP-hard problem. However, the computational and memory complexity of these LPs and SDPs grow exponentially with the progress of the sequence - meaning that improving the accuracy of the solutions requires solving SDPs with tens of thousands of decision variables and constraints. Setting up and solving such problems is a significant challenge. Unfortunately, the existing optimization algorithms and software are only designed to use desktop computers or small cluster computers - machines which do not have sufficient memory for solving such large SDPs. Moreover, the speed-up of

these algorithms does not scale beyond dozens of processors. This in fact is the reason we seek parallel algorithms for setting-up and solving large SDPs on large cluster- and/or super-computers.

We propose parallel algorithms for stability analysis of two classes of systems: 1) Linear systems with a large number of uncertain parameters; 2) Nonlinear systems defined by polynomial vector fields. First, we develop a distributed parallel algorithm which applies Polya's and/or Handelman's theorems to some variants of parameter-dependent Lyapunov inequalities with parameters defined over the standard simplex. The result is a sequence of SDPs which possess a block-diagonal structure. We then develop a parallel SDP solver which exploits this structure in order to map the computation, memory and communication to a distributed parallel environment. We produce a Message Passing Interface (MPI) implementation of our parallel algorithms and provide a comprehensive theoretical and experimental analysis on its complexity and scalability. Numerical tests on a supercomputer demonstrate the ability of the algorithm to efficiently utilize hundreds and potentially thousands of processors and analyze systems with 100+ dimensional state-space. We then apply our algorithms to two real-world problems: Stability of plasma in a Tokamak reactor, and optimal electricity pricing in a smart grid environment. Finally, we extend our algorithms to analyze robust stability over more complicated geometries such as hypercubes and arbitrary convex polytopes. Our algorithms can be readily extended to address a wide variety of problems in control; e.g., $\mathcal{H}_2/\mathcal{H}_\infty$ control synthesis for systems with parametric uncertainty, computing control Lyapunov functions for optimal control problems, and analysis and control of switched/hybrid systems.

TABLE OF CONTENTS

Page

LIST OF TABLES ix

LIST OF FIGURES xi

LIST OF SYMBOLS 1

CHAPTER

1 INTRODUCTION 1

1.1 Sum of squares method 2

1.2 Moments method 3

1.3 Quantifier elimination 5

1.4 Reformulation linear techniques 6

1.5 Groebner basis technique 6

1.6 Blossoming technique 7

1.7 Bernstein's, Polya's and Handelman's theorems 8

1.8 Motivations and summary of contributions 9

2 FUNDAMENTAL RESULTS for OPTIMIZATION of POLYNOMIALS . 14

2.1 Background on positivity results 14

2.2 Polynomial optimization and optimization of polynomials 20

2.3 Algorithms for optimization of polynomials 23

2.3.1 Case 1: Optimization over the standard simplex Δ^n 23

2.3.2 Case 2: Optimization over the hypercube Φ^n 24

2.3.3 Case 3: Optimization over the convex polytope Γ^K 27

2.3.4 Case 4: Optimization over compact semi-algebraic sets 29

2.3.5 Case 5: Tests for non-negativity on $\mathbb{R}^n$: 32

3 SEMI-DEFINITE PROGRAMMING AND INTERIOR-POINT ALGORITHMS 35

3.1 Convex optimization and duality 35

3.2 Descent algorithms for convex optimization 37

3.3 Interior-point algorithms for convex problems with inequality constraints 40

3.4 Semi-definite programming 44

3.5 A primal-dual interior-point algorithm for semi-definite programming 48

4 PARALLEL ALGORITHMS FOR ROBUST STABILITY ANALYSIS OVER SIMPLEX 53

4.1 Background and motivations 53

4.1.1 Our contributions 59

4.2 Notation and Preliminaries on homogeneous polynomials 60

4.3 Setting up the problem of robust stability analysis over a simplex .. 62

4.3.1 General formulae for calculating coefficients β and H 65

4.3.2 Number of coefficients $\beta_{\langle h \rangle, \langle \gamma \rangle}$ and $H_{\langle h \rangle, \langle \gamma \rangle}$ 66

4.3.3 The elements of the SDP problem associated with Polya's theorem 69

4.3.4 A parallel algorithm for setting-up the SDP 71

4.4 Complexity analysis of the set-up algorithm 71

4.4.1 Computational complexity analysis 74

4.4.2 Communication complexity analysis 75

4.5 A parallel SDP solver 77

4.5.1 Structure of the SDP variables 80

4.5.2 A parallel implementation for the SDP solver 83

4.6 Computational complexity analysis of the SDP algorithm 83

4.6.1 Complexity analysis for systems with large number of states 87
4.6.2 Complexity of increasing accuracy/decreasing Conservativeness 88
4.6.3 Analysis of scalability/speed-up 89
4.6.4 Synchronization and load balancing analysis 91
4.6.5 Communication graph of the algorithm 93
4.7 Testing and validation 93
4.7.1 Example 1: Application to control of a discretized PDE model in fusion research 94
4.7.2 Example 2: Accuracy and convergence 98
4.7.3 Example 3: Evaluating speed-up 100
4.7.4 Example 4: Maximum state-space and parameter dimensions for a 9-node Linux-based cluster computer 103
5 PARALLEL ALGORITHMS FOR ROBUST STABILITY ANALYSIS OVER HYPERCUBES 105
5.1 Background and motivation 105
5.1.1 Our Contributions 106
5.2 Notation and preliminaries on multi-homogeneous polynomials 106
5.3 Setting up the problem of robust stability analysis over multi-simplex 111
5.3.1 General formulae for calculating coefficients β and H 113
5.3.2 The SDP elements associated with the multi-simplex version of Polya's theorem 114
5.3.3 A parallel algorithm for setting up the SDP 116
5.4 Computational complexity analysis of the set-up algorithm 116

5.4.1 Computational cost of the set-up algorithm: 116
5.4.2 Communication cost of the set-up algorithm: 118
5.4.3 Speed-up and memory requirement of the set-up algorithm: . 119
5.5 Testing and validation 121
5.5.1 Example 1: Evaluating speed-up 121
5.5.2 Example 2: Verifying robust stability over a hypercube 121
5.5.3 Example 2: Evaluating accuracy 123
6 PARALLEL ALGORITHMS FOR NONLINEAR STABILITY ANALYSIS125
6.1 Background and Motivation 125
6.1.1 Our contributions 127
6.2 Definitions and Notation 128
6.3 Statement of the stability problem 132
6.4 Expressing the stability problem as a linear program 133
6.5 Computational complexity analysis 139
6.5.1 Complexity of the LP associated with Handelman's representation 141
6.5.2 Complexity of the SDP associated with Polya's algorithm . . . 142
6.5.3 Complexity of the SDP associated with SOS algorithm 143
6.5.4 Comparison of the Complexities 145
6.6 Numerical results 145
7 OPTIMIZATION OF SMART GRID OPERATION: OPTIMAL UTILITY PRICING AND DEMAND RESPONSE 151
7.1 Background and motivation 151
7.1.1 Our contributions 155

7.2 Problem statement: user-level and utility level problems 157
7.2.1 A model for the building thermodynamics 157
7.2.2 Calibrating the thermodynamics model 159
7.2.3 User-level problem I: optimal thermostat programming 160
7.2.4 User-level problem II: 4-setpoint thermostat program 162
7.2.5 Utility-level optimization problem 163
7.3 Solving user- and utility-level problems by dynamic programming . . 166
7.4 Policy implications and analysis 172
7.4.1 Effect of electricity prices on peak demand and production costs 175
7.4.2 Optimal thermostat programming with optimal electricity prices 179
7.4.3 Optimal thermostat programming for solar customers - impact of distributed solar generation on non-solar customers . . 181
8 SUMMARY, CONCLUSIONS AND FUTURE DIRECTIONS OF OUR RESEARCH 183
8.1 Summary and conclusions 183
8.2 Future directions of our research 187
8.2.1 A Parallel algorithm for nonlinear stability analysis using Polya's theorem 187
8.2.2 Parallel computation for parameter-varying $\mathcal{H}_\infty$-optimal control synthesis 191
8.2.3 Parallel computation of value functions for approximate dynamic programming 194

CHAPTER Page

REFERENCES 198

LIST OF TABLES

Table Page

4.1 Per processor, per iteration computational complexity of the set-up algorithm. L_0 is the number of monomials is $P(\alpha)$; L is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_1} P(\alpha)$; M is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_2} P(\alpha)A(\alpha)$. 75

4.2 Per processor, per iteration communication complexity of the set-up algorithm. L_0 is the number of monomials is $P(\alpha)$; L is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_1} P(\alpha)$; M is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_2} P(\alpha)A(\alpha)$. 77

4.3 Data for Example 1: Nominal values of the plasma resistivity 95

4.4 Upper bounds found for L_{opt} by the SOS algorithm using different degrees for x and α (inf: infeasible, O.M.: Out of Memory) 100

5.1 The lower-bounds on r^* computed by Algorithm 7 using different degree vector D_p and using methods in Bliman (2004a) and Chesi (2005). 124

7.1 Building's parameters as determined in Section 7.2.1 175

7.2 On-peak, off-peak & demand prices of Arizona utility APS 176

7.3 CASE I: Electricity bills (for three days) and demand peaks for different strategies. Electricity prices are from APS. 177

7.4 CASE I: Costs of production (for three days) and Demand peaks for various prices and strategies. Prices are non-regulated and SRP's coefficients of utility cost are: τ =0.00401 \$/(MWh)^2, ν =4.54351 \$/(MWh) 177

7.5 CASE II: Production costs (for three days) and demand peaks associated with regulated optimal electricity prices (calculated by Algorithm 10) and SRP's electricity prices. SRP's marginal costs: $a = 0.0814\frac{\$}{kWh}, b = 59.76\frac{\$}{kW}$ 180

7.6 CASE III: Optimal electricity prices, bills (for three days) and demand peaks for various customers. Marginal costs from SRP: $a = 0.0814 \frac{\$}{kWh}, b = 59.76 \frac{\$}{kW}$ 182

LIST OF FIGURES

Figure Page

4.1 Various interconnections of nodes in a cluster computer (Top), Typical memory hierarchies of a GPU and a multi-core CPU (bottom) 56

4.2 Number of $\beta_{\langle h \rangle, \langle \gamma \rangle}$ coefficients vs. the number of uncertain parameters for different Polya's exponents and for $d_p = 2$ 67

4.3 Number of $H_{\langle h \rangle, \langle \gamma \rangle}$ coefficients vs. the number of uncertain parameters for different Polya's exponents and for $d_p = d_a = 2$ 67

4.4 Memory required to store the coefficients $\beta_{\langle h \rangle, \langle \gamma \rangle}$ and $H_{\langle h \rangle, \langle \gamma \rangle}$ vs. number of uncertain parameters, for different d_1, d_2 and $d_p = d_a = 2$ 68

4.5 Graph representation of the network communication of the set-up algorithm. (a) Communication directed graph for the case $\alpha \in \Delta^3, d_p = 2$. (b) Communication directed graph for the case $\alpha \in \Delta^4, d_p = 2$. 76

4.6 Theoretical speed-up vs. No. of processors for different system dimensions n for $l = 10$, $d_p = 2$, $d_a = 3$ and $d_1 = d_2 = 4$, where $L + M = 53625$ 91

4.7 The number of blocks of the SDP elements assigned to each processor. An illustration of load balancing. 92

4.8 The communication graph of the SDP algorithm 93

4.9 Speed-up of set-up and SDP algorithms vs. number of processors for a discretized model of magnetic flux in Tokamak 98

4.10 Upper bound on optimal L vs. Polya's exponents d_1 and d_2, for different degrees of $P(\alpha)$. ($d_1 = d_2$).. 101

4.11 Error of the approximation for the optimal value of L vs. degrees of $P(\alpha)$, for different Polya's exponents 101

4.12 Computation time of the parallel set-up algorithm vs. number of processors for different dimensions of linear system n and numbers of uncertain parameters l- executed on Blue Gene supercomputer of Argonne National Labratory 102

4.13 Computation time of the parallel SDP algorithm vs. number of processors for different dimensions of primal variable $(L+M)n$ and of dual variable K- executed on Karlin cluster computer 103

4.14 Comparison between the speed-up of the present SDP solver and SDPARA 7.3.1, executed on Karlin cluster computer 104

4.15 Largest number of uncertain parameters of n-dimensional systems for which the set-up algorithm (Left) and SDP solver (Right) can solve the robust stability problem of the system using 24 and 216 Gig of RAM104

5.1 Number of operations versus dimension of the hypercube, for different Polya's exponents $d_1 = d_2 = d$. (H): hypercube and (S): simplex. 119

5.2 Required memory for the calculation of SDP elements vs. number of uncertain parameters in hypercube and simplex, for different state-space dimensions and Polya's exponents $d_1 = d_2$. (H): hypercube, (S): simplex. 120

5.3 Execution time of the set-up algorithm vs. number of processors, for different state-space dimensions n and Polya's exponents 122

6.1 An illustration of a D-decomposition of a 2D polytope. $\lambda_i(x) := h_{i,j}^T x + g_{i,j}$ for $j = 1, \cdots, m_i$. 129

6.2 Decomposition of the hypercube in $1-$,$2-$ and $3-$dimensions 141

6.3 Number of decision variables and constraints of the optimization problems associated with Algorithm 1, Polya's algorithm and SOS algorithm for different degrees of the Lyapunov function and the vector field $f(x)$ 146
6.4 The largest level-set of Lyapunov function (6.21) inscribed in Polytope (6.20) 147
6.5 Largest level sets of Lyapunov functions of different degrees and their associated parallelograms 150
6.6 Largest level sets of Lyapunov functions of different degrees and their associated polytopes 150
7.1 Effect of solar power on demand: Net loads for typical summer and winter days in Arizona in 2014 and for 2029 (projected), from Arizona Public Service (2014) 152
7.2 Peak to average demand of electricity and its trend-line in California and New England from 1993 to 2012, data adopted from Shear (2014) . 153
7.3 A schematic view of our thermal mass model 159
7.4 Simulated and measured power consumptions 160
7.5 An illustration for the programming periods of the 4-Setpoint thermostat problem, switching times t_i, pricing function r, L_i and $\overline{\Delta t_i}$. 163
7.6 External temperature of three typical summer days in Phoenix, Arizona. Shaded areas correspond to on-peak hours. 175
7.7 CASE I: Power consumption and temperature settings for various programming strategies using APS's rates. 178

Table Page

7.8 CASE I: Power consumption and optimal temperature settings for high, medium and low demand penalties. Shaded areas correspond to on-peak hours. 178

7.9 CASE I: Power consumption and temperature settings for high, medium and low demand penalties using 4-Setpoint thermostat programming. . . 179

7.10 CASE III: Power consumption, solar generated power and optimal temperature settings for the non-solar and solar users. 182

Chapter 1

INTRODUCTION

Consider problems such as portfolio optimization, path-planning, structural design, local stability of nonlinear ordinary differential equations, control of time-delay systems and control of systems with uncertainties. These problems can all be formulated as *polynomial optimization* and/or *optimization of polynomials*. In this dissertation, we show how computation can be applied in a variety of ways to solve these classes of problems. A simple example of polynomial optimization is $\beta^* = \min_{x \in Q} p(x)$, where $p : \mathbb{R}^n \to \mathbb{R}$ is a multi-variate polynomial and $Q \subset \mathbb{R}^n$. In general, since $p(x)$ and Q are not convex, this is not a convex optimization problem. In fact, it has been proved that polynomial optimization is NP-hard (L. Blum and Smale (1998)). Fortunately, algorithms such as branch-and-bound can find arbitrarily precise solutions to polynomial optimization problems by repeatedly partitioning Q into subsets Q_i and computing lower and upper bounds on $p(x)$ over each Q_i. To find an upper bound for $p(x)$ over each Q_i, one could use a local optimization algorithm such as sequential quadratic programming. To find a lower bound on $p(x)$ over each Q_i, one can solve the following optimization problem.

$$\begin{aligned} \beta^* &= \max_{y \in \mathbb{R}} \; y \\ \text{subject to} \quad & p(x) - y \geq 0 \text{ for all } x \in Q_i. \end{aligned} \tag{1.1}$$

This problem is in fact an instance of the problem of optimization of polynomials. Optimization of polynomials is convex, yet again NP-hard. We will discuss optimization of polynomials in more depth in Chapter 2. In the following, we discuss some of the state-of-the-art methods for solving optimization of polynomials - hence finding lower bounds on β^*.

1.1 Sum of squares method

One approach to find lower bounds on the optimal objective β^* is to apply *Sum of Squares (SOS) programming* (Parrilo (2000), Papachristodoulou *et al.* (2013)). A polynomial p is SOS if there exist polynomials q_i such that $p(x) = \sum_{i=1}^{r} q_i(x)^2$. The set $\{q_i \in \mathbb{R}[x], i = 1, \cdots, r\}$ is called an SOS *decomposition* of $p(x)$, where $\mathbb{R}[x]$ is the ring of real polynomials. An SOS program is an optimization problem of the form

$$\begin{aligned} \min_{x \in \mathbb{R}^m} \quad & c^T x \\ \text{subject to} \quad & A_{i,0}(y) + \sum_{j=1}^{m} x_j A_{i,j}(y) \text{ is SOS}, \; i = 1, \cdots, k, \end{aligned} \tag{1.2}$$

where $c \in \mathbb{R}^m$ and $A_{i,j} \in \mathbb{R}[y]$ are given. If $p(x)$ is SOS, then clearly $p(x) \geq 0$ on $\mathbb{R}^n$. While verifying $p(x) \geq 0$ on $\mathbb{R}^n$ is NP-hard, verifying whether $p(x)$ is SOS - hence non-negative - can be done in polynomial time (Parrilo (2000)). It was first shown in Parrilo (2000) that verifying the existence of a SOS decomposition is a Semi-Definite Program (SDP). Fortunately, there exist several algorithms (Monteiro (1997); Helmberg *et al.* (1996); Alizadeh *et al.* (1998)) and solvers (Yamashita *et al.* (2010); Sturm (1999); Tutuncu *et al.* (2003)) which solve SDPs to arbitrary precision in polynomial time. To find lower bounds on $\beta^* = \min_{x \in \mathbb{R}^n} p(x)$, consider the SOS program

$$\begin{aligned} y^* = \max_{y \in \mathbb{R}} \quad & y \\ \text{subject to} \quad & p(x) - y \text{ is SOS}. \end{aligned}$$

Clearly $y^* \leq \beta^*$. One can compute y^* by performing a bisection search on y and using semi-definite programming to verify $p(x) - y$ is SOS. SOS programming can also be used to find lower bounds on global minimum of polynomials over a semi-algebraic set $S := \{x \in \mathbb{R}^n : g_i(x) \geq 0,\ i = 1, \cdots, m\}$ generated by $g_i \in \mathbb{R}[x]$. Given Problem (1.1) with $x \in S$, *Positivstellensatz* results (Stengle (1974), Putinar (1993), Schmudgen (1991)) define a sequence of SOS programs whose objective values form a sequence of lower bounds on the global minimum β^*. For instance, Putinar's Positivstellensatz defines the optimization problem

$$\begin{aligned} y_d := \max_{y \in \mathbb{R}} \quad & y \\ \text{subject to} \quad & p(x) - y = s_0(x) + \sum_{i=1}^{m} s_i(x) g_i(x),\ \ s_i \in \Sigma_{2d}, \end{aligned} \tag{1.3}$$

where Σ_{2d} denotes the cone of SOS polynomials of degree $2d$. Putinar (1993) has shown that under certain conditions (verifiable by semi-definite programming) on S and for sufficiently large d, $y_d = \beta^*$. See Laurent (2009) for a comprehensive discussion on the Positivstellensatz results.

1.2 Moments method

As a dual to SOS program, Lasserre (2001) used the theory of *moments* to define a sequence of lower bounds for global optima of polynomials. Let $\beta^* := \min_{x \in S} p(x)$, where $S := \{x \in \mathbb{R}^n : g_i(x) \geq 0, i = 1, \cdots, m\}$ is compact and $p(x) := \sum_{\alpha \in W_p} p_\alpha x^\alpha$ with the index set $W_p := \{\alpha \in \mathbb{N}^n : \|\alpha\|_1 \leq p\}$. Let us denote the degree of g_i by e_i. Then, Lasserre (2001) showed that z_d defined as

$$\begin{aligned} z_d := \min_{z} \quad & \sum_{\alpha \in W_p} p_\alpha z_\alpha \\ \text{subject to} \quad & M_d(z) \geq 0 \\ & M_{d-e_i}(g_i\, z) \geq 0 \quad \text{for } i = 1, \cdots, m, \end{aligned} \tag{1.4}$$

is a lower bound on β^*. In Equation (1.4), $z := \{z_\alpha\}_{\alpha \in I_{2d}}$, where $z_\alpha := \int_S x^\alpha \mu(dx)$ is called the *moment of order* α and is represented by any probability measure[1] μ on $\mathbb{R}^n$ such that $\mu(\mathbb{R} \backslash S) = 0$. Moreover, $M_d(z)$ is called the *moment matrix* associated with sequence z and in two dimensions is defined as

$$M_d(z) = \left(\begin{array}{c:cc:ccc:c:ccc} 1 & z_{[1,0]} & z_{[0,1]} & z_{[2,0]} & z_{[1,1]} & z_{[0,2]} & \cdots & z_{[d,0]} & \cdots & z_{[0,d]} \\ \hdashline z_{[1,0]} & z_{[2,0]} & z_{[1,1]} & z_{[3,0]} & z_{[2,1]} & z_{[1,2]} & \cdots & z_{[d+1,0]} & \cdots & z_{[1,d]} \\ z_{[0,1]} & z_{[1,1]} & z_{[0,2]} & z_{[2,1]} & z_{[1,2]} & z_{[0,3]} & \cdots & z_{[d,1]} & \cdots & z_{[0,d+1]} \\ \hdashline z_{[2,0]} & z_{[3,0]} & z_{[2,1]} & z_{[4,0]} & z_{[3,1]} & z_{[2,2]} & \cdots & z_{[d+2,0]} & \cdots & z_{[2,d]} \\ z_{[1,1]} & z_{[2,1]} & z_{[1,2]} & z_{[3,1]} & z_{[2,2]} & z_{[1,3]} & \cdots & z_{[d+1,1]} & \cdots & z_{[1,d+1]} \\ z_{[0,2]} & z_{[1,2]} & z_{[0,3]} & z_{[2,2]} & z_{[1,3]} & z_{[0,4]} & \cdots & z_{[d,2]} & \cdots & z_{[0,d+2]} \\ \hdashline \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \cdots & \cdots & \cdots \\ \hdashline z_{[d,0]} & z_{[d+1,0]} & z_{[d,1]} & z_{[d+2,0]} & z_{[d+1,1]} & z_{[d,2]} & \cdots & z_{[2d,0]} & \cdots & z_{[d,d]} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ z_{[0,d]} & z_{[1,d]} & z_{[0,d+1]} & z_{[2,d]} & z_{[1,d+1]} & z_{[0,d+2]} & \cdots & z_{[d,d]} & \cdots & z_{[0,2d]} \end{array} \right).$$

It can be shown that the SDPs in (1.4) are duals to the SDPs in (1.3) - implying that $y_d \leq z_d$. Indeed, if S has a non-empty interior, then for all sufficiently large d, the duality gap is zero, i.e., $y_d = z_d$. See Laurent (2009) and Jeyakumar *et al.* (2014) for conditions on convergence of the lower bounds to global minima and extension of moments method to polynomial optimization over non-compact semi-algebraic sets.

In the sequel, we explore the merits of some of the alternatives to SOS programming and moments method. There exist several results in the literature that can be applied to polynomial optimization; e.g., Quantifier Elimination (QE) algo-

[1] Let X be a set and M be a $\sigma-$algebra over X. Then $\mu : M \to [0,1]$ is a probability measure if

1. $\mu(\emptyset) = 0$ and $\mu(X) = 1$.

2. For all countable collections $\{S_i\}_{i \in N}$ of pairwise disjoint subsets of M, $\mu(\bigcup_{i \in N} S_i) = \sum_{i \in N} \mu(S_i)$.

rithms (Collins and Hoon (1991)) for testing the feasibility of semi-algebraic sets, Reformulation Linear Techniques (RLTs) (Sherali and Tuncbilek (1992, 1997)) for linearizing polynomial optimizations, Polya's theorem (G. Hardy and Polya (1934)) for positivity over the positive orthant, Bernstein's (Boudaoud *et al.* (2008); Leroy (2012)) and Handelman's (Handelman (1988a)) theorems for positivity over simplices and convex polytopes, and other results based on Groebner bases (Adams and Loustaunau (1994)) and Blossoming (Ramshaw (1987)) techniques. In particular, we will focus on Polya's, Bernstein's and Handelman's results in more depth and elaborate on the computational advantages of these results over the others. The discussion of the other results are beyond the scope of this dissertation, however the ideas behind these results can be summarized as follows.

1.3 Quantifier elimination

QE algorithms apply to First-Order Logic formulae, e.g.,

$$\forall x \, \exists y \, (f(x, y) \geq 0 \Rightarrow ((g(a) < xy) \wedge (a > 2)),$$

to eliminate the *quantified* variables x and y (preceded by quantifiers $\forall, \exists$) and construct an equivalent formula in terms of the *unquantified* variable a. The key result underlying QE algorithms is Tarski-Seidenberg theorem (Tarski (1951)). The theorem implies that for every formula of the form $\forall x \in \mathbb{R}^n \, \exists y \in \mathbb{R}^m (f_i(x, y, a) \geq 0)$, where $f_i \in \mathbb{R}[x, y, a]$, there exists an equivalent quantifier-free formula of the form $\wedge_i (g_i(a) \geq 0) \vee_j (h_j(a) \geq 0)$ with $g_i, h_j \in \mathbb{R}[a]$. QE implementations (e.g., Brown (2003) and Dolzmann and Sturm (1997)) with a bisection search yields the exact solution to optimization of polynomials, however the complexity scales double exponentially in the dimension of variables x, y.

1.4 Reformulation linear techniques

RLT was initially developed to find the convex hull of feasible solutions of zero-one linear programs (Sherali and Adams (1990)). It was later generalized by Sherali and Tuncbilek (1992) to address polynomial optimizations of the form $\min_x p(x)$ subject to $x \in [0,1]^n \cap S$. RLT constructs a $\delta-$hierarchy of linear programs by performing two steps. In the first step (reformulation), RLT introduces the new constraints $\prod_i x_i \prod_j (1 - x_j) \geq 0$ for all $i, j : i + j = \delta$. In the second step (linearization), RTL defines a linear program by replacing every product of variables x_i by a new variable. By increasing δ and repeating the two steps, one can construct a $\delta-$hierarchy of lower bounding linear programs. A combination of RLT and branch-and-bound partitioning of $[0,1]^n$ was developed by Sherali and Tuncbilek (1997) to achieve tighter lower bounds on the global minimum. For a survey of different extensions of RLT see Sherali and Liberti (2009).

1.5 Groebner basis technique

Groebner bases can be used to reduce a polynomial optimization over a semi-algebraic set $S := \{x \in \mathbb{R}^n : g_i(x) \geq 0,\ h_j(x) = 0\}$ to the problem of finding the roots of univariate polynomials (Chang and Wah (1994)). First, one needs to construct the system of polynomial equations

$$[\nabla_x L(x, \lambda, \mu), \nabla_\lambda L(x, \lambda, \mu), \nabla_\mu L(x, \lambda, \mu)] = 0, \tag{1.5}$$

where $L := p(x) + \sum_i \lambda_i g_i(x) + \sum_j \mu_j h_j(x)$ is the Lagrangian function. It is well-known that the set of solutions to (1.5) is the set of extrema of the polynomial optimization $\min_{x \in S} p(x)$. Let

$$[f_1(x, \lambda, \mu), \cdots, f_N(x, \lambda, \mu)] := [\nabla_x L(x, \lambda, \mu), \nabla_\lambda L(x, \lambda, \mu), \nabla_\mu L(x, \lambda, \mu)].$$

Using the elimination property (Adams and Loustaunau (1994)) of the Groebner bases, the minimal Groebner basis of the ideal of $f_1, \cdots, f_N$ defines a triangular-form system of polynomial equations. This system can be solved by calculating one variable at a time and back-substituting into other polynomials. The most computationally expensive part is the calculation of the Groebner basis, which in the worst case scales double-exponentially in the number of decision variables.

1.6 Blossoming technique

The blossoming technique involves a bijective map between the space of polynomials $p : \mathbb{R}^n \to \mathbb{R}$ and the space of multi-affine functions $q : \mathbb{R}^{d_1+d_2+\cdots+d_n} \to \mathbb{R}$ (polynomials that are affine in each variable), where d_i is the degree of p in variable x_i. For instance, the blossom of a cubic polynomial $p(x) = ax^3 + bx^2 + cx + d$ is the multi-affine function

$$q(z_1, z_2, z_3) = az_1z_2z_3 + \frac{b}{3}(z_1z_2 + z_1z_3 + z_2z_3) + \frac{c}{3}(z_1 + z_2 + z_3) + d.$$

It can be shown that the blossom, q, of any polynomial $p \in \mathbb{R}[x]$ with degree d_i in variable x_i satisfies the so-called *diagonal* property (Ramshaw (1987)), i.e.,

$$p(z_1, z_2, \cdots, z_n) = q(\underbrace{z_1, \cdots, z_1}_{d_1 \text{ times}}, \cdots, \underbrace{z_n, \cdots, z_n}_{d_n \text{ times}}) \quad \text{for all } z \in \mathbb{R}.$$

By using this property, one can reformulate any polynomial optimization $\min_{x\in S} p(x)$ as

$$\begin{aligned} &\min_{z\in Q} \quad q(z) \\ &\text{subject to} \quad z_{\phi(i)} = z_{\phi(i)-j} \quad \text{for } i = 1, \cdots, n \text{ and for } j = 1, \cdots, d_i - 1, \end{aligned} \tag{1.6}$$

where $\phi(i) := \sum_{k=1}^{i} d_i$ and Q is the semi-algebraic set defined by the blossoms of the generating polynomials of S. In the special case, where S is a hypercube, Sassi and Girard (2012) showed that the Lagrangian dual optimization problem to Problem (1.6) is a linear program. Hence, the optimal objective value of this linear program is a lower bound on the minimum of $p(x)$ over the hypercube. Application of blossoming in estimation of reachability sets of discrete-time dynamical systems can be found in Sassi *et al.* (2012).

1.7 Bernstein's, Polya's and Handelman's theorems

While QE, RLT, Groebner bases and blossoming are all useful techniques with advantages and disadvantages (such as exponential complexity), we focus on Polya's, Bernstein's and Handelman's theorems - results which yield polynomial-time tests for positivity of polynomials. Polya's theorem yields a basis to represent the cone of polynomials that are positive over the positive orthant. Bernstein's and Handelman's theorems yield bases which represent the cones of polynomials that are positive over simplices and convex polytopes, respectively. Similar to SOS programming, one can find certificates of positivity using Polya's, Bernstein's and Handelman's representations by solving a sequence of Linear Programs (LPs) and/or SDPs. However, unlike the SDPs associated with SOS programming, the SDPs associated with these theorems have a block-diagonal structure. In this dissertation, we exploit this structure to design parallel algorithms for optimization of polynomials of high degrees with several independent variables. See Kamyar and Peet (2012a), Kamyar and Peet (2012b), Kamyar and Peet (2013) and Kamyar *et al.* (2013) for parallel implementations of variants of Polya's theorem applied to various Lyapunov inequalities.

Unfortunately, unlike the SOS methodology, the bases given by Polya's theorem, Bernstein's theorem and Handelman's theorem cannot be used to represent the cone of non-negative polynomials which have zeros in the interior of simplices and polytopes. This is indeed a barrier against using these theorems to compute polynomial Lyapunov functions, since Lyapunov functions, by definition, have a zero at the origin. There do, however, exist some variants of Polya's theorem which consider zeros at the corners Powers and Reznick (2006) and edges Castle *et al.* (2011) by constructing local certificates of non-negativity over closed subsets, C_i, of the simplex such that $\cup C_i$ is the simplex. These results apply to non-negative polynomials whose zeros are on the corners and/or edges of the simplex. Moreover, Oliveira *et al.* (2008) and Kamyar and Peet (2012b) propose versions of Polya's theorem which prove positivity over hypercubes by: 1) Providing certificates of positivity on the Cartesian product of unit simplices; and 2) Introducing a one-to-one map between products of unit simplices (multi-simplex) and hypercubes. A generalization of Polya's theorem for proving positivity on the entire $\mathbb{R}^n$ was introduced by de Loera and Santos (1996). This generalization first applies Polya's theorem to each orthant of $\mathbb{R}^n$ to compute a certificate of positivity over each orthant. Then, it uses the merging technique in Lombardi (1991) to obtain a unified certificate - in the form of SOS of rationals - over $\mathbb{R}^n$. A recent extension of Polya's theorem by Dickinson and Pohv (2014) can be used to prove positivity over an intersection of a semi-algebraic set with the positive orthant. Finally, positivity of polynomials with rational exponents can be verified by a weak version of Polya's theorem in Delzell (2008).

1.8 Motivations and summary of contributions

The novelty of our research centers on the areas of: computation and energy. In the realm of computation, we observed that processors speeds are not growing at the rate they once were. The entire controls community seems to have ignored this fact, since everyone speaks of polynomial-time algorithms as the gold standard for what the solution to a control problem should look like. But what good is a polynomial-time algorithm when the degree of the polynomial is bounded by the current state-of-the-art computers. Our solution was to look at the only area where the computing world was getting faster (growing) - supercomputers. Surprisingly, there have been no studies on the use of parallel computers for controls since the 1970's. The reason was that the mathematical machinery for analysis and control is based on Semidefinite Programming, which is inherently sequential (NC-hard). Our idea, however, was that if the SDP problem has special structure, then this structure can be exploited to distribute computation among processors. With this in mind, we decided to seek out alternatives to the classical Sum-of-Squares approach to nonlinear and robust stability analyses. We identified more than seven different alternatives to the Sum-of-Squares approach. In the end, not all of these had usable structures for parallelization. However, we identified three which did: polynomial positivity results by Handelman, Polya and Bernstein. To demonstrate how well this approach works in practice, we developed a Message Passing Interface code for Polya's theorem. The result enabled stability analysis for systems three times larger (in terms of number of states) than any other algorithm. As a real-world application, we further used our code to analyze robust stability of plasma in the Tore Supra Tokamak reactor.

In the realm of energy, we noticed that the two electrical utility companies of Arizona (APS and SRP) have recently started charging their customers for their maximum rate of electricity usage. This intrigued us as a mathematical problem of how to optimize the thermostat settings of HVAC systems (the major sources of electricity consumption in Arizona) in order to minimize the electricity bill. This problem is interesting in that the time of peak electricity use is not usually at the hottest time of day, but rather a couple of hours after - a behaviour which is usually associated with a diffusion PDE. We used the heat equation to model the thermostat programming problem as an optimal control problem and it turned out to be unsolved. The mathematical reason being that the cost function is not separable in time - a property which is necessary for optimal control algorithms to converge to an optimal solution. We noticed that an arbitrarily precise approximation of the cost function however, satisfy certain properties which make it solvable on a Pareto-optimal front. The result is an optimal thermostat which can significantly reduce the electricity bills and peak demand of both solar and nonsolar customers under the current pricing plans. Expanding this approach, we started thinking about related topics, such as how to set the demand price on order to influence customers' behavior in an optimal manner. Based on that, we proposed an optimal pricing algorithm which resulted in a moderate reduction in the cost of generating, transmission and distribution of electricity at SRP.

We highlight our contributions as follows. In Chapter 4, we propose a parallel set-up algorithm which applies Polya's theorem to the parameter-dependent Lyapunov inequalities $P(\alpha) > 0$ and $A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha) < 0$ with α belonging to the standard simplex. Feasibility of these inequalities implies robust stability of the system of linear Ordinary Differential Equations (ODEs) $\dot{x}(t) = A(\alpha)x(t)$ over the simplex. The output of our set-up algorithm is a sequence of SDPs of increasing size and precision. A solution to any of these SDPs yield a Lyapunov function which is quadratic in the states and depends polynomially on the uncertain parameters. An interesting property of these SDPs is that they possess a block-diagonal structure. We show how this structure can be exploited to design a parallel *interior-point primal-dual* SDP solver which distributes the computation of search direction among a large number of processors. We then produce a Message Passing Interface (MPI) implementation of our set-up and solver algorithms. Through numerical experiments, we show that these algorithms achieve a near-linear theoretical and experimental speed-up (the increase in processing speed per additional processor). Moreover, our numerical experiments on cluster computers demonstrate the ability of our algorithms in utilizing hundreds and potentially thousands of processors to analyze systems with 100+ dimensional state-space.

In Chapter 5, we generalize our methodology to perform robust stability analysis over hypercubes. We first propose an extended version of Polya's theorem. This theorem parameterizes every homogeneous polynomial which is positive over a hypercube. We then propose an extended set-up algorithm which maps the computation and memory - associated with applying the extended Polya's theorem to stability analysis problems - to parallel machines. This set-up algorithm has no centralized computation and its per-core communication complexity scales polynomially with the state-space dimension and the number of uncertain parameters. As the result, it demonstrates a near-linear speed-up.

In Chapter 6, we further extend our analysis to address stability of nonlinear ODEs defined by a polynomial vector field f. Our proposed solution to this problem is to reformulate the nonlinear stability problem using only strictly positive forms. Specifically, we use our extended version of Polya's theorem in Chapter 5 to compute a matrix-valued homogeneous polynomial $P(x)$ such that $P(x) > 0$ and $\langle \nabla(x^T P(x)x), f(x) \rangle < 0$ for all x inside a hypercube containing the origin in its interior. This yields a Lyapunov function of the form $V(x) = x^T P(x)x$ for the system $\dot{x}(t) = f(x(t))$. To do this, we design a new parallel set-up algorithm which applies Polya's theorem to the inequalities $P(x) > 0$ and $\langle \nabla(x^T P(x)x), f(x) \rangle < 0$. The result is a sequence of SDPs with coefficients of P as decision variables. Again, we show that these SDPs have a block-diagonal structure - thus can be solved in parallel using our SDP solver in Chapter 4. As an extension to stability analysis over arbitrary convex polytopes, we then propose an algorithm which applies Handelman's theorem to the aforementioned Lyapunov inequalities. Unfortunately, as in the case of Polya's theorem, Handelman's theorem is incapable of parameterizing polynomials which possess zeros in the interior of a polytope. However, we show that this is not the case if the zeros are on the vertices of the polytope. By using this property, we propose the following methodology: 1) Decompose the polytope into several convex sub-polytopes with a common vertex on the equilibrium; 2) Apply Handelman's theorem to Lyapunov inequalities defined on each sub-polytope. The result is a sequence of linear programs whose solutions define a piecewise polynomial Lyapunov function V- hence proving asymptotic stability over the sublevel-set of V inscribed in the original polytope. We provide a comprehensive comparison between the computational complexities of SOS algorithm, our Polya's algorithms and our Handelman algorithm. Our analysis shows that by using a certain decomposition scheme, our algorithm (based on Handelman's theorem) has the lowest computational complexity compared to the SOS and Polya's algorithms.

Chapter 2

FUNDAMENTAL RESULTS FOR OPTIMIZATION OF POLYNOMIALS

In this chapter, we first provide an overview of fundamental theorems on positivity of polynomials over various sets. Then, we show how applying these theorems to optimization of polynomials problems of the Form (1.1) yields tractable convex optimization problems in the forms of LPs and/or SDPs. Any solution to these LPs and/or SDPs yields a lower-bound on the global minimum of the polynomial optimization problem $\min_{x \in Q} \; p(x)$.

2.1 Background on positivity results

In 1900, Hilbert published a list of mathematical problems, one of which is: For every non-negative $f \in \mathbb{R}[x]$, does there exist any non-zero $q \in \mathbb{R}[x]$ such that $q^2 f$ is a sum of squares? In other words, is every non-negative polynomial a sum of squares of rational functions? This question was motivated by his earlier works (Hilbert (1888, 1893)), in which he proved: 1) Every non-negative bi-variate degree 4 *homogeneous* polynomial (A polynomial whose monomials all have the same degree) is a SOS of three polynomials; 2) Every bi-variate non-negative polynomial is a SOS of four rational functions; 3) Not every non-negative homogeneous polynomial with more than two variables and degree greater than 5 is SOS of polynomials. While there exist systematic ways (e.g., semi-definite programming) to prove that a non-negative polynomial is SOS, proving that a non-negative polynomial is not a SOS of polynomials is not straightforward. Indeed, the first example of a non-negative non-SOS polynomial was published eighty years after Hilbert posed his 17^{th} problem. Motzkin (1967) constructed a PSD degree 6 polynomial with three variables which is not SOS:

$$M(x_1, x_2, x_3) = x_1^4 x_2^2 + x_1^2 x_2^4 - 3x_1^2 x_2^2 x_3^2 + x_3^6. \tag{2.1}$$

Non-negativity of M follows directly from the inequality of arithmetic and geometric means, i.e., $(a_1 + \cdots + a_n)/n \geq \sqrt[n]{a_1 \cdots a_n}$, by letting $n = 3, a_1 = x_1^4 x_2^2, a_2 = x_1^2 x_2^4$ and $a_3 = x_3^6$. To show that M is not SOS, first by contradiction suppose that there exist some $N \in \mathbb{N}$ and coefficients $b_{i,j} \in \mathbb{R}$ such that

$$M(x_1, x_2, x_3) = \sum_{i=1}^{N} \left(\begin{bmatrix} b_{i,1} & \cdots & b_{i,20} \end{bmatrix} \begin{bmatrix} x_1^3 & x_1^2 x_2 & x_1^2 x_3 & x_1 x_2^2 & x_1 x_2 x_3 & x_1 x_3^2 & x_2^3 \\ x_2^2 x_3 & x_2 x_3^2 & x_3^3 & x_1^2 & x_1 x_2 & x_2 x_3 & x_2^2 & x_2 x_3 & x_3^2 & x_1 & x_2 & x_3 & 1 \end{bmatrix}^T \right)^2. \tag{2.2}$$

By substituting (2.1) in (2.2) and equating the coefficients of both sides of (2.2), it follows that $\sum_{i=1}^{N} b_{i,5}^2 = -3$. This is a contradiction, thus M is not SOS of polynomials. A generalization of Motzkin's example is given by Robinson (Reznick (2000)). Polynomials of the form $(\prod_{i=1}^{n} x_i^2) f(x_1, \cdots, x_n) + 1$ are not SOS if polynomial f of degree $< 2n$ is not SOS. Hence, although the non-homogeneous Motzkin polynomial $M(x_1, x_2, 1) = x_1^2 x_2^2 (x_1^2 + x_2^2 - 3) + 1$ is non-negative it is not SOS.

Artin (1927) answered Hilbert's problem in the following theorem.

Theorem 1. *(Artin's theorem) A polynomial $f \in \mathbb{R}[x]$ satisfies $f(x) \geq 0$ on $\mathbb{R}^n$ if and only if there exist SOS polynomials N and $D \neq 0$ such that $f(x) = \frac{N(x)}{D(x)}$.*

Although Artin settled Hilbert's problem, his proof was neither constructive nor gave a characterization of the numerator N and denominator D. In 1939, Habicht provided some structure on N and D for a certain class of polynomials f. Habicht (1939) showed that if a homogeneous polynomial f is positive definite and can be expressed as $f(x_1, \cdots, x_n) = g(x_1^2, \cdots, x_n^2)$ for some polynomial g, then one can choose the denominator $D = \sum_{i=1}^{n} x_i^2$. Moreover, he showed that by using $D = \sum_{i=1}^{n} x_i^2$, the numerator N can be expressed as a sum of squares of monomials. Habicht used Polya's theorem (Hardy *et al.* (1934), Theorem 56) to obtain the above characterizations for N and D.

Theorem 2. *(Polya's theorem) Suppose a homogeneous polynomial p satisfies $p(x) > 0$ for all $x \in \{x \in \mathbb{R}^n : x_i \geq 0, \sum_{i=1}^n x_i \neq 0\}$. Then $p(x)$ can be expressed as*

$$p(x) = \frac{N(x)}{D(x)},$$

where $N(x)$ and $D(x)$ are homogeneous polynomials with all positive coefficients. Furthermore, for every homogeneous $p(x)$ and some $e \geq 0$, the denominator $D(x)$ can be chosen as $(x_1 + \cdots + x_n)^e$.

To see Habicht's result, suppose f is homogeneous and positive on the positive orthant and can be expressed as $f(x_1, \cdots, x_n) = g(x_1^2, \cdots, x_n^2)$ for some homogeneous polynomial g. By using Polya's theorem, $g(y) = \frac{N(y)}{D(y)}$, where $y := (y_1, \cdots, y_n)$ and polynomials N and D have all positive coefficients. Furthermore, from Theorem 2 we may choose $D(y) = \left(\sum_{i=1}^n y_i\right)^e$. Then, $\left(\sum_{i=1}^n y_i\right)^e g(y) = N(y)$. Now let $x_i = \sqrt{y_i}$, then $\left(\sum_{i=1}^n x_i^2\right)^e f(x_1, \cdots, x_n) = N(x_1^2, \cdots, x_n^2)$. Since N has all positive coefficients, $N(x_1^2, \cdots, x_n^2)$ is a sum of squares of monomials.

Similar to the case of positive definite polynomials, ternary positive semi-definite polynomials of the form $g(x_1^2, x_2^2, x_3^2)$ can be parameterized using the denominator $D = (x_1^2 + x_2^2 + x_3^2)^N$ (Scheiderer (2006)). However, in any dimension higher than three, there exist positive semi-definite polynomials f such that if $h^2 f$ is SOS, then h has a zero other than the origin. Thus, for such polynomials f, Df cannot be SOS. Indeed, it has been shown by Reznick (2005) that there exists no single SOS polynomial $D \neq 0$ which satisfies $f = \frac{N}{D}$ for every positive semi-definite f and some SOS polynomial N.

As in the case of positivity on $\mathbb{R}^n$, there has been an extensive research regarding positivity of polynomials on bounded sets. A pioneering result on local positivity is Bernstein's theorem (Bernstein (1915)). Bernstein's theorem uses the polynomials $h_{i,j} = (1+x)^i(1-x)^j$ as a basis to parameterize univariate polynomials which are positive on $[-1,1]$.

Theorem 3. *(Bernstein's theorem) If a polynomial* $f(x) > 0$ *on* $[-1,1]$*, then there exist* $c_{i,j} > 0$ *such that*

$$f(x) = \sum_{i,j\in\mathbb{N}:\, i+j=d} c_{i,j}(1+x)^i(1-x)^j$$

for some $d > 0$*.*

Powers and Reznick (2000) used Goursat's transformation of f to find an upper bound on d. Unfortunately, the bound itself is a function of the minimum of f on $[-1,1]$. In order to reduce the computational complexity of testing positivity, Boudaoud *et al.* (2008) proposed a decomposition scheme for breaking $[-1,1]$ into a collection of sub-intervals. Subsequently, Bernstein's theorem was applied to f over each sub-interval to find a certificate of positivity over each sub-interval. An extension of this technique was proposed in Leroy (2012) to verify positivity over simplices (a simplex is the convex hull of $n+1$ vertices in $\mathbb{R}^n$). Moreover, Leroy (2012) provided a degree bound as a function of the minimum of f over the simplex, the number of variables in f, the degree of f and the maximum of certain affine combinations of the coefficients $c_{i,j}$.

Handelman (1988b) also used products of affine functions as a basis (the Handelman basis) to extend Bernstein's theorem to multi-variate polynomials which are positive on convex polytopes.

Theorem 4. *(Handelman's Theorem) Given* $w_i \in \mathbb{R}^n$ *and* $u_i \in \mathbb{R}$*, define the polytope* $\Gamma^K := \{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \cdots, K\}$*. If a polynomial* $f(x) > 0$ *on* Γ^K*, then* *there exist* $b_\alpha \geq 0$*,* $\alpha \in \mathbb{N}^K$ *such that for some* $d \in \mathbb{N}$*,*

$$f(x) = \sum_{\substack{\alpha \in \mathbb{N}^K \\ \alpha_1 + \cdots + \alpha_K \leq d}} b_\alpha (w_1^T x + u_1)^{\alpha_1} \cdots (w_K^T x + u_K)^{\alpha_K}. \tag{2.3}$$

Recently, S. Sankaranarayanan and Abrahm (2013) combined the Handelman basis with positive basis functions

$$x_1^{\alpha_1} \cdots x_n^{\alpha_n} - l_\alpha \quad \text{and} \quad u_\alpha - x_1^{\alpha_1} \cdots x_n^{\alpha_n}$$

to compute Lyapunov functions over a hypercube Φ, where l_α and u_α are the minimum and maximum of $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ over the hypercube Φ. A generalization of Handelman's theorem was made by Schweighofer (2005) to verify non-negativity of polynomials over compact semi-algebraic sets. Schweighofer used the cone of polynomials[1] defined in (2.5) to parameterize any polynomial f which has the following properties:

1. f is non-negative over the compact semi-algebraic set S defined in (2.4)

2. $f = q_1 p_1 + q_2 p_2 + \cdots$ for some q_i in the Cone (2.5) and for some $p_i > 0$ over

$$S \cap \{x \in \mathbb{R}^n : f(x) = 0\}$$

Theorem 5. *(Schweighofer's theorem) Suppose*

$$S := \{x \in \mathbb{R}^n : g_i(x) \geq 0, g_i \in \mathbb{R}[x] \text{ for } i = 1, \cdots, K\} \tag{2.4}$$

is compact. Define the following set of polynomials which are positive on S*.*

$$\Theta_d := \left\{ \sum_{\lambda \in \mathbb{N}^K : \lambda_1 + \cdots + \lambda_K \leq d} s_\lambda g_1^{\lambda_1} \cdots g_K^{\lambda_K} : s_\lambda \text{ are SOS} \right\} \tag{2.5}$$

If $f \geq 0$ *on* S *and there exist* $q_i \in \Theta_d$ *and polynomials* $p_i > 0$ *on* $S \cap \{x \in \mathbb{R}^n : f(x) = 0\}$ *such that* $f = \sum_i q_i p_i$ *for some* d*, then* $f \in \Theta_d$*.*

[1] A set of polynomials $S \subset \mathbb{R}[x_1, \cdots, x_n]$ is a *cone* if: 1) $f_1 \in S$ and $f_2 \in S$ imply $f_1 f_2 \in S$ and $f_1 + f_2 \in S$; and 2) $f \in \mathbb{R}[x_1, \cdots, x_n]$ implies $f^2 \in S$.

On the assumption that g_i are affine functions, $p_i = 1$ and s_λ are constant, Schweighofer's theorem gives the same parameterization of f as in Handelman's theorem. Another special case of Schweighofer's theorem is when $\lambda \in \{0,1\}^K$. In this case, Schweighofer's theorem reduces to Schmudgen's Positivstellensatz (Schmudgen (1991)). Schmudgen's Positivstellensatz states that the cone

$$\Lambda_g := \left\{ \sum_{\lambda \in \{0,1\}^K} s_\lambda g_1^{\lambda_1} \cdots g_K^{\lambda_K} : s_\lambda \text{ are } SOS \right\} \subset \Theta_d \tag{2.6}$$

is sufficient to parameterize every $f > 0$ over the semi-algebraic set S generated by $\{g_1, \cdots, g_K\}$. Unfortunately, the cone Λ_g contains 2^K products of g_i, thus finding a representation of Form (2.6) for f requires a search for at most 2^K SOS polynomials. Putinar's Positivstellensatz (Putinar (1993)) reduces the complexity of Schmudgen's parameterization in the case where the quadratic module M_g (as defined in (2.8)) of polynomials g_i is *Archimedean*, i.e., there exists $N \in \mathbb{N}$ such that

$$N - \sum_{i=1}^{n} x_i^2 \in M_g. \tag{2.7}$$

Equivalently, if there exists some $f \in M_g$ such that $\{x \in \mathbb{R}^n : f(x) \geq 0\}$ is compact, then M_g is Archimedean.

Theorem 6. *(Putinars's Positivstellensatz) Let $S := \{x \in \mathbb{R}^n : g_i(x) \geq 0, g_i \in \mathbb{R}[x]$ for $i = 1, \cdots, K\}$ and define*

$$M_g := \left\{ s_0 + \sum_{i=1}^{K} s_i g_i : s_i \text{ are } SOS \right\}. \tag{2.8}$$

If there exist some $N > 0$ such that $N - \sum_{i=1}^{n} x_i^2 \in M_g$, then M_g is Archimedean. If M_g is Archimedean and $f > 0$ over S, then $f \in M_g$.

Finding a representation of Form (2.8) for f, only requires a search for $K+1$ SOS polynomials using SOS programming. Verifying the Archimedian Condition (2.7) is also an SOS program. Observe that if M_g is not Archimedean, one can add a redundant constraint $r - \sum_{i=1}^n x_i^2 \geq 0$ (for sufficiently large $r \in \mathbb{R}$) to S in order to make M_g Archimedean. Archimedean condition clearly implies compactness of the semi-algebraic set S because for any $f \in M_g$, $S \subset \{x \in \mathbb{R}^n : f(x) \geq 0\}$. The following theorem lifts the compactness requirement for the semi-algebraic set S.

Theorem 7. *(Stengle's Positivstellensatz) Let $S := \{x \in \mathbb{R}^n : g_i(x) \geq 0, g_i \in \mathbb{R}[x]$ for $i = 1, \cdots, K\}$ and define the cone*

$$\Lambda_g := \left\{ \sum_{\lambda \in \{0,1\}^K} s_\lambda g_1^{\lambda_1} \cdots g_K^{\lambda_K} : s_\lambda \text{ are SOS} \right\}.$$

If $f > 0$ on S, then there exist $p, g \in \Lambda_g$ such that $qf = p + 1$.

Notice that the Parameterziation (2.3) in Handelman's theorem is affine in f and the coefficients b_α. Likewise, the parameterizations in Theorems 5 and 6, i.e., $f = \sum_\lambda s_\lambda g_1^{\lambda_1} \cdots g_K^{\lambda_K}$ and $f = s_0 + \sum_i s_i g_i$ are affine in f, s_λ and s_i. Thus, one can use convex optimization to find b_α, s_λ, s_i and f efficiently. Unfortunately, since the parameterization $qf = p + 1$ in Stengle's Positivstellensatz is non-convex (bilinear in q and f), it is more difficult to verify $qf = p + 1$ compared to Handelman's and Putinar's parameterizations.

For a comprehensive discussion on the Positivstellensatz and other results on polynomial positivity in algebraic geometry see Laurent (2009); Scheiderer (2009), and Prestel and Delzell (2004).

2.2 Polynomial optimization and optimization of polynomials

Given $f, g_i, h_j \in \mathbb{R}[x]$ for $i = 1, \cdots, m$ and $j = 1, \cdots, r$, define a semi-algebraic set S as

$$S := \{y \in \mathbb{R}^n : g_i(y) \geq 0,\ h_j(y) = 0 \text{ for } i = 1, \cdots, m \text{ and } j = 1, \cdots, r\}. \tag{2.9}$$

We then define *polynomial optimization* problems as

$$\beta^* = \min_{x \in S} \; f(x). \tag{2.10}$$

For example, the integer program

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} && p(x) \\ &\text{subject to} && a_i^T x \geq b_i \text{ for } i = 1, \cdots, m, \\ & && x \in \{-1, 1\}^n, \end{aligned} \tag{2.11}$$

with given $a_i \in \mathbb{R}^n, b_i \in \mathbb{R}$ and $p \in \mathbb{R}[x]$, can be formulated as a polynomial optimization problem by setting $f = p$ in (2.10) and setting

$$\begin{aligned} g_i(x) &= a_i^T x - b_i && \text{for } i = 1, \cdots, m \\ h_j(x) &= x_j^2 - 1 && \text{for } j = 1, \cdots, n. \end{aligned}$$

in the definition of S in (2.9).

Given $c \in \mathbb{R}^n$ and g_i, $h_j \in \mathbb{R}[x]$ for $i = 1, \cdots, m$ and $j = 1, \cdots, r$, we define *Optimization of polynomials* problems as

$$\begin{aligned} &\gamma^* = \max_{x\in\mathbb{R}^q} && c^T x \\ &\text{subject to} && F(x, y) := F_0(y) + \sum_{i=1}^{q} x_i F_i(y) \geq 0 \text{ for all } y \in S, \end{aligned} \tag{2.12}$$

where S is defined in (2.9) and

$$F_i(y) := \sum_{\alpha \in E_{d_i}} F_{i,\alpha} y_1^{\alpha_1} \cdots y_n^{\alpha_n}$$

with $E_{d_i} := \{\alpha \in \mathbb{N}^n : \sum_{i=1}^n \alpha_i \leq d_i\}$, where coefficients $F_{i,\alpha} \in \mathbb{R}^{t\times t}$, $i = 0, \cdots, q$ are given. If the goal is to optimize over a polynomial variable, $p(y)$, this may be achieved using a basis of monomials for $F_i(y)$ so that the polynomial variable becomes

$p(y) = \sum_i x_i F_i(y)$. Optimization of polynomials can be used to find β^* in (2.10). For example, we can compute the optimal objective value η^* of the polynomial optimization problem

$$\begin{aligned} \eta^* = \min_{x \in \mathbb{R}^n} \quad & p(x) \\ \text{subject to} \quad & a_i^T x - b_i \geq 0 \qquad \text{for } i = 1, \cdots, m, \\ & x_j^2 - 1 = 0 \qquad \text{for } j = 1, \cdots, n, \end{aligned}$$

by solving the problem

$$\begin{aligned} \eta^* = \max_{\eta \in \mathbb{R}} \quad & \eta \\ \text{subject to} \quad & p(y) \geq \eta \qquad \text{for } y \in \{y \in \mathbb{R}^n : a_i^T y \geq b_i,\, y_j^2 - 1 = 0 \text{ for } i = 1, \cdots, m \\ & \qquad\qquad\qquad \text{and } j = 1, \cdots, n\}, \end{aligned} \tag{2.13}$$

where Problem (2.13) can be expressed in the Form (2.12) by setting

$$c = 1, \qquad q = 1, \qquad t = 1, \qquad F_0 = p \qquad F_1 = -1,$$

$$S := \{y \in \mathbb{R}^n : a_i^T y \geq b_i,\, y_j^2 - 1 = 0 \text{ for } i = 1, \cdots, m, \text{ and } j = 1, \cdots, n\}.$$

Optimization of polynomials (2.12) can be reformulated as the feasibility problem

$$\begin{aligned} \gamma^* = \min_{\gamma} \quad & \gamma \\ \text{subject to } S_\gamma := & \Big\{x \in \mathbb{R}^q : c^T x > \gamma,\, F(x, y) \geq 0 \text{ for all } y \in S\Big\} = \emptyset, \end{aligned} \tag{2.14}$$

where c and F are given and

$$S := \{y \in \mathbb{R}^n : g_i(y) \geq 0,\, h_j(y) = 0 \text{ for } i = 1, \cdots, m \text{ and } j = 1, \cdots, r\},$$

where polynomials g_i and h_j are given. The question of feasibility of a semi-algebraic set is NP-hard (L. Blum and Smale (1998)). However, if we have a test to verify

$S_\gamma = \emptyset$, we can find γ^* by performing a bisection on γ. In the following section, we use the results of Section 2.1 to provide sufficient conditions, in the form of Linear Matrix Inequalities (LMIs), for $S_\gamma = \emptyset$.

2.3 Algorithms for optimization of polynomials

In this section, we discuss how to find lower bounds on β^* for different classes of polynomial optimization problems. The results in this section are primarily expressed as methods for verifying $S_\gamma = \emptyset$ and can be used with bisection to solve polynomial optimization problems.

2.3.1 Case 1: Optimization over the standard simplex Δ^n

Define the standard unit simplex as

$$\Delta^n := \{x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1, x_i \geq 0\}. \tag{2.15}$$

Consider the polynomial optimization problem

$$\gamma^* = \min_{x \in \Delta^n} \quad f(x),$$

where f is a homogeneous polynomial of degree d. If f is not homogeneous, we can homogenize it by multiplying each monomial $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ in f by $(\sum_{i=1}^n x_i)^{d - \|\alpha\|_1}$. Notice that since $\sum_{i=1}^n x_i = 1$ for all $x \in \Delta^n$, the homogenized f is equal to f for every $x \in \Delta^n$. To find γ^*, one can solve the following optimization of polynomials problem.

$$\begin{aligned} \gamma^* = \max_{\gamma \in \mathbb{R}} \quad & \gamma \\ \text{subject to } & f(x) \geq \gamma \text{ for all } x \in \Delta^n \end{aligned} \tag{2.16}$$

Clearly, Problem (2.16) can be re-stated as the following feasibility problem

$$\gamma^* = \min_{\gamma \in \mathbb{R}} \quad \gamma$$
$$\text{subject to } S_\gamma := \{x \in \mathbb{R}^n : f(x) - \gamma < 0, \sum_{i=1}^n x_i = 1,\, x_i \geq 0\} = \emptyset.$$

For a given γ, we can use the following version of Polya's theorem to verify $S_\gamma = \emptyset$.

Theorem 8. *(Polya's theorem, simplex version) If a homogeneous matrix-valued polynomial F satisfies $F(x) > 0$ for all $x \in \Delta^n := \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1, x_i \geq 0\}$, then there exists $e \geq 0$ such that all the coefficients of*

$$\left(\sum_{i=1}^n x_i\right)^e F(x)$$

are positive definite.

See pages 57-59 of G. Hardy and Polya (1934) for a proof. The converse of the theorem only implies $F \geq 0$ over the unit simplex. Given $\gamma \in \mathbb{R}$, it follows from the converse of Theorem 8 that $S_\gamma = \emptyset$ if there exists some $e \geq 0$ such that

$$\left(\sum_{i=1}^n x_i\right)^e \left(f(x) - \gamma \left(\sum_{i=1}^n x_i\right)^d\right) \tag{2.17}$$

has all positive coefficients, where recall that d is the degree of f. We can compute lower bounds on γ^* by performing a bisection on γ. For each γ of the bisection, if there exists some $e \geq 0$ such that all of the coefficients of (2.17) are positive, then $\gamma \leq \gamma^*$. We have detailed this procedure in Algorithm 1.

In Chapter 4, we will propose a decentralized version of Algorithm 1 to perform robust stability analysis over a simplex.

2.3.2 Case 2: Optimization over the hypercube Φ^n

Given $r_i \in \mathbb{R}$, define the hypercube

$$\Phi^n := \{x \in \mathbb{R}^n : |x_i| \leq r_i, i = 1, \cdots, n\}. \tag{2.18}$$

Input:

Polynomial f; maximum polya's exponent $e_{\max}$; lower-bound γ_l and upper-bound γ_u for bisection search; number of bisection iterations $b_{\max}$;

Initialization:

Set Polya's exponent $e = 0$.

Set $k = 0$.

Main Loop:

while $d \leq b_{\max}$ **do**
  Set $\gamma = \frac{\gamma_u + \gamma_l}{2}$.
  while *Eq.* (2.17) *has some negative coefficient or* $e \leq e_{\max}$ **do**
    Set $e = e + 1$.
    Calculate the Product (2.17).
  end
  if *Eq.* (2.17) *has all positive coefficients* **then**
    Set $\gamma_l = \gamma$.
  else
    Set $\gamma_u = \gamma$.
  end
  Set $k = k + 1$.
end

Output:

γ: a lower bound on the minimum of f over the standard simplex.

Algorithm 1: Polya's algorithm for polynomial optimization over the simplex

Define the set of n-variate *multi-homogeneous* polynomials of degree vector $d \in \mathbb{N}^n$ as

$$\left\{ p \in \mathbb{R}[x, y] : p(x, y) = \sum_{\substack{h,g \in \mathbb{N}^n \\ h+g=d}} p_{h,g} x_1^{h_1} y_1^{g_1} \cdots x_n^{h_n} y_n^{g_n},\ p_{h,g} \in \mathbb{R} \right\}. \tag{2.19}$$

In a more general case, if the coefficients $p_{h,g}$ are matrices, we call p a *matrix-valued multi-homogeneous* polynomial. Now consider the polynomial optimization problem

$$\gamma^* = \min_{x \in \Phi^n} f(x).$$

To find γ^*, one can solve the following feasibility problem.

$$\begin{aligned} \gamma^* = \min_{\gamma \in \mathbb{R}} \quad & \gamma \\ \text{subject to} \quad & S_{\gamma,r} := \{x \in \mathbb{R}^n : f(x) - \gamma < 0,\ |x_i| \leq r_i,\ i = 1, \cdots, n\} = \emptyset \end{aligned} \tag{2.20}$$

For a given γ, we propose the following version of Polya's theorem (Kamyar and Peet (2012b)) to verify $S_{\gamma,r} = \emptyset$.

Theorem 9. *(Polya's theorem: multi-simplex version) A matrix-valued multi-homogeneous polynomial F satisfies $F(x, y) > 0$ for all $(x_i, y_i) \in \Delta^2, i = 1, \cdots, n$, if there exist $e \geq 0$ such that all the coefficients of*

$$\left(\prod_{i=1}^{n} (x_i + y_i)^e \right) F(x, y)$$

are positive definite.

We will prove this result in Section 5.2. The converse of Theorem 9 only implies non-negativity of F over the hypercube. To find lower bounds on γ, we first obtain the multi-homogeneous form p of the polynomial f in (2.20). In 5.2 we have provided a procedure to construct p. Given $\gamma \in \mathbb{R}$ and $r \in \mathbb{R}^n$, it follows from the converse of Theorem 9 that $S_{\gamma,r}$ defined in (2.20) is empty if there exists some $e \geq 0$ such that

$$\left(\prod_{i=1}^{n} (x_i + y_i)^e \right) \left(p(x, y) - \gamma \left(\prod_{i=1}^{n} (x_i + y_i)^{d_i} \right) \right) \tag{2.21}$$

has all positive coefficients, where d_i is the degree of x_i in $p(x, y)$. We can compute lower bounds on γ^*, as defined in (2.20), by performing a bisection on γ. For each γ of the bisection, if there exists some $e \geq 0$ such that all of the coefficients of (2.21) are positive, then $\gamma \leq \gamma^*$. By replacing (2.17) with (2.21) in Algorithm 1, this algorithm computes γ. In Chapter 5, we will propose a parallel algorithm to perform robust stability analysis for systems with uncertain parameters inside a hypercube.

2.3.3 Case 3: Optimization over the convex polytope Γ^K

Given $w_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}$, define the convex polytope

$$\Gamma^K := \{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \cdots, K\}. \tag{2.22}$$

Suppose Γ^K is bounded. Consider the polynomial optimization problem

$$\gamma^* = \min_{x \in \Gamma^K} f(x),$$

where f is a polynomial of degree d_f. To find γ^*, one can solve the feasibility problem

$$\begin{aligned} \gamma^* = \min_{\gamma \in \mathbb{R}} \quad & \gamma \\ \text{subject to} \quad & S_{\gamma,K} := \{x \in \mathbb{R}^n : f(x) - \gamma < 0,\ w_i^T x + u_i \geq 0,\ i = 1, \cdots, K\} = \emptyset. \end{aligned}$$

Given γ, one can use Handelman's theorem to verify $S_{\gamma,K} = \emptyset$.

Theorem 10. *(Handelman's Theorem) Given $w_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}$, define the polytope $\Gamma^K := \{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \cdots, K\}$. If a polynomial $f(x) > 0$ on Γ^K, then there exist $b_\alpha \geq 0$, $\alpha \in \mathbb{N}^K$ such that for some $d \in \mathbb{N}$,*

$$f(x) = \sum_{\substack{\alpha \in \mathbb{N}^K \\ \alpha_1 + \cdots + \alpha_K \leq d}} b_\alpha (w_1^T x + u_1)^{\alpha_1} \cdots (w_K^T x + u_K)^{\alpha_K}. \tag{2.23}$$

Consider the Handelman basis associated with polytope Γ^K defined as

$$H_s := \left\{ \lambda_\alpha \in \mathbb{R}[x] : \lambda_\alpha(x) = \prod_{i=1}^{K} \left(w_i^T x + u_i \right)^{\alpha_i}, \alpha \in \mathbb{N}^K, \sum_{i=1}^{K} \alpha_i \leq s \right\}.$$

Basis H_s spans the space of polynomials of degree s or less, however it is not minimal. As a special case, if we take Γ^K to be the standard unit simplex of $\mathbb{R}^K$, i.e.,

$$\Gamma^K := \{x \in \mathbb{R}^K : 1 - \sum_{i=1}^{K} x_i \geq 0, x_i \geq 0 \text{ for } i = 1, \cdots, K\},$$

then the following set of polynomials is called the *Bernstein basis* associated with Γ^K.

$$B_s := \left\{ \lambda_\alpha \in \mathbb{R}[x] : \lambda_\alpha(x) = \frac{s!}{\alpha_1! \cdots \alpha_{K+1}!} \left(\prod_{i=1}^{K} x_i^{\alpha_i} \right) \left(1 - \sum_{i=1}^{K} x_i \right)^{\alpha_{K+1}}, \alpha \in \mathbb{N}^{K+1}, \sum_{i=1}^{K} \alpha_i = s \right\}.$$

Unlike H_s, B_s is a minimal basis[2] for the vector space of polynomials of degree $\leq s$.

Given $\gamma \in \mathbb{R}$, polynomial $f(x)$ of degree d_f and $d_{\max} \in \mathbb{N}$, if there exist

$$c_\alpha \geq 0 \text{ for all } \alpha \in I_d := \{\alpha \in \mathbb{N}^K : \|\alpha\|_1 \leq d\} \tag{2.24}$$

such that

$$f(x) - \gamma = \sum_{\alpha \in I_d} c_\alpha \prod_{i=1}^{K} (w_i^T x + u_i)^{\alpha_i} \tag{2.25}$$

for some $d \geq d_f$, then $f(x) - \gamma \geq 0$ for all $x \in \Gamma^K$. Thus $S_{\gamma,K} = \emptyset$. Since H_s is not a minimal basis, if (2.24) is feasible, then c_α are not unique. Feasibility of Conditions (2.24) and (2.25) can be determined using linear programming. To set-up the linear program, we first represent the right and left hand side of (2.25) in the canonical basis as

$$f(x) - \gamma = \begin{bmatrix} b_1 & b_2 & \cdots & b_M \end{bmatrix} z_{n,d}(x) \tag{2.26}$$

$$\sum_{\alpha \in I_d} c_\alpha \prod_{i=1}^{K} (w_i^T x + u_i)^{\alpha_i} = \begin{bmatrix} l_1(c_\alpha, w, u) & l_2(c_\alpha, w, u) & \cdots & l_M(c_\alpha, w, u) \end{bmatrix} z_{n,d}(x), \tag{2.27}$$

where it can be shown that $l_i : \mathbb{R}^N \times \mathbb{R}^{n \times K} \times \mathbb{R}^K \to \mathbb{R}$ are *affine* in c_α and $N := \binom{K+d}{d}$ is the cardinality of the index set $\{\alpha \in \mathbb{N}^K : \|\alpha\|_1 \leq d\}$. In (2.27),

$$w := \begin{bmatrix} w_1 & w_2 & \cdots & w_K \end{bmatrix} \text{ and } u := \begin{bmatrix} u_1 & u_2 & \cdots & u_K \end{bmatrix}^T,$$

[2]This follows from the fact that every polynomial can be uniquely represented in the canonical basis and every member of the canonical basis is a unique linear combination of $\lambda_\alpha \in B_s$. A derivation for these linear combinations can be found in Farin (2002)

where $w_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}$ define the polytope Γ^K in (2.22). Recall that in (2.27), $z_{n,d}(x)$ denotes the vector of all $n-$variate monomials of degree d or less. By equating (2.26) and (2.27) and cancelling $z_{n,d}(x)$ from both sides, the problem of finding a lower bound γ_d on γ^* can be expressed as the following linear program.

$$\begin{aligned} \gamma_d := \max_{\gamma \in \mathbb{R}, c_\alpha \geq 0} \quad & \gamma \\ \text{subject to} \quad & l_i(c_\alpha, w, u) = b_i \qquad \text{for } \; i = 1, \cdots, M. \end{aligned} \tag{2.28}$$

If Linear Program (2.28) is infeasible for some d, then one can increase d and repeat setting-up and solving Linear Program (2.28). From Handelman's theorem, if $f(x) - \gamma > 0$ for all $x \in \Gamma^K$, then for some $d \geq d_f$, Conditions (2.24) and (2.25) hold and Linear Program (2.28) will have a solution. We have outlined this procedure in Algorithm 2. Unfortunately, to this date all the proposed upper-bounds on d (see e.g., Powers and Reznick (2001) and Leroy (2012)) are functions of the minimum of $f(x) - \gamma$ over the polytope Γ^K. In Chapter 6, we will combine this algorithm with a polytope decomposition scheme to construct Lyapunov functions for nonlinear systems with polynomial vector fields.

2.3.4 *Case 4: Optimization over compact semi-algebraic sets*

Recall that we defined a semi-algebraic set as

$$S := \{x \in \mathbb{R}^n : g_i(x) \geq 0, i = 1, \cdots, m, \; h_j(x) = 0, j = 1, \cdots, r\}. \tag{2.29}$$

Suppose S is bounded. Consider the polynomial optimization problem

$$\begin{aligned} \gamma^* = \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \geq 0 \text{ for } i = 1, \cdots, m \\ & h_j(x) = 0 \text{ for } j = 1, \cdots, r. \end{aligned}$$

Input: Polynomial f of degree d_f; maximum degree $d_{\max}$ for Handelman's basis; polytope data: $w_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}$ in (2.22); tolerance $\epsilon > 0$ for stopping criterion.

Initialization:

Set $d = d_f$.

Set $\gamma_{\text{old}} = -10^{100}$.

Set $\gamma_{\text{new}} = -10^{100} + 2\epsilon$.

Main Loop:

while $(d < d_{\max})$ *and* $(\gamma_{new} - \gamma_{old} < \epsilon)$ **do**

Express $f(x) - \gamma_{\text{new}}$ in the canonical basis as in (2.26).

Express $\sum_{\alpha \in I_d} c_\alpha \prod_{i=1}^{K} (w_i^T x + u_i)^{\alpha_i}$ in the canonical basis as in (2.27).

Set-up LP (2.28).

if *LP* (2.28) *is feasible* **then**

Set $\gamma_{\text{old}} = \gamma_{\text{new}}$.

Set $\gamma_{\text{new}} = \gamma_d$.

end

Set $d = d + 1$.

end

Output: γ_{new}: a lower bound on the minimum of f over polytope Γ^K.

Algorithm 2: Polynomial optimization over convex polytopes using Handelman's theorem

Define the following cone of polynomials which are positive over S.

$$M_{g,h} := \left\{ m \in \mathbb{R}[x] : m(x) - \sum_{i=1}^{m} s_i(x) g_i(x) - \sum_{i=1}^{r} t_i(x) h_i(x) \text{ is SOS}, s_i \in \Sigma_{2d}, t_i \in \mathbb{R}[x] \right\}, \tag{2.30}$$

where Σ_{2d} denotes the cone of SOS polynomials of degree $2d$. From Putinar's Positivstellensatz (Theorem 6) it follows that if the Cone (2.30) is Archimedean, then the solution to the following SOS program is a lower bound on γ^*. Given $d \in \mathbb{N}$, define

$$\begin{aligned} \gamma^d := & \max_{\gamma \in \mathbb{R}, s_i, t_i} \quad \gamma \\ & \text{subject to} \;\; f(x) - \gamma - \sum_{i=1}^{m} s_i(x) g_i(x) - \sum_{i=1}^{r} t_i(x) h_i(x) \text{ is SOS }, \; t_i \in \mathbb{R}[x], s_i \in \Sigma_{2d}. \end{aligned} \tag{2.31}$$

On the other hand, every $F \in \Sigma_{2d}$ has a quadratic representation with a positive semi-definite matrix. To see this, suppose $F(x) = \sum_i q_i(x)^2$, where q_i are polynomials of degree d. Each q_i can be written in the canonical basis as $q_i(x) = c_i^T z_{n,d}(x)$, where $z_{n,d}(x)$ is the vector of all $n-$variate monomials of degree d or less. Hence, we can write F as

$$\begin{aligned} F(x) &= \sum_i q_i(x)^2 = \sum_i z_{n,d}(x)^T c_i c_i^T z_{n,d}(x) \\ &= z_{n,d}(x)^T \left(\sum_i c_i c_i^T \right) z_{n,d}(x) = z_{n,d}(x)^T Q z_{n,d}(x), \end{aligned}$$

where clearly $Q \geq 0$. Therefore, for given $\gamma \in \mathbb{R}$ and $d \in \mathbb{N}$, Problem (2.31) can be formulated as the following linear matrix inequality.

$$\begin{aligned} & \text{Find} \quad Q_i \geq 0 \text{ and } P_j \;\text{ for } i = 0, \cdots, m \text{ and } j = 1, \cdots, r \\ & \text{such that} \;\; f(x) - \gamma = z_{n,d}^T(x) \left(Q_0 + \sum_{i=1}^{m} Q_i g_i(x) + \sum_{j=1}^{r} P_j h_j(x) \right) z_{n,d}(x), \end{aligned} \tag{2.32}$$

where Q_i and $P_j \in \mathbb{S}^N$, where $\mathbb{S}^N$ is the subspace of symmetric matrices in $\mathbb{R}^{N \times N}$ and $N := \begin{pmatrix} n+d \\ d \end{pmatrix}$. See G. Blekherman and Thomas (2013) for methods of solving SOS programs. Also Papachristodoulou *et al.* (2013) provide a MATLAB package called SOSTOOLs for solving SOS programs.

If the Cone (2.30) is not Archimedean, then we can use Schmudgen's Positivstellensatz to obtain the following SOS program with solution $\gamma^d \leq \gamma^*$.

$$\begin{aligned} \gamma^d = & \max_{\gamma \in \mathbb{R}, s_i \in \Sigma_{2d}, t_i \in \mathbb{R}[x]} \quad \gamma \\ & \text{subject to} \quad f(x) - \gamma = 1 + \sum_{\lambda \in \{0,1\}^m} s_\lambda(x) g_1(x)^{\lambda_1} \cdots g_m(x)^{\lambda_m} + \sum_{i=1}^{r} t_i(x) h_i(x). \end{aligned} \tag{2.33}$$

The Positivstellensatz and SOS programming can also be applied to polynomial optimization over a more general form of semi-algebraic sets defined as

$$T := \{x \in \mathbb{R}^n : g_i(x) \geq 0,\ i = 1, \cdots, m, h_j(x) = 0, j = 1, \cdots, r, q_k(x) \neq 0, k = 1, \cdots, l\}.$$

It can be shown that $T = \emptyset$ if and only if

$$\begin{aligned} \hat{T} := \{(x, y) \in \mathbb{R}^{n+l} : g_i(x) \geq 0,\ i = 1, \cdots, m,\ h_j(x) = 0,\ j = 1, \cdots, r, \\ y_k q_k(x) = 1,\ k = 1, \cdots, l\} = \emptyset. \end{aligned}$$

Thus, for any $f \in \mathbb{R}[x]$, we have

$$\min_{x \in T} f(x) = \min_{(x,y) \in \hat{T}} f(x).$$

Therefore, to find lower bounds on $\min_{x \in T} f(x)$, one can apply SOS programming and Putinar's Positivstellensatzs to $\min\limits_{(x,y) \in \hat{T}} f(x)$.

2.3.5 Case 5: Tests for non-negativity on $\mathbb{R}^n$:

The following result from Habicht (1939) defines a test for non-negativity of homogeneous polynomials over $\mathbb{R}^n$.

Theorem 11. *(Habicht theorem) For every homogeneous polynomial f that satisfies $f(x_1, \cdots, x_n) > 0$ for all $x \in \mathbb{R}^n \setminus \{0\}$, there exists some $e \geq 0$ such that all of the coefficients of*

$$\left(\sum_{i=1}^{n} x_i^2\right)^e f(x_1, \cdots, x_n) \tag{2.34}$$

are positive. In particular, the product is a sum of squares of monomials.

Using this theorem, one can verify non-negativity of any homogeneous polynomial f over $\mathbb{R}^n$ by multiplying F repeatedly by $\sum_{i=1}^{n} x_i^2$. If for some $e \in \mathbb{N}$, the Product (2.34) has all positive coefficients, then $f \geq 0$. We can define an alternative test for non-negativity over $\mathbb{R}^n$ using the following theorem (de Loera and Santos (1996)).

Theorem 12. *Define $E_n := \{-1, 1\}^n$. Suppose a polynomial $f(x_1, \cdots, x_n)$ of degree d satisfies $f(x_1, \cdots, x_n) > 0$ for all $x \in \mathbb{R}^n$ and its homogenization[3] is positive definite. Then*

1. *there exist $\lambda_e \geq 0$ and coefficients $c_\alpha \in \mathbb{R}$ such that*

$$\left(1 + e^T x\right)^{\lambda_e} f(x_1, \cdots, x_n) = \sum_{\alpha \in I_e} c_\alpha x_1^{\alpha_1} \cdots x_n^{\alpha_n} \text{ for all } e \in E_n, \tag{2.35}$$

   where $I_e := \{\alpha \in \mathbb{N}^n : \|\alpha\|_1 \leq d + \lambda_e\}$ and $sgn(c_\alpha) = e_1^{\alpha_1} \cdots e_n^{\alpha_n}$.

2. *there exist positive $N, D \in \mathbb{R}[x_1^2, \cdots, x_n^2, f^2]$ such that $f = \frac{N}{D}$.*

Based on the converse of Theorem 12, we propose the following test for non-negativity of polynomials over the cone $\Lambda_e := \{x \in \mathbb{R}^n : sgn(x_i) = e_i, i = 1, \cdots, n\}$ for some $e \in E_n$. Multiply a given polynomial f repeatedly by $1 + e^T x$ for some $e \in E_n$. If there exists some $\lambda_e \geq 0$ such that $sgn(c_\alpha) = e_1^{\alpha_1} \cdots e_n^{\alpha_n}$, then (2.35)

[3] Associated to every polynomial $f(x_1, \cdots, x_n)$, $x \in \mathbb{R}^n$ of degree d, there exists a degree e homogeneous polynomial $h(x_1, \cdots, x_n, y) := y^e f(\frac{x_1}{y}, \cdots, \frac{x_n}{y})$, where $e \geq d$.

clearly implies that $f(x) \geq 0$ for all $x \in \Lambda_e$. Since $\mathbb{R}^n = \cup_{e \in E_n} \Lambda_e$, we can repeat the test 2^n times to obtain a test for non-negativity of f over $\mathbb{R}^n$.

The second part of Theorem 12 gives a solution to Hilbert's 17^{th} problem (see Section 2.1). For a construction of this solution (i.e., numerator N and denominator D) see de Loera and Santos (1996).

Chapter 3

SEMI-DEFINITE PROGRAMMING AND INTERIOR-POINT ALGORITHMS

As discussed in Chapter 2, Polya's theorem, Handelman's theorem and the Positivstellensatz results can be used to approximate the minimum of a polynomial over simplicies, hypercubes, polytopes and semi-algebraic sets. We showed that these theorems define sequences of Linear/Semi-Definite Programs (SDPs) whose solutions define lower bounds on the objective of the polynomial optimization problem. In this section, we focus on solving these SDPs. In particular, we discuss the primal and dual forms of semi-definite programming problems and introduce a state-of-the-art primal-dual interior-point algorithm for solving SDPs. In Section 4.5, we will propose a new parallel version of this algorithm - an algorithm which is specifically designed to solve the SDPs defined by applying Polya's theorem to optimization of polynomials arising in robust stability and control problems.

3.1 Convex optimization and duality

Let us define the constrained optimization problem

$$\begin{aligned} f^* := \min_{x \in \mathbb{R}^n} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \cdots, p \\ & h_i(x) = 0, \quad i = 1, \cdots, q, \end{aligned} \tag{3.1}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ and $h_i : \mathbb{R}^n \to \mathbb{R}$. For every problem of Form (3.1), one can define the *Lagrangian function* $L : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R}^q$ as

$$L(x, \lambda, \nu) := f_0(x) + \sum_{i=1}^{p} \lambda_i f_i(x) + \sum_{i=1}^{q} \nu_i h_i(x), \tag{3.2}$$

where $\lambda_i \in \mathbb{R}$ and $\nu_i \in \mathbb{R}$ are called the *Lagrange multipliers* associated with the inequality constraints and the equality constraints in (3.1), respectively. The vectors $\lambda = [\lambda_1, \cdots, \lambda_p]$ and $\nu = [\nu_1, \cdots, \nu_q]$ are called the *dual variables* of Problem (3.1). Let us define the *Lagrange dual function* $g : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ as

$$g(\lambda, \nu) := \inf_x \left(f_0(x) + \sum_{i=1}^{p} \lambda_i f_i(x) + \sum_{i=1}^{q} \nu_i h_i(x) \right).$$

The Lagrange dual functions have some interesting properties. First, because the Lagrangian is affine in λ_i and ν_i and the pointwise infimum of a family of affine functions is concave (Boyd and Vandenberghe (2004)), g is a concave function. Second, it is easy to show that the dual functions yield lower bounds on f^* as define in (3.1), i.e., $g(\lambda, \nu) \leq p^*$. To find the best lower bound on f^* using the Lagrange dual function, one can solve the *Lagrange dual problem* defined as

$$\begin{aligned} d^* := \max_{\lambda,\nu} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0. \end{aligned} \tag{3.3}$$

Every pair (λ, ν) which satisfies $\lambda \geq 0$ and $g(\lambda, \nu) > -\infty$ is called a *dual feasible* point for Problem (3.3). Likewise, every $x \in \mathbb{R}^n$ satisfying $f_i(x) \leq 0$ for $i = 1, \cdots, p$ and $h_i(x) \leq 0$ for $i = 1, \cdots, q$ is a *primal feasible* point for Problem (3.1). Dual feasible points can be used to bound sub-optimality of a primal feasible point. In particular, for every primal feasible point x and dual feasible point (λ, ν),

$$f_0(x) - f^* \leq f_0(x) - g(\lambda, \nu),$$

where $f_0(x) - g(\lambda, \nu)$ is called the *duality gap* associated with x and (λ, ν). For certain problems, the duality gap associated with primal optimal point x^* and dual optimal point (λ^*, ν^*) is zero, i.e.,

$$f_0(x^*) = f^* = d^* = g(\lambda^*, \nu^*).$$

This property is often called *strong duality*. One important class of problems which usually posses this property is convex optimization problems. A convex optimization problem is an optimization problem of Form (3.1), where the functions f_i, $i = 0, \cdots, p$ are convex[1] and h_i, $i = 1, \cdots, q$ are affine. For example, the Lagrange dual problem (3.3) is by definition a convex problem (it is a maximization of a concave function) whether or not its primal (Eq. (3.1)) is convex. It can be shown that (Slater (2014)) if the primal problem (3.1) is convex and there exists some $x \in \mathbb{R}^n$ such that

$$\begin{aligned} f_i(x) < 0 &\quad \text{for } i = 1, \cdots, p \quad \text{and} \\ h_i(x) = 0 &\quad \text{for } i = 1, \cdots, q, \end{aligned} \tag{3.4}$$

then strong duality holds. Strong duality can be exploited to solve the primal problem via its dual. This is useful specially when the dual is easier or computationally less expensive to solve. Suppose a dual optimal solution (λ^*, ν^*) is known and strong duality holds. If

$$x^* := \arg\min L(x, \lambda^*, \nu^*)$$

is unique and primal feasible, x^* is the primal optimal solution.

3.2 Descent algorithms for convex optimization

Suppose $f_0 : \mathbb{R}^n \to \mathbb{R}$ is differentiable. For $\hat{x}$ to be a minimum of f_0, the necessary condition is that $[\nabla_x f_0(x)]_{x=\hat{x}} = 0$. The Karush-Kuhn-Tucker (KKT) conditions (Kuhn *et al.* (1951)) generalize this necessary condition for the constrained optimization problem (3.1), under the assumption that the functions f_i and g_i are differentiable. The KKT conditions can be stated as follows: Suppose $x^* \in \mathbb{R}^n$ is a primal optimal point for (3.1) and λ_i^*, $i = 1, \cdots, p$ and ν_i^*, $i = 1, \cdots, q$ are dual optimal

[1] A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y)$ for all $x, y \in \mathbb{R}^n$ and for all $\alpha, \beta \in \mathbb{R}$ such that $\alpha + \beta = 1$ and $\alpha, \beta \geq 0$.

points for (3.3). Moreover, suppose the strong duality holds, i.e., $f_0(x^*) = g(\lambda^*, \nu^*)$. Then, the optimal primal and dual points satisfy the following.

$$\begin{aligned}
\left[\nabla_x f_0(x) + \sum_{i=1}^{p} \lambda_i^* \nabla_x f_i(x) + \sum_{i=1}^{q} \nu_i^* \nabla_x h_i(x)\right]_{x=x^*} &= 0 \\
f_i(x^*) &\leq 0 \qquad \text{for } i = 1, \cdots, p \\
h_i(x^*) &= 0 \qquad \text{for } i = 1, \cdots, q \\
\lambda_i^* &\geq 0 \qquad \text{for } i = 1, \cdots, p \\
\lambda_i^* f_i(x^*) &= 0 \qquad \text{for } i = 1, \cdots, p. \qquad (3.5)
\end{aligned}$$

The first line follows from the fact that x^* is a minimizer of the Lagrangian $L(x, \lambda^*, \nu^*)$. The second, third and fourth lines indicate that x^* and (λ^*, ν^*) are primal and dual feasible. The last line is called the *complementary slackness* and follows from strong duality. This condition implies that for $i = 1, \cdots, p$, either the i^{th} primal constraint must be active at x^* (i.e., $f_i(x^*) = 0$) or its corresponding optimal dual variable λ_i^* must be zero.

In general, the KKT conditions are only necessary conditions for optimality. Indeed, under certain regularity conditions, local minima of the primal Problem (3.1) satisfy the KKT conditions. However, when the primal problem is convex and there exists $x \in \mathbb{R}^n$ which satisfies (3.4), the KKT conditions become necessary and sufficient. Motivated by this result, many of the existing convex optimization algorithms are in principle algorithms for solving the KKT conditions iteratively. These algorithms are often called *descent algorithms* because they generate a sequence $\{x^k\}_{k=1,2,\cdots}$ of primal feasible solutions which satisfy

$$f_0(x^k) > f_0(x^{k+1}) \quad \text{for } k = 1, 2, 3, \cdots, \qquad (3.6)$$

unless x^k is optimal. One example of descent algorithms is the Newton's algorithm. Given a primal feasible starting point x^0, Newton's algorithm finds a sequence of *search directions* $\Delta x^k \in \mathbb{R}^n$ and *step length* $t^k \in \mathbb{R}^+$ such that all the iterates

$$x^{k+1} = x^k + t^k \Delta x^k \quad k = 1, 2, 3, \cdots$$

are feasible and satisfy (3.6). Given a primal feasible point x^k, Newton's algorithm calculates the search directions Δx^k by first defining the convex optimization problem

$$\begin{aligned} \hat{f}_k := \min_{v \in \mathbb{R}} \quad & \left[f_0(x) + \nabla_x f_0(x)^T v + \frac{1}{2} v^T \nabla_x^2 f_0(x) v \right]_{x=x^k} \\ \text{subject to} \quad & h_i(x^k + v) = 0 \qquad \text{for } i = 1, \cdots, m. \end{aligned} \tag{3.7}$$

The objective function of Problem (3.7) is the second-order Taylor's approximation of the objective function $f_0(x)$ at $x = x^k$. Then, the KKT optimality conditions for Problem (3.7) yield the following system of linear equations.

$$\begin{bmatrix} [\nabla_x^2 f_0(x)]_{x=x^k} & D_x h(x)^T \\ D_x h(x) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \nu^k \end{bmatrix} = \begin{bmatrix} -[\nabla_x f_0(x)]_{x=x^k} \\ \mathbf{0} \end{bmatrix}, \tag{3.8}$$

where $D_x h(x) := [\nabla_x h_1(x), \cdots, \nabla_x h_m(x)]^T$. If the coefficient matrix in (3.8) is non-singular, then there exist a unique Newton's search direction Δx^k and optimal dual point ν^k for the dual to Problem (3.8). Finally, Newton's algorithm calculates the new iterate as $x^{k+1} = x^k + t^k \Delta x^k$, where a step length t^k can be obtained using a line search method such as *backtracking* (Dennis Jr and Schnabel (1996)) or bisection. A typical stopping criterion for Newton's algorithm is $f_0(x^k) - \hat{f}^k \leq \epsilon$ for some desired $\epsilon > 0$, where recall that $\hat{f}^k$ is the minimum of the second-order Taylor's approximation of f_0 at x^k, subject to the equality constraints in (3.7). The difference between $f_0(x^k)$ and $\hat{f}^k$ can also be interpreted as the size of Newton's search direction defined by the following weighted norm of Δx^k:

$$f_0(x^k) - \hat{f}^k = \|\Delta x^k\|_{\nabla_x^2 f_0(x)|_{x=x^k}} := 2(\Delta x^k)^T \left. \nabla_x^2 f_0(x) \right|_{x=x^k} \Delta x^k. \tag{3.9}$$

For a comprehensive discussion on the complexity and convergence of Newton's algorithm, refer to Boyd and Vandenberghe (2004).

3.3 Interior-point algorithms for convex problems with inequality constraints

Suppose in Problem (3.1), f_i are convex and differentiable and h_i are affine and differentiable. One of the most successful class of algorithms for solving this type of problems is *interior-point* algorithms. Typically, interior-point algorithms solve this problem in two steps: 1- Reducing the problem to a sequence of convex optimization programs with only linear equality constraints; and 2- Applying a descent algorithm, e.g., Newton's algorithm, to solve the equality constrained problem. One way to define this sequence of equality constrained problems is to incorporate the inequality constraints into the objective function using *barrier functions.* For example, by using logarithmic barrier functions one can approximate Problem (3.1) as

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \quad f_0(x) - \sum_{i=1}^{p} \left(\frac{1}{b}\right) \log(-f_i(x)) \\ &\text{subject to} \quad h_i(x) = 0, \quad i = 1, \cdots, q. \end{aligned} \tag{3.10}$$

for some $b > 0$. Clearly, if any of the inequality constraints becomes active ($f_i(x) \to 0$), then the objective function blows up. Thus, any solution to Problem (3.10) lies in the interior (as the name 'interior-point' suggests) of the feasible set

$$\{x \in \mathbb{R}^n : f_i(x) \le 0, h_j(x) = 0, \text{ for } i = 1, \cdots, p \text{ and } j = 1, \cdots, q\}.$$

Since Problem (3.10) is convex, one can use Newton's algorithm to find the optimal solution x_b^* for any $b > 0$. In particular, given $b > 0$ and feasible x^0, Newton's algorithm finds a sequence $\{x^k\}_{k=1,2,\cdots} \to x_b^*$ by solving the modified KKT conditions

$$\begin{bmatrix} \left[b\nabla_x^2 f_0(x) - \nabla_x^2 \sum_{i=1}^{p} \log(-f_i(x)) \right]_{x=x^k} & D_x h(x)^T \\ D_x h(x) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \nu^k \end{bmatrix} = \begin{bmatrix} -\left[\nabla_x f_0(x) + \nabla \sum_{i=1}^{p} \log(-f_i(x)) \right]_{x=x^k} \\ \mathbf{0} \end{bmatrix} \tag{3.11}$$

for Δx^k and ν^k and setting $x^{k+1} = x^k + t^k x^k$. The set of optimal solutions x_b^* for all $b > 0$ is called the *central path*. Corresponding to any x_b^* in the central path, one can verify that

$$\lambda_i^* = -\frac{1}{b\, f_i(x_b^*)} \quad \text{for } i = 1, \cdots, p \quad \text{ and } \quad \nu^* = \frac{\nu^k}{b} \in \mathbb{R}^q,$$

are dual feasible and together with x_b^* yield the duality gap $\frac{p}{b}$. This indicates that as $b \to \infty$, x_b^* converges to the optimal solution of Problem 3.1 under the assumption that f_i are convex and differentiable and h_i are affine and differentiable. Based on this result, we can summarize the interior-point barrier algorithm for inequality constrained problems in Algorithm 3.

An alternative subclass of interior-point algorithms for solving inequality constrained problems is the *primal-dual* algorithms. Similar to the barrier algorithm, primal-dual algorithms find their search direction by solving the KKT optimality conditions. However, instead of incorporating the inequality constraints into the objective function (equivalently, eliminating the dual variable λ from the KKT condition (3.5)), primal-dual algorithms simultaneously solve the primal problem and its dual by computing independent Newton's search directions $\Delta x \in \mathbb{R}^n$, $\Delta\lambda \in \mathbb{R}^p$ and $\Delta\nu \in \mathbb{R}^q$ for primal and dual variables x, λ and ν. Given a feasible point (x^k, λ^k, ν^k) for Problem (3.1) and $b > 0$, the basic version of primal-dual algorithms computes the

Input: Convex functions $f_0, \cdots, f_p$; affine functions $h_1, \cdots, h_q$; a feasible starting point x^0; initial barrier parameter b_0; tolerances $\epsilon_b > 0$ and $\epsilon_N > 0$ for stopping criteria.

Initialization:

Set $b = b_0$.

Choose $\mu > 1$.

Barrier Algorithm:

while $\dfrac{p}{b} > \epsilon_b$ **do**

  Set $x = x^0$.

  Set $\Delta x = 10^{100} \cdot \mathbf{1}_n$.

  Newton's Algorithm:

  while $\|\Delta x\|$ *as defined in* (3.9) *is greater than or equal to* ϵ_N **do**

    Calculate Newton's search direction Δx by solving the system of linear equations in (3.11).

    Choose step length t using backtracking line search.

    Update Newton's iterate as $x := x + t\,\Delta x$.

  end

  Set $x_b^* = x$.

  Update the barrier parameter as $b := \mu b$.

end

Output: x_b^*: A $\frac{p}{b}$-suboptimal solution to Problem (3.1).

Algorithm 3: Barrier algorithm for inequality constrained convex optimization problems search directions $(\Delta x^k, \Delta\lambda^k, \Delta\nu^k)$ by approximating the modified KKT conditions

$$R(x,\lambda,\nu,b) = \begin{bmatrix} \nabla f_0(x) + D_x f(x)^T\lambda + D_x h(x)^T\nu \\ \lambda_1 f_1(x) - \dfrac{1}{b} \\ \vdots \\ \lambda_p f_p(x) - \dfrac{1}{b} \\ h_1(x) \\ \vdots \\ h_q(x) \end{bmatrix} = 0$$

at the point (x^k, λ^k, ν^k) as

$$\begin{aligned} &R(x^k + \Delta x^k, \lambda^k + \Delta\lambda^k, \nu^k + \nabla\nu^k, b) \\ &\approx R(x,\lambda,\nu,b) + \begin{bmatrix} \left[\nabla R_1(x,\lambda,\nu,b)^T\right]_{\substack{x=x^k\\ \lambda=\lambda^k\\ \nu=\nu^k}} \\ \vdots \\ \left[\nabla R_{n+p+q}(x,\lambda,\nu,b)^T\right]_{\substack{x=x^k\\ \lambda=\lambda^k\\ \nu=\nu^k}} \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta\lambda^k \\ \Delta\nu^k \end{bmatrix} = 0, \end{aligned} \tag{3.12}$$

and solving for $(\Delta x^k, \Delta\lambda^k, \Delta\nu^k)$. The primal-dual iterates are then updated according to

$$x^{k+1} = x^k + t^k\Delta x^k, \quad \lambda^{k+1} = \lambda^k + t^k\Delta\lambda^k, \quad \nu^{k+1} = \nu^k + t^k\Delta\nu^k.$$

Similar to the barrier algorithm, the duality gap corresponding to any feasible primal-dual iterate (x^k, λ^k, ν^k) is $\dfrac{p}{b}$. Thus, as $b \to \infty$ in (3.12), the resulting iterates converge to the optimal solution of Problem (3.1), assuming that f_i are convex and h_i are affine. In the sequel, we describe a primal-dual algorithm for solving semi-definite programs - a class of convex optimization problems which has numerous applications in control theory.

3.4 Semi-definite programming

Consider the delay-differential equation

$$\dot{x}(t) = Ax(t) + \sum_{i=1}^{N} A_i x(t - \tau_i) \tag{3.13}$$

where $x(t) \in \mathbb{R}^n$ and $\tau_i > 0$, $i = 1, \cdots, N$. From Repin (1965), a sufficient condition for asymptotic stability of this system is existence of $P_0 > 0, \cdots, P_N > 0$ such that the quadratic functional

$$V(x,t) = x^T(t)P_0x(t) + \sum_{i=1}^{N} \int_0^{\tau_i} x(t-s)^T P_i\, x(t-s)ds$$

satisfies $\dot{V}(x,t) < 0$ for all $x(t) \in \mathbb{R}^n \setminus \{0\}$ and $t > 0$. The derivative $\dot{V}(x,t)$ can be expanded as

$$\begin{aligned}\dot{V}(x,t) = x(t)^T &\left(A^T P_0 + P_0 A + \sum_{i=1}^{N} P_i \right) x(t) + x(t)^T \left(\sum_{i=1}^{N} P_0 A_i x(t-\tau_i) \right) \\ &+ \left(\sum_{i=1}^{N} x(t-\tau_i)^T A_i^T P_0 \right) x(t) - \sum_{i=1}^{N} x(t-\tau_i)^T P_i x(t-\tau_i).\end{aligned}$$

Thus, $\dot{V}(x,t) = z(t)^T Q(P_0, \cdots, P_N) z(t)$, where $z(t) := [x(t) \;\; x(t-\tau_1) \;\; \cdots \;\; x(t-\tau_N)]^T$ and

$$Q(P_0, \cdots, P_N) := \begin{bmatrix} A^T P_0 + P_0 A + \sum_{i=1}^{N} P_i & P_0 A_1 & \cdots & P_0 A_N \\ A_1^T P_0 & -P_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A_N^T P_0 & 0 & \cdots & -P_N \end{bmatrix}.$$

Thus, stability of System (3.13) can be verified by solving the following feasibility problem:

$$\begin{aligned} &\text{Find} \quad P_0 > 0, \cdots, P_N > 0 \\ &\text{such that} \;\; Q(P_0, \cdots, P_N) < 0. \end{aligned} \tag{3.14}$$

Now let us parameterize each P_i as

$$P_i(y_{iL+1}, \cdots, y_{(i+1)L}) = \begin{bmatrix} y_{iL+1} & y_{iL+2} & \cdots & y_{iL+n} \\ y_{iL+2} & y_{iL+n+1} & \cdots & y_{iL+2n-1} \\ \vdots & & \ddots & \vdots \\ y_{iL+n} & y_{iL+2n-1} & \cdots & y_{(i+1)L} \end{bmatrix}$$

for $i = 0, \cdots, N$, where $L := n(n+1)/2$ and $y_j \in \mathbb{R}$ for $j = 0, \cdots, (N+1)L$. Then, we can formulate the problem of stability of System (3.13) as the convex optimization problem

$$\begin{aligned} \min_{\substack{y \in \mathbb{R}^{(N+1)L} \\ Z \in \mathbb{S}^{(N+2)n}}} \quad & \langle \mathbf{1}_{(N+1)L}, y \rangle \\ \text{subject to} \quad & \sum_{i=1}^{(N+1)L} F_i y_i = Z \\ & Z \geq 0, \end{aligned} \tag{3.15}$$

where the matrices $F_i \in \mathbb{S}^{(N+2)n}$ for $i = 1, \cdots, (N+1)L$ are defined as

$$F_i = \operatorname{diag}\{P_0(x_0, \cdots, x_L), Q(P_0(x_0, \cdots, x_L), \cdots, P_N(x_{NL+1}, \cdots, x_{(N+1)L}))\}, \tag{3.16}$$

where

$$x_j = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} \qquad \text{for } j = 1, \cdots, (N+1)L.$$

Problem (3.15) is an example of the dual form of the Semi-Definite Programming (SDP) problem. We define SDP as the optimization of a linear objective function over the cone of positive definite matrices subject to linear matrix equality and linear matrix inequality constraints. Given $C \in \mathbb{S}^n, B_i \in \mathbb{S}^n$ for $i = 1, \cdots, k$, $G_i \in \mathbb{S}^n$ for

$i = 1, \cdots, l$, $a \in \mathbb{R}^k$ and $b \in \mathbb{R}^l$, the *primal* SDP problem is

$$\begin{aligned} p^* := \max_{X \in \mathbb{S}^n} \quad & \operatorname{tr}(CX) \\ \text{subject to} \quad & B(X) = a \\ & G(X) \leq b \\ & X \geq 0, \end{aligned} \tag{3.17}$$

where the linear maps $B : \mathbb{S}^n \to \mathbb{R}^k$ and $G : \mathbb{S}^n \to \mathbb{R}^l$ are defined as

$$B(X) = \begin{bmatrix} \operatorname{tr}(B_1 X) \\ \operatorname{tr}(B_2 X) \\ \vdots \\ \operatorname{tr}(B_k X) \end{bmatrix} \qquad \text{and} \qquad G(X) = \begin{bmatrix} \operatorname{tr}(G_1 X) \\ \operatorname{tr}(G_2 X) \\ \vdots \\ \operatorname{tr}(G_l X) \end{bmatrix}. \tag{3.18}$$

To derive the dual SDP to Problem (3.17), we employ Lagrange multipliers $t \in \mathbb{R}^l_+$ and $y \in \mathbb{R}^k$ as follows.

$$p^* = \max_{X \geq 0} \min_{y \in \mathbb{R}^k, t \in \mathbb{R}^l_+} \quad \operatorname{tr}(CX) + t^T(b - G(X)) + y^T(a - B(x))$$

Then, from the *min-max inequality*, i.e.,

$$\max_{u \in U} \min_{v \in V} \; f(u, v) \leq \min_{v \in V} \max_{u \in U} \; f(u, v)$$

it follows that

$$\begin{aligned} p^* &\leq \max_{y \in \mathbb{R}^k, t \in \mathbb{R}^l_+} \min_{X \geq 0} \operatorname{tr}(CX) + t^T(b - G(X)) + y^T(a - B(x)) \\ &= \min_{y \in \mathbb{R}^k, t \in \mathbb{R}^l_+} \max_{X \geq 0} \operatorname{tr}(C - \sum_{i=1}^{k} B_i y_i - \sum_{i=1}^{l} G_i t_i) X + a^T y + b^T t. \end{aligned}$$

Note that

$$\max_{X \geq 0} \operatorname{tr}(C - \sum_{i=1}^{k} B_i y_i - \sum_{i=1}^{l} G_i t_i) X < \infty$$

only if $C - \sum_{i=1}^{k} B_i y_i - \sum_{i=1}^{l} G_i t_i \leq 0$. In this case, clearly the maximum occurs when

$$C - \sum_{i=1}^{k} B_i y_i - \sum_{i=1}^{l} G_i t_i = 0.$$

Therefore, we can the write *dual* SDP problem as

$$\begin{aligned} \max_{y \in \mathbb{R}^k, t \in \mathbb{R}^l_+} \quad & a^T y + b^T t \\ \text{subject to} \quad & \sum_{i=1}^{k} B_i y_i + \sum_{i=1}^{l} G_i t_i - C = Z \\ & Z \geq 0. \end{aligned} \tag{3.19}$$

From (3.15) and (3.19) it is clear that the problem of stability of the delay-differential Equation (3.13) can be formulated as the dual SDP defined by the elements

$$a := \mathbf{1}_{(N+1)L}, \quad b := \mathbf{0}, \quad G_i = \mathbf{0}, \quad C = \mathbf{0}, \quad B_i = F_i,$$

where we have defined F_i in (3.16).

SDPs are popular among controls community because not only they can be solved efficiently using convex optimization algorithms, but also a wide variety of problems in controls can be formulated as SDPs; e.g., robust stability (Bliman (2004a); Oliveira and Peres (2007)) and robust performance (Peaucelle and Arzelier (2001); Scherer (2006)) of uncertain systems, H_2/H_∞-optimal filter design (Li and Fu (1997); Geromel and de Oliveira (2001)), estimation of regions of attraction (Wang *et al.* (2005); Tan and Packard (2008); Topcu *et al.* (2010)) and reachability sets (Wang *et al.* (2013)) of nonlinear systems, stability and control of hybrid systems (Boukas (2006); Papchristodoulou and Prajna (2009)) and game theory (Parrilo (2006)). In the next section, we describe a state-of-the-art primal-dual algorithm by Helmberg *et al.* (2005) for solving SDPs.

Fortunately, there exists several interior-point algorithms in the literature for solving SDPs; e.g., dual scaling (Benson (2001); Benson *et al.* (1998)), primal-dual (Alizadeh *et al.* (1998); Monteiro (1997); Helmberg *et al.* (1996)) and cutting-plane/spectral bundle (Helmberg and Rendl (2000); Sivaramakrishnan (2010); Nayakkankuppam (2007)) algorithms. In our study, we are particularly interested in a state-of-the-art primal-dual algorithm proposed by Helmberg *et al.* (2005) mainly because at each iteration, it preserves a certain *property* (see (4.47)) of the primal and dual search directions. In Section 4.5, we will exploit this property to propose a distributed parallel version of this algorithm for solving large-scale SDPs in robust and/or nonlinear stability analysis. In the following, we briefly discuss the original version of this algorithm algorithm.

Similar to the barrier method described in Section 3.3, we can incorporate the inequality constraints in the dual SDP (3.19) using logarithmic barrier functions and the barrier parameter $\mu > 0$ as

$$\begin{aligned} \max_{y \in \mathbb{R}^k, t \in \mathbb{R}^l} \quad & a^T y + b^T t - \mu \left(\log\det Z + \sum_{i=1}^{l} \log t_i \right) \\ \text{subject to} \quad & \sum_{i=1}^{k} B_i y_i + \sum_{i=1}^{l} G_i t_i - C = Z. \end{aligned} \tag{3.20}$$

The Lagrangian for Problem (3.20) is defined as

$$\begin{aligned} L(X, y, t, Z) := a^T y + b^T t - \mu & \left(\log\det Z + \sum_{i=1}^{l} \log t_i \right) \\ & + \operatorname{tr}\left(\left(Z + C - \sum_{i=1}^{k} B_i y_i - \sum_{i=1}^{l} G_i t_i \right) X \right). \end{aligned}$$

Then, the KKT optimality conditions for Problem (3.20) is $\nabla L(X, y, t, Z) = 0$, which can be expanded as

$$\nabla_X L(X,y,t,Z) = Z + C - \sum_{i=1}^{k} B_i y_i - \sum_{i=1}^{l} G_i t_i = 0 \tag{3.21}$$

$$\nabla_y L(X,y,t,Z) = a - B(X) = 0 \tag{3.22}$$

$$\nabla_t L(X,y,t,Z) = b - G(X) - \mu\,[1/t_1, \cdots, 1/t_l]^T = 0 \tag{3.23}$$

$$\nabla_Z L(X,y,t,Z) = X - \mu Z^{-1} = 0, \tag{3.24}$$

where $B(X)$ and $G(x)$ are defined in (3.18).

Given a barrier parameter $\mu > 0$, at each iteration, the primal-dual algorithm finds a search direction $\Delta s := [\Delta X, \Delta y, \Delta t, \Delta Z]$ such that the new iterate $[X + \Delta X, y + \Delta y, t + \Delta t, Z + \Delta Z]$ belongs to the central path, i.e.,

$$\{[X_\mu, y_\mu, t_\mu, Z_\mu] : \mu \in [0, \infty] \text{ and } X_\mu, y_\mu, t_\mu, Z_\mu \text{ satisfy Conditions (3.21)-(3.24)}\}.$$

Conversely, given a point $[X, y, t, Z]$, one can use (3.23) and (3.24) to find its corresponding barrier parameter as

$$\mu = \frac{\mathrm{tr}(ZX) + [1/t_1, \cdots, 1/t_l]\,(b - G(X))}{n + l}. \tag{3.25}$$

The search direction Δs of the primal-dual algorithm is the sum of two steps: the *predictor* step $\Delta\hat{s} := [\Delta\hat{X}, \Delta\hat{y}, \Delta\hat{t}, \Delta\hat{Z}]$ and the *corrector* or *centering* step $\Delta\bar{s} := [\Delta\bar{X}, \Delta\bar{y}, \Delta\bar{t}, \Delta\bar{Z}]$. The predictor step is defined as the Newton's step for solving the optimality conditions (3.21)-(3.24) with $\mu = 0$, starting at any point (X, y, t, Z) which satisfies

$$X > 0, \quad Z > 0, \quad t > 0, \quad G(X) < b. \tag{3.26}$$

Similar to the Taylor's approximation in (3.12), we find the Newton's step by solving

$$\nabla L(X,y,t,Z) + \nabla^2 L(X,y,t,Z)\Delta\hat{s}^T = 0 \tag{3.27}$$

for $\Delta\hat{s}$. Substituting for ∇L from (3.21)-(3.24) into (3.27) yields the following system of equations for the predictor step.

$$\begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix} \begin{bmatrix} \Delta\hat{y} \\ \Delta\hat{t} \end{bmatrix} = \begin{bmatrix} B\left(Z^{-1}TX\right) - a) \\ G(Z^{-1}TX) - b) \end{bmatrix}, \tag{3.28}$$

$$\Delta\hat{X} = Z^{-1}TX - Z^{-1}\left(\sum_{i=1}^{k} B_i\Delta\hat{y} + \sum_{i=1}^{l} G_i\Delta\hat{t}\right)X - X \tag{3.29}$$

$$\Delta\hat{Z} = -T + \sum_{i=1}^{k} B_i\Delta\hat{y} + \sum_{i=1}^{l} G_i\Delta\hat{t} \tag{3.30}$$

where

$$T = -\sum_{i=1}^{k} B_i y + \sum_{i=1}^{l} G_i t + C + Z$$

$$\Lambda_{11} = \begin{bmatrix} B(Z^{-1}B_1X) & \cdots & B(Z^{-1}B_kX) \end{bmatrix}$$

$$\Lambda_{12} = \begin{bmatrix} B(Z^{-1}G_1X) & \cdots & B(Z^{-1}G_lX) \end{bmatrix}$$

$$\Lambda_{21} = \begin{bmatrix} G(Z^{-1}B_1X) & \cdots & G(Z^{-1}B_kX) \end{bmatrix}$$

$$\Lambda_{22} = \begin{bmatrix} G(Z^{-1}G_1X) & \cdots & G(Z^{-1}G_lX) \end{bmatrix} + \operatorname{diag}\left\{\frac{b_1 - \operatorname{tr}(G_1X)}{t_1}, \cdots, \frac{b_l - \operatorname{tr}(G_lX)}{t_l}\right\}.$$

The corrector step is defined as the Newton's step for solving the KKT conditions (3.21)-(3.24), using the barrier parameter μ as defined in (3.25) and starting at

$$[X + \Delta\hat{X}, y + \Delta\hat{y}, t + \Delta\hat{t}, Z + \Delta\hat{Z}],$$

where $[X, y, t, Z]$ can be any point satisfying (3.26) and $[\Delta\hat{X}, \Delta\hat{y}, \Delta\hat{t}, \Delta\hat{Z}]$ can be calculated using (3.28)-(3.30). Thus, to derive the corrector step, we substitute for ∇L from KKT conditions (3.21)-(3.24) into

$$\left[\nabla L(\bar{X}, \bar{y}, \bar{t}, \bar{Z}) + \nabla^2 L(\bar{X}, \bar{y}, \bar{t}, \bar{Z})\right]_{\substack{\bar{X}=X+\Delta\hat{X} \\ \bar{Z}=Z+\Delta\hat{Z} \\ \bar{y}=y+\Delta\hat{y} \\ \bar{t}=t+\Delta\hat{t}}} \Delta\bar{s}^T = 0$$

This yields the following set of equations for the corrector step.

$$\begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{bmatrix} \begin{bmatrix} \Delta\bar{y} \\ \Delta\bar{t} \end{bmatrix} = \begin{bmatrix} \mu B(Z^{-1}) - B(Z^{-1}\Delta\hat{Z}\Delta\hat{X}) \\ \mu \begin{bmatrix} \frac{1}{t_1} & \cdots & \frac{1}{t_l} \end{bmatrix} + G\left(X + \Delta\hat{X} + \mu Z^{-1} - Z^{-1}\Delta\hat{Z}\Delta\hat{X}\right) \end{bmatrix} \tag{3.31}$$

$$\Delta\bar{X} = Z^{-1}\left(-\Delta\hat{Z}\Delta\hat{X} + \mu I - \Delta\bar{Z}X\right) \tag{3.32}$$

$$\Delta\bar{Z} = \sum_{i=1}^{k} B_i \Delta\bar{y} + \sum_{i=1}^{l} G_i \Delta\bar{t} \tag{3.33}$$

By solving (3.28)-(3.30) for the predictor step and solving (3.31)-(3.33) for the corrector step, we can calculate the search direction as

$$\Delta s = \left[\mathrm{Sym}(\Delta\hat{X} + \Delta\bar{X}), \Delta\hat{y} + \Delta\bar{y}, \Delta\hat{t} + \Delta\bar{t}, \Delta\hat{Z} + \Delta\bar{Z}\right], \tag{3.34}$$

where $\mathrm{Sym}(W) := (W + W^T)/2$ is the symmetric part of matrix W. We have provided an outline of the discussed primal-dual algorithm in Algorithm 4.

Input: SDP elements C, a, b, B_i for $i = 1, \cdots, k$ and G_i for $i = 1, \cdots, l$; starting point satisfying (3.26); tolerance $\epsilon > 0$ the stopping criterion.

Initialization:

Set the duality gap $\gamma = 2\epsilon$.

while *duality gap* $\gamma > \epsilon$ **do**

Calculating the predictor step:

Solve $\Delta\hat{y}$ and $\Delta\hat{t}$ by solving system of the equations in (3.28).

Calculate $\Delta\hat{y}$ and $\Delta\hat{t}$ using (3.29) and (3.30).

Calculating the corrector step:

Calculate the barrier parameter μ using (3.25).

Solve $\Delta\bar{y}$ and $\Delta\bar{t}$ by solving system of the equations in (3.31).

Calculate $\Delta\bar{y}$ and $\Delta\bar{t}$ using (3.32) and (3.33).

Updating the primal and dual variables:

Calculate the search direction as

$$\Delta X := \mathrm{Sym}(\Delta\hat{X} + \Delta\bar{X}),\ \Delta y := \Delta\hat{y} + \Delta\bar{y},\ \Delta t := \Delta\hat{t} + \Delta\bar{t},\ \Delta Z := \Delta\hat{Z} + \Delta\bar{Z}.$$

Calculate primal and dual step lengths α_p and α_d using an appropriate line-search algorithm.

Set the primal and dual variables as

$$X := X + \alpha_p \Delta X, \quad y := y + \Delta y, \quad t := t + \Delta t, \quad Z := Z + \Delta Z.$$

Calculate the duality gap as $\gamma = \mathrm{tr}(CX) - (a^T y + b^T t)$.

end

Output: $[X^*, y^*, t^*, Z^*]$: A γ-suboptimal solution to Problems (3.17) and (3.19).

Algorithm 4: An interior-point central-path primal-dual algorithm for SDP

Chapter 4

PARALLEL ALGORITHMS FOR ROBUST STABILITY ANALYSIS OVER SIMPLEX

4.1 Background and motivations

Control system theory when applied in practical situations often involves the use of large state-space models, typically due to inherent complexity of the system, the interconnection of subsystems, or the reduction of an infinite-dimensional or PDE model to a finite-dimensional approximation. One approach to dealing with such large scale models has been to use model reduction techniques such as *balanced truncation* (Gugercin and Antoulas (2004)). However, the use of model reduction techniques are not necessarily robust and can result in arbitrarily large errors. In addition to large state-space, practical problems often contain uncertainty in the model due to modeling errors, linearization, or fluctuation in the operating conditions. The problem of stability and control of systems with uncertainty has been widely studied. See, e.g. the texts Ackermann *et al.* (2001); Bhattacharyya *et al.* (1995); Green and Limebeer (1995); Zhou and Doyle (1998); Dullerud and Paganini (2000). Famous results such as the *small-gain* theorem, *Popov's* criterion, *passivity* theorems and *Kharitonov's* theorem have been widely used to find tractable solutions to certain robust stability problems of a single and/or interconnected uncertain systems. As an example, Kharitonov's theorem reduces the stability problem of an infinite family of differential equations

$$a_1 \frac{d^n}{dt^n} x(t) + a_2 \frac{d^{n-1}}{dt^{n-1}} x(t) + \cdots + a_{n-2} \frac{d}{dt} x(t) + a_{n+1} x(t) + a_{n+2} = 0, \quad a_i \in [\underline{u}_i, \bar{u}_i] \subset \mathbb{R} \tag{4.1}$$

to verifying whether the following four characteristic polynomials

$$k_1(s) = \underline{u}_{n+2} + \underline{u}_{n+1}s + \bar{u}_n s^2 + \bar{u}_{n-1}s^3 + \underline{u}_{n-2}s^4 + \underline{u}_{n-3}s^5 + \cdots$$

$$k_2(s) = \bar{u}_{n+2} + \bar{u}_{n+1}s + \underline{u}_n s^2 + \underline{u}_{n-1}s^3 + \bar{u}_{n-2}s^4 + \bar{u}_{n-3}s^5 + \cdots$$

$$k_3(s) = \underline{u}_{n+2} + \bar{u}_{n+1}s + \bar{u}_n s^2 + \underline{u}_{n-1}s^3 + \underline{u}_{n-2}s^4 + \bar{u}_{n-3}s^5 + \cdots$$

$$k_4(s) = \bar{u}_{n+2} + \underline{u}_{n+1}s + \underline{u}_n s^2 + \bar{u}_{n-1}s^3 + \bar{u}_{n-2}s^4 + \underline{u}_{n-3}s^5 + \cdots$$

have all their roots in the open left half-plane - a problem which can be tractably solved (in $\mathcal{O}(n^2)$ operations) using the Routh-Hurwitz criterion. Despite all the progress in robust control theory during the past few decades, a drawback of existing computational methods for analysis and control of systems with uncertainty is high computational complexity. This is a consequence of the fact that a wide range of problems in robust stability and control of systems with parametric uncertainty are known to be NP-hard. For example, even the classical problem of stability of $\dot{x}(t) = A(a)x(t)$ for all a inside a hypercube (the matrix analog of System (4.1)) is NP-hard[1]. Other examples are calculation of structured singular values for robust performance analysis and μ-synthesis (Zhou *et al.* (1996)), deciding null-controllability[2] of $x(k+1) = f(x(k), u(k))$ for a given $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ (Blondel and Tsitsiklis (1999)), and computing arbitrarily precise bounds on the joint spectral radius of matrices for stability analysis of systems with time-varying uncertainty (Gripenberg (1996)). See Blondel and Tsitsiklis (2000) for a comprehensive survey on NP-hard problems in control theory. The result of such complexity is that for systems with

[1]Nemirovskii (1993) proves that the $\{-1, +1\}$-integer linear programming problem (a well-known NP-complete problem) admits a polynomial-time reduction to the problem of verifying positive semi-definiteness of a family of symmetric matrices with entries belonging to an interval on $\mathbb{R}$.

[2]A system $x(k+1) = f(x(k), u(k))$ is called null-controllable if for every initial state $x(0)$, there exist some $T > 1$ and controls $u(k), k = 0, \cdots, T-1$ such that $x(T) = 0$

parametric uncertainty and with hundreds of states, existing algorithms fail with the primary point of failure usually being lack of unallocated memory.

In this dissertation, we seek to distribute the computation over an array of processors within the context of existing computational resources; specifically cluster-computers and supercomputers. When designing algorithms to run in a parallel computing environment, one must both synchronize computational tasks among the processors while minimizing communication overhead among the processors. This can be difficult, as each architecture has a specific memory hierarchy and communication graph (See Figure 4.1). Likewise, in a lower level, individual computing units may have different processing architectures and memory hierarchies; e.g., see a comparison of the memory hierarchy of a multi-core CPU and a GPU in Figure 4.1. We account for communication by explicitly modeling the required communication graph between processors. This communication graph is then mapped to the processor architecture using the Message-Passing Interface (MPI) (Walker and Dongarra (1996)). While there are many algorithms for robust stability analysis and control of linear systems, ours is the first which explicitly accounts for the processing architecture in the emerging multi-core computing environment.

Our approach to robust stability is based on the well-established use of parameter-dependent Quadratic-In-The-State (QITS) Lyapunov functions. The use of parameter-dependent Lyapunov QITS functions eliminates the conservativity associated with e.g. quadratic stability (Packard and Doyle (1990)), at the cost of requiring some restriction on the rate of parameter variation. Specifically, our QITS Lyapunov variables are polynomials in the vector of uncertain parameters. This is a generalization of the use of QITS Lyapunov functions with affine parameter dependence as in Barmish and DeMarco (1986) and expanded in, e.g. Gahinet *et al.* (1996); Oliveira and Peres (2005, 2001) and Ramos and Peres (2001). The use of polynomial QITS Lyapunov

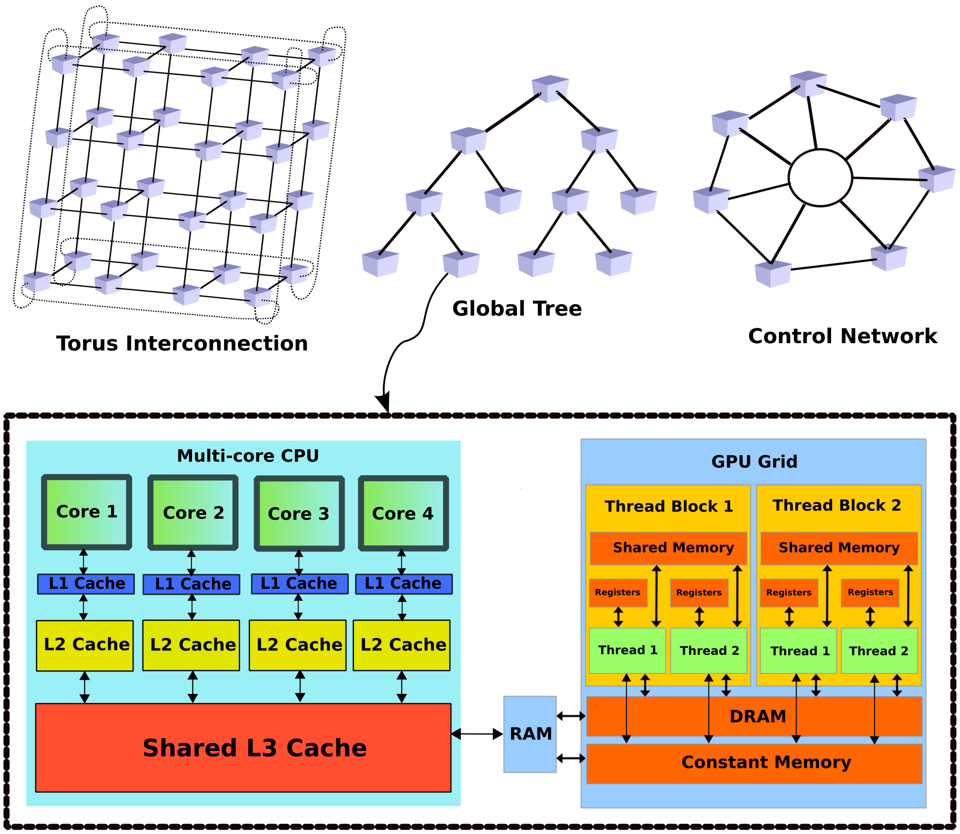


Figure 4.1: Various interconnections of nodes in a cluster computer (Top), Typical memory hierarchies of a GPU and a multi-core CPU (bottom)

variables can be motivated by Bliman (2004b), wherein it is shown that any feasible parameter-dependent LMI with parameters inside a compact set has a polynomial solution or Peet (2009) wherein it is shown that local stability of a nonlinear vector field implies the existence of a polynomial Lyapunov function.

There are several results which use polynomial QITS Lyapunov functions to prove robust stability. In most cases, the stability problem is reduced to the general problem of optimization of polynomial variables subject to LMI constraints - an NP-hard problem (Ben-Tal and Nemirovski (1998)). To avoid NP-hardness, the optimization of polynomials problem is usually solved in an asymptotic manner by posing a sequence of sufficient conditions of increasing accuracy and decreasing conservatism. For example, building on the result in Bliman (2004b), Bliman (2004a) proposes a sequence of increasingly precise LMIs for robust stability analysis of linear systems with affine dependency on uncertain parameters on the complex unit ball. Necessary and sufficient stability conditions for linear systems with one uncertain parameter are derived in Zhang and Tsiotras (2003), providing an explicit bound on the degree of the polynomial-type Lyapunov function. This result is extended to multi-parameter-dependent linear systems in Zhang *et al.* (2005). Another important approach to optimization of polynomials is the SOS methodology which replaces the polynomial positivity constraint with the constraint that the polynomial admits a representation as a sum of squares of polynomials. See Sections 2.3.4 and 1.1 for a review of this approach. Applications of the SOS methodology in robust stability analysis of linear and nonlinear systems can be found in Scherer and Hol (2006); Lavaei and Aghdam (2008) and Tan and Packard (2008). While the SOS methodology have been extensively utilized in the literature, we have not, as of yet, been able to adapt algorithms for solving the resulting LMI conditions to a parallel-computing environment. Finally, there have been multiple results in recent years on the use of Polya's theorem to solve optimization of polynomials problems (Oliveira and Peres (2007)) on the simplex. An extension of Polya's theorem for uncertain parameters on the multisimplex or hypercube can be found in Oliveira *et al.* (2008). In this section, we propose an extension of Polya's theorem and its use for solving optimization of polynomials problems in a parallel computing environment.

Our goal is to create algorithms which explicitly map computation, communication and storage to existing parallel processing architectures. This goal is motivated by the failure of existing general-purpose Semi-Definite Programming (SDP) solvers to efficiently utilize platforms for large-scale computation. Specifically, it is well-established that linear programming and semi-definite programming both belong to the complexity class P-Complete, also known as the class of inherently sequential problems. Although there have been several attempts to map certain SDP solvers to a parallel computing environment (Borchers and Young (2007); Yamashita *et al.* (2003)), certain critical steps cannot be distributed. The result is that as the number of processors increases, certain computational and communication bottlenecks dominate - leading to a saturation in the speed-up (the increase in processing speed per additional processor) of these solvers (Amdahl's law (Amdahl (1967))). We avoid these bottlenecks by exploiting the particular structure of the LMI conditions associated with Polya's theorem. Note that, in principle, a perfectly designed general-purpose SDP algorithm could identify the structure of the SDP, as we have, and map the communication, computation and memory constraints to a parallel architecture. Indeed, there has been a great deal of research on creating programming languages which attempt to do just this (Kalé *et al.* (1994); Deitz (2005)). However, at present such languages are mostly theoretical and have certainly not been incorporated into existing SDP solvers.

In addition to parallel SDP solvers, there have been some efforts to exploit structure in certain polynomial optimization algorithms to reducing the size and complexity of the resulting LMI's. For example, for the case of finding SOS representations for symmetric polynomials[3], Gatermann and Parrilo (2004) exploited symmetry to reduce the number of decision variables and constraints in the associated SDPs. Another example is the use of an specific sparsity structure in Parrilo (2005); Kim *et al.* (2005) and Waki *et al.* (2008) to reduce the complexity of the linear algebra calculations associated with the SOS methodology. The use of generalized Lagrangian duals and Groebner basis techniques for reducing the complexity of the SDPs associated with the SOS decompositions of sparse polynomial optimization problems can

[3]A symmetric polynomial is a polynomial which is invariant under all permutations of its variables, e.g., $f(x,y,z) = x^4 + y^4 + z^4 - 4xyz + x + y + z$.

be found in Kim *et al.* (2005) and Permenter and Parrilo (2012).

4.1.1 Our contributions

In this section, we focus on robust stability analysis of: 1- Systems with parametric uncertainty inside a simplex; and 2- Systems with parametric uncertainty inside a hypercube. We solve each problem in two phases by proposing the following algorithms: 1- A decentralized algorithm for Setting up the sequence of structured SDPs associated with Polya's theorem; and 2- A parallel SDP solver to solve the SDPs. Note that the problem of decentralizing the set-up algorithm is significant in that for large-scale systems, the instantiation of the problem may be beyond the memory and computational capacity of a single processing node. For the set-up problem, the algorithm that we propose has no centralized memory/computational requirements whatsoever. Furthermore, we show that for a sufficiently large number of available processors, the communication complexity is independent of the size of the state-space or the number of Polya's iterations.

In the second phase, we propose a variant of Helmberg's primal-dual algorithm (Helmberg *et al.* (2005)) and map the computational, memory and communication requirements to a parallel computing environment. Unlike the set-up algorithm, the primal-dual algorithm does have a "relatively small" centralized computation associated with the update of the dual variables. However, we have structured the algorithm so that the size of this centralized computation is solely a function of the degree of the polynomial Lyapunov function and does not depend on the number of Polya's iterations. In addition, there is no point-to-point communication between the processors, which means that the algorithm is compatible with most of the existing parallel computing architectures. We will provide a graph representation of the communication architecture of both the set-up and SDP algorithms.

By linking the set-up and SDP algorithms and conducting tests on various cluster computers, we demonstrate the ability of our algorithms in performing robust stability analysis on systems with 100+ states and several uncertain parameters. Specifically, we ran a series of numerical experiments using the Linux-based cluster computer Karlin at Illinois Institute of Technology and the Blue Gene supercomputer (with 200 processor allocation). First, we applied the algorithm to a current problem in robust stability analysis of magnetic confinement fusion using a discretized PDE model. Next, we examine the accuracy of the algorithm as Polya's iterations progress and compare this accuracy with the SOS approach. We show that unlike the general-purpose parallel SDP solver SDPARA Yamashita *et al.* (2003), the speed-up of our algorithm shows no evidence of saturation. Finally, we calculate the envelope of the algorithm on the cluster computer Karlin in terms of the maximum state-space dimension, number of processors and Polya's iterations.

4.2 Notation and Preliminaries on homogeneous polynomials

Let us denote an $l-$variate monomial as $\alpha^\gamma = \prod_{i=1}^{l} \alpha_i^{\gamma_i}$, where $\alpha \in \mathbb{R}^l$ is the vector of variables, $\gamma \in \mathbb{N}^l$ is the vector of exponents and $\sum\limits_{i=1}^{l} \gamma_i = d$ is the degree of the monomial. We define

$$W_d := \left\{ \gamma \in \mathbb{N}^l : \sum_{i=1}^{l} \gamma_i = d \right\} \tag{4.2}$$

as the totally ordered set of the exponents of $l-$variate monomials of degree d, where the ordering is lexicographic. Recall that in lexicographical ordering $\gamma \in W_d$ precedes $\eta \in W_d$, if the left most non-zero entry of $\gamma - \eta$ is positive. The lexicographical index of every $\gamma \in W_d$ can be calculated using the map $\langle \cdot \rangle : \mathbb{N}^l \to \mathbb{N}$ defined as (Peet and Peet (2010))

$$\langle \gamma \rangle = \sum_{j=1}^{l-1} \sum_{i=1}^{\gamma_i} f\left(l-j, d+1-\sum_{k=1}^{j-1} \gamma_k - i \right) + 1$$

where

$$f(l,d) := \begin{cases} 0 & \text{for } \ l=0 \\ \begin{pmatrix} l+d-1 \\ l-1 \end{pmatrix} = \dfrac{(d+l-1)!}{d!(l-1)!} & \text{for } \ l>0, \end{cases} \tag{4.3}$$

is the cardinality of W_d, i.e., the number of $l-$variate monomials of degree d. For convenience, we also denote the index of a monomial α^γ by $\langle\gamma\rangle$. We represent $l-$variate homogeneous polynomials of degree d_p as

$$P(\alpha) = \sum_{\gamma \in W_{d_p}} P_{\langle\gamma\rangle}\alpha^\gamma,$$

where $P_{\langle\gamma\rangle} \in \mathbb{R}^{n\times n}$ is the matrix coefficient of the monomial α^γ.

Now consider the linear system

$$\dot{x}(t) = A(\alpha)x(t), \tag{4.4}$$

where $A(\alpha) \in \mathbb{R}^{n\times n}$ and $\alpha \in Q \subset \mathbb{R}^l$ is a vector of uncertain parameters. We assume that $A(\alpha)$ is a homogeneous polynomial and $Q = \Delta^l \subset \mathbb{R}^l$, where Δ^l is the unit simplex, i.e.,

$$\Delta^l = \left\{ \alpha \in \mathbb{R}^l : \sum_{i=1}^{l} \alpha_i = 1, \alpha_i \geqslant 0 \right\}.$$

If $A(\alpha)$ is not homogeneous, we can homogenize it in the following manner. Suppose $A(\alpha)$ with $\alpha \in \Delta^l$ is a non-homogeneous polynomial of degree d_a and has N_a monomials with non-zero coefficients. Define $D = \left(d_{a_1}, \cdots, d_{a_{N_a}}\right)$, where d_{a_i} is the degree of the i^{th} monomial of $A(\alpha)$ according to the lexicographical ordering. Now define the polynomial $B(\alpha)$ as per the following.

1. Let $B = A$.

2. For $i = 1, \cdots, N_a$, multiply the i^{th} monomial of $B(\alpha)$, according to lexicographical ordering, by $\left(\sum_{j=1}^{l} \alpha_j\right)^{d_a - d_{a_i}}$.

Then, since $\sum_{j=1}^{l} \alpha_j = 1$, $B(\alpha) = A(\alpha)$ for all $\alpha \in \Delta^l$ and hence all properties of $\dot{x}(t) = A(\alpha)x(t)$ for any $\alpha \in \Delta^l$ are retained by the homogeneous system $\dot{x}(t) = B(\alpha)x(t)$. To further clarify the homogenization procedure, we provide the following example.

Example: Construction of the homogeneous system $\dot{x}(t) = B(\alpha)x(t)$.

Consider the non-homogeneous polynomial $A(\alpha) = C\alpha_1^2 + D\alpha_2 + E\alpha_3 + F$ of degree $d_a = 2$, where $[\alpha_1, \alpha_2, \alpha_3] \in \Delta^3$. Using the above procedure, the homogeneous polynomial $B(\alpha)$ can be constructed as

$$\begin{aligned} B(\alpha) &= C\alpha_1^2 + D\alpha_2(\alpha_1+\alpha_2+\alpha_3) + E\alpha_3(\alpha_1+\alpha_2+\alpha_3) + F(\alpha_1+\alpha_2+\alpha_3)^2 \\ &= \underbrace{(C+F)}_{B_1}\alpha_1^2 + \underbrace{(D+2F)}_{B_2}\alpha_1\alpha_2 + \underbrace{(E+2F)}_{B_3}\alpha_1\alpha_3 + \underbrace{(D+F)}_{B_4}\alpha_2^2 \\ &+ \underbrace{(D+E+2F)}_{B_5}\alpha_2\alpha_3 + \underbrace{(E+F)}_{B_6}\alpha_3^2 = \sum_{\gamma \in W_2} B_{\langle\gamma\rangle}\alpha^\gamma. \end{aligned} \tag{4.5}$$

4.3 Setting up the problem of robust stability analysis over a simplex

In this section, we show that applying Polya's Theorem to the robust stability problem, i.e., the inequalities in Theorem 13 yields a semi-definite program with a block-diagonal structure - hence can be an efficiently distributed among processing units. We start by stating the following well-known Lyapunov result on stability of System (4.4).

Theorem 13. *System* (4.4) *is stable if and only if there exists a polynomial matrix $P(\alpha)$ such that $P(\alpha) \succ 0$ for all $\alpha \in \Delta^l$ and*

$$A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha) \prec 0 \quad \textit{for all } \alpha \in \Delta^l. \tag{4.6}$$

A similar condition also holds for discrete-time linear systems. The conditions associated with Theorem 13 are infinite-dimensional LMIs, meaning they must hold at infinite number of points. Such problems are known to be NP-hard (Ben-Tal and Nemirovski (1998)). Our goal is to derive a sequence of polynomial-time algorithms such that their outputs converge to a solution of the parameter-dependent LMI in (5.8). Key to this result is Polya's Theorem (Hardy *et al.* (1934)). A variation of this theorem for matrices is given as follows.

Theorem 14. *(Polya's theorem, simplex version) If a homogeneous matrix-valued polynomial F satisfies $F(\alpha) > 0$ for all $\alpha \in \Delta^l$, then there exists $d \geq 0$ such that all the coefficients of*

$$\left(\sum_{i=1}^{l} \alpha_i\right)^d F(\alpha) \tag{4.7}$$

are positive definite.

See Chapter 2 for a more detailed discussion on this result.

Consider the stability of the system described by Equation (4.4). We are interested in finding a $P(\alpha)$ which satisfies the conditions of Theorem 13. According to Polya's theorem, the constraints of Theorem 13 are satisfied if for some sufficiently large d_1 and d_2, the polynomials

$$\left(\sum_{i=1}^{l} \alpha_i\right)^{d_1} P(\alpha) \qquad \text{and} \tag{4.8}$$

$$-\left(\sum_{i=1}^{l} \alpha_i\right)^{d_2} \left(A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha)\right) \tag{4.9}$$

have all positive definite coefficients.

Let $P(\alpha)$ be a homogeneous polynomial of degree d_p which can be represented as

$$P(\alpha) = \sum_{\gamma \in W_{d_p}} P_{\langle\gamma\rangle} \alpha^\gamma, \tag{4.10}$$

where the coefficients $P_{\langle\gamma\rangle} \in \mathbb{S}^n$ and where we recall that $W_{d_p} := \left\{\gamma \in \mathbb{N}^l : \sum_{i=1}^{l} \gamma_i = d_p\right\}$ is the set of the exponents of all l-variate monomials of degree d_p. Since $A(\alpha)$ is a homogeneous polynomial of degree d_a, we can write it as

$$A(\alpha) = \sum_{\gamma \in W_{d_a}} A_{\langle \gamma \rangle} \alpha^{\gamma}, \tag{4.11}$$

where the coefficients $A_{\langle \gamma \rangle} \in \mathbb{R}^{n \times n}$. By substituting (4.10) and (4.11) into (4.8) and (4.9) and defining d_{pa} as the degree of $P(\alpha)A(\alpha)$, the conditions of Theorem 14 can be represented in the form

$$\left(\sum_{i=1}^{l} \alpha_i \right)^{d_1} \left(\sum_{h \in W_{d_p}} P_{\langle h \rangle} \alpha^h \right) = \sum_{g \in W_{d_p+d_1}} \left(\sum_{h \in W_{d_p}} \beta_{\langle h \rangle, \langle \gamma \rangle} P_{\langle h \rangle} \right) \alpha^{\gamma}$$

and

$$-\left(\sum_{i=1}^{l} \alpha_i \right)^{d_2} \left(\left(\sum_{h \in W_{d_a}} A_{\langle h \rangle}^T \alpha^h \right) \left(\sum_{h \in W_{d_p}} P_{\langle h \rangle} \alpha^h \right) + \left(\sum_{h \in W_{d_p}} P_{\langle h \rangle} \alpha^h \right) \left(\sum_{h \in W_{d_a}} A_{\langle h \rangle} \alpha^h \right) \right)$$
$$= \sum_{\gamma \in W_{d_{pa}+d_2}} \left(\sum_{h \in W_{d_p}} H_{\langle h \rangle, \langle \gamma \rangle}^T P_{\langle h \rangle} + P_{\langle h \rangle} H_{\langle h \rangle, \langle \gamma \rangle} \right) \alpha^{\gamma}$$

have all positive coefficients. This means that

$$\sum_{h \in W_{d_p}} \beta_{\langle h \rangle, \langle \gamma \rangle} P_{\langle h \rangle} > 0 \qquad \text{for all} \quad \gamma \in W_{d_p+d_1} \text{ and} \tag{4.12}$$

$$\sum_{h \in W_{d_p}} (H_{\langle h \rangle, \langle \gamma \rangle}^T P_{\langle h \rangle} + P_{\langle h \rangle} H_{\langle h \rangle, \langle \gamma \rangle}) < 0 \qquad \text{for all } \gamma \in W_{d_{pa}+d_2}. \tag{4.13}$$

Here we have defined $\beta_{\langle h \rangle, \langle \gamma \rangle}$ to be the scalar coefficient which multiplies $P_{\langle h \rangle}$ in the $\langle \gamma \rangle$-th monomial of the homogeneous polynomial $\left(\sum_{i=1}^{l} \alpha_i \right)^{d_1} P(\alpha)$ using the lexicographical ordering. Likewise, $H_{\langle h \rangle, \langle \gamma \rangle} \in \mathbb{R}^{n \times n}$ is the term which left or right multiplies $P_{\langle h \rangle}$ in the $\langle \gamma \rangle$-th monomial of $\left(\sum_{i=1}^{l} \alpha_i \right)^{d_2} \left(A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha) \right)$ using the lexicographical ordering. For an intuitive explanation as to how these β and H terms are calculated, we consider a simple example. Precise formulae for these terms will follow the example.

Example: Calculating the β and H coefficients.

Consider $A(\alpha) = A_1\alpha_1 + A_2\alpha_2$ and $P(\alpha) = P_1\alpha_1 + P_2\alpha_2$. By expanding Equation (4.8) for $d_1 = 1$ we have $(\alpha_1 + \alpha_2)P(\alpha) = P_1\alpha_1^2 + (P_1 + P_2)\alpha_1\alpha_2 + P_2\alpha_2^2$. The coefficients $\beta_{\langle h\rangle,\langle\gamma\rangle}$ are then extracted as

$$\beta_{1,1} = 1,\ \beta_{2,1} = 0,\ \beta_{1,2} = 1,\ \beta_{2,2} = 1,\ \beta_{1,3} = 0,\ \beta_{2,3} = 1.$$

Next, by expanding Equation (4.9) for $d_2 = 1$ we have

$$\begin{aligned}&(\alpha_1 + \alpha_2)\left(A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha)\right) = \\ &\left(A_1^TP_1 + P_1A_1\right)\alpha_1^3 + \left(A_1^TP_1 + P_1A_1 + A_2^TP_1 + P_1A_2 + A_1^TP_2 + P_2A_1\right)\alpha_1^2\alpha_2 + \left(A_2^TP_1\right. \\ &\qquad \left. + P_1A_2 + A_1^TP_2 + P_2A_1 + A_2^TP_2 + P_2A_2\right)\alpha_1\alpha_2^2 + \left(A_2^TP_2 + P_2A_2\right)\alpha_2^3.\end{aligned}$$

The coefficients $H_{\langle h\rangle,\langle\gamma\rangle}$ are then extracted as

$$\begin{array}{llll} H_{1,1} = A_1, & H_{2,1} = \mathbf{0}, & H_{1,2} = A_1 + A_2, & H_{2,2} = A_1, \\ H_{1,3} = A_2, & H_{2,3} = A_1 + A_2, & H_{1,4} = \mathbf{0}, & H_{2,4} = A_2. \end{array}$$

4.3.1 *General formulae for calculating coefficients β and H*

The set $\{\beta_{\langle h\rangle,\langle\gamma\rangle}\}$ of coefficients can be formally defined recursively as follows. Let the initial values for $\beta_{\langle h\rangle,\langle\gamma\rangle}$ be defined as

$$\beta^{(0)}_{\langle h\rangle,\langle\gamma\rangle} = \begin{cases} 1 & \text{if } h = \gamma \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } \ \gamma \in W_{d_p} \text{ and } h \in W_{d_p}. \tag{4.14}$$

Then, iterating for $i = 1, \ldots d_1$, we let

$$\beta^{(i)}_{\langle h\rangle,\langle\gamma\rangle} = \sum_{\lambda \in W_1} \beta^{(i-1)}_{\langle h\rangle,\langle\gamma-\lambda\rangle} \qquad \text{for all } \ \gamma \in W_{d_p+i} \text{ and } h \in W_{d_p}. \tag{4.15}$$

Finally, we set $\{\beta_{\langle h\rangle,\langle\gamma\rangle}\} = \{\beta^{d_1}_{\langle h\rangle,\langle\gamma\rangle}\}$.

To obtain the set $\{H_{\langle h\rangle,\langle\gamma\rangle}\}$ of coefficients, set the initial values as

$$H^{(0)}_{\langle h\rangle,\langle\gamma\rangle} = \sum_{\lambda \in W_{d_a} : \lambda + h = \gamma} A_{\langle\lambda\rangle} \ \text{ for all } \ \gamma \in W_{d_p+d_a} \text{ and } h \in W_{d_p}. \tag{4.16}$$

Then, iterating for $i = 1, \dots d_2$, we let

$$H^{(i)}_{\langle h \rangle, \langle \gamma \rangle} = \sum_{\lambda \in W_1} H^{(i-1)}_{\langle h \rangle, \langle \gamma - \lambda \rangle} \quad \text{for all } \ \gamma \in W_{d_{pa}+i} \text{ and } h \in W_{d_p}. \tag{4.17}$$

Finally, set $\{H_{\langle h \rangle, \langle \gamma \rangle}\} = \{H^{d_2}_{\langle h \rangle, \langle \gamma \rangle}\}$.

For the case of large-scale systems, computing and storing $\{\beta_{\langle h \rangle, \langle \gamma \rangle}\}$ and $\{H_{\langle h \rangle, \langle \gamma \rangle}\}$ is a significant challenge due to the number of these coefficients. Specifically, the number of terms increases with l (number of uncertain parameters in System (4.4)), d_p (degree of $P(\alpha)$), d_{pa} (degree of $P(\alpha)A(\alpha)$) and d_1, d_2 (Polya's exponents) as follows.

4.3.2 *Number of coefficients $\beta_{\langle h \rangle, \langle \gamma \rangle}$ and $H_{\langle h \rangle, \langle \gamma \rangle}$*

Given l, d_p and d_1, since $h \in W_{d_p}$ and $\gamma \in W_{d_p + d_1}$, the number of coefficients $\beta_{\langle h \rangle, \langle \gamma \rangle}$ is the product of $L_0 := \text{card}(W_{d_p})$ and $L := \text{card}(W_{d_p+d_1})$. Recall that $\text{card}(W_{d_p})$ is the number of all l-variate monomials of degree d_p and can be calculated using (4.3) as follows.

$$L_0 = f(l, d_p) = \begin{cases} 0 & \text{for } \ l = 0 \\ \begin{pmatrix} d_p + l - 1 \\ l - 1 \end{pmatrix} = \dfrac{(d_p + l - 1)!}{d_p!(l-1)!} & \text{for } \ l > 0. \end{cases} \tag{4.18}$$

Likewise, $\text{card}(W_{d_p+d_1})$, i.e., the number of all $l-$variate monomials of degree $d_p + d_1$ is calculated using (4.3) as follows.

$$L = f(l, d_p + d_1) = \begin{cases} 0 & \text{for } \ l = 0 \\ \begin{pmatrix} d_p + d_1 + l - 1 \\ l - 1 \end{pmatrix} = \dfrac{(d_p + d_1 + l - 1)!}{(d_p + d_1)!(l-1)!} & \text{for } \ l > 0. \end{cases} \tag{4.19}$$

The number of coefficients $\beta_{\langle h \rangle, \langle \gamma \rangle}$ is $L_0 \cdot L$. In Figure 4.2, we have plotted the number of coefficients $\beta_{\langle h \rangle, \langle \gamma \rangle}$ in terms of the number of uncertain parameters l and for different polya's exponents.

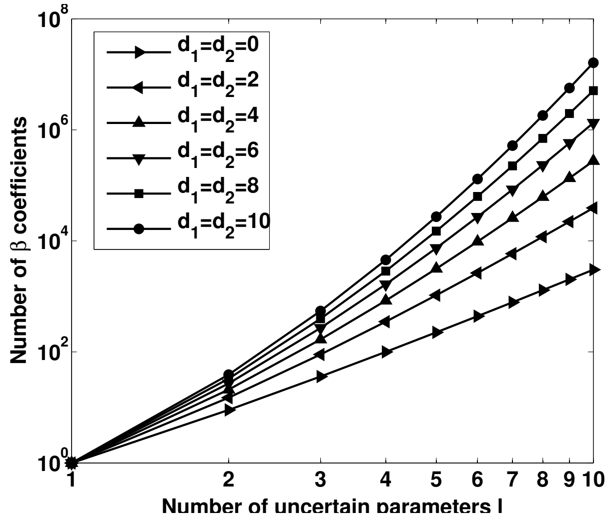


Figure 4.2: Number of $\beta_{\langle h\rangle,\langle\gamma\rangle}$ coefficients vs. the number of uncertain parameters for different Polya's exponents and for $d_p = 2$

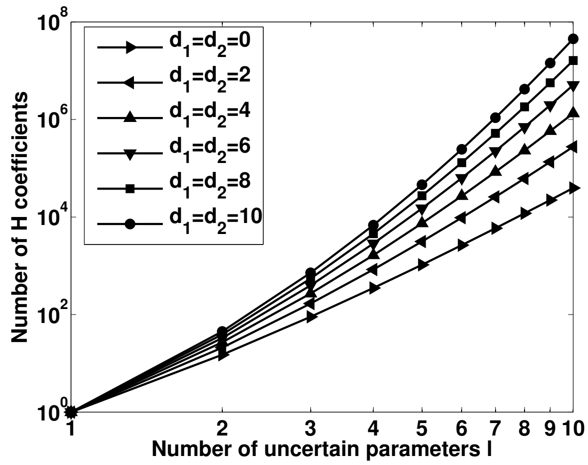


Figure 4.3: Number of $H_{\langle h\rangle,\langle\gamma\rangle}$ coefficients vs. the number of uncertain parameters for different Polya's exponents and for $d_p = d_a = 2$

Given l, d_p, d_a and d_2, since $h \in W_{d_p}$ and $\gamma \in W_{d_{pa}+d_2}$, the number of coefficients $H_{\langle h\rangle,\langle\gamma\rangle}$ is the product of $L_0 := \text{card}(W_{d_p})$ and $M := \text{card}(W_{d_{pa}+d_2})$. By using (4.3), we have

$$M = f(l, d_{pa}+d_2) = \begin{cases} 0 & \text{for } \ l = 0 \\ \begin{pmatrix} d_{pa}+d_2+l-1 \\ l-1 \end{pmatrix} = \dfrac{(d_{pa}+d_2+l-1)!}{(d_{pa}+d_2)!(l-1)!} & \text{for } \ l > 0. \end{cases} \tag{4.20}$$

The number of $H_{\langle h\rangle,\langle\gamma\rangle}$ coefficients is $L_0 \cdot M$. In Figure 4.3, we have plotted the number of coefficients $H_{\langle h\rangle,\langle\gamma\rangle}$ in terms of the number of uncertain parameters l and

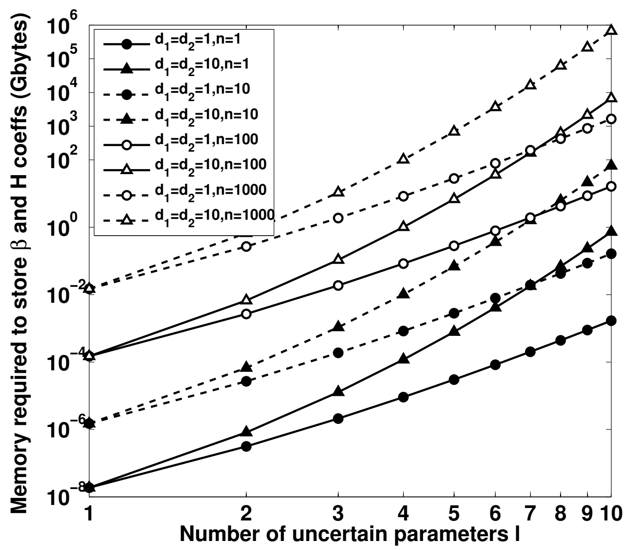


Figure 4.4: Memory required to store the coefficients $\beta_{\langle h\rangle,\langle\gamma\rangle}$ and $H_{\langle h\rangle,\langle\gamma\rangle}$ vs. number of uncertain parameters, for different d_1, d_2 and $d_p = d_a = 2$

for different polya's exponents.

We have shown the required memory to store the coefficients $\beta_{\langle h\rangle,\langle\gamma\rangle}$ and $H_{\langle h\rangle,\langle\gamma\rangle}$ in Figure 4.4 in terms of the number of uncertain parameters l and for different Polya's exponents. It is observed from Figure 4.4 that even for small degree d_p of $P(\alpha)$ and small degree d_a of the system matrix $A(\alpha)$, the required memory is in the Terabyte range. Peet and Peet (2010) proposed a decentralized computing approach to the calculation of $\{\beta_{\langle h\rangle,\langle\gamma\rangle}\}$ on a cluster computer. In the work, we extend this method to the calculation of $\{H_{\langle h\rangle,\langle\gamma\rangle}\}$ and the SDP elements which will be discussed in the following section. We express the LMIs associated with conditions (4.12) and (4.13) as an SDP in both primal and dual forms. We will also discuss the structure of the primal and dual SDP variables and the constraints.

4.3.3 The elements of the SDP problem associated with Polya's theorem

Recall from Section 3.4 that a semi-definite program can be stated either in primal or in dual format. Given $C \in \mathbb{S}^m$, $a \in \mathbb{R}^K$ and $B_i \in \mathbb{S}^m$, here we consider

$$\begin{aligned} \max_{X \in \mathbb{S}^m} \quad & \operatorname{tr}(CX) \\ \text{subject to} \quad & B(X) = a \\ & X \geq 0, \end{aligned} \tag{4.21}$$

as the primal SDP form, where the linear operator $B : \mathbb{S}^m \to \mathbb{R}^K$ is defined in (3.18). The associated dual problem is

$$\begin{aligned} \min_{y,Z} \quad & a^T y \\ \text{subject to} \quad & \sum_{i=1}^{K} B_i y_i - C = Z \\ & Z \geq 0 \ , \ y \in \mathbb{R}^K. \end{aligned} \tag{4.22}$$

The elements C, B_i and a of the SDP problem associated with the LMIs in (4.12) and (4.13) are defined as follows. We define the element C as

$$C := \operatorname{diag}(C_1, \cdots C_L, C_{L+1}, \cdots C_{L+M}), \tag{4.23}$$

where

$$C_i := \begin{cases} \delta I_n \cdot \left(\sum_{h \in W_{d_p}} \beta_{\langle h \rangle, i} \frac{d_p!}{h_1! \cdots h_l!} \right), & \text{for} \quad 1 \leq i \leq L \\ 0_n, & \text{for} \quad L+1 \leq i \leq L+M, \end{cases} \tag{4.24}$$

where recall that $L = \operatorname{card}(W_{d_p+d_1})$ is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_1} P(\alpha)$, $M = \operatorname{card}(W_{d_{pa}+d_2})$ is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_2} P(\alpha)A(\alpha)$, n is the dimension of System (4.4), l is the number of uncertain parameters and δ is a small positive parameter.

For $i = 1, \cdots, K$, define B_i elements as

$$B_i := \mathrm{diag}(B_{i,1}, \cdots B_{i,L}, B_{i,L+1}, \cdots B_{i,L+M}), \tag{4.25}$$

where K is the number of dual variables in (4.22) and is equal to the product of the number of upper-triangular elements in each $P_\gamma \in \mathbb{S}^n$ (the coefficients in $P(\alpha)$) and the number of monomials in $P(\alpha)$ (i.e. the cardinality of W_{d_p}). Since there are $f(l, d_p) = \binom{d_p + l - 1}{l - 1}$ coefficients in $P(\alpha)$ and each coefficient has $\tilde{N} := \frac{1}{2}n(n+1)$ upper-triangular elements, we find K as

$$K = \frac{(d_p + l - 1)!}{d_p!(l-1)!}\tilde{N}. \tag{4.26}$$

To define the $B_{i,j}$ blocks, first we define the map $V_{\langle h \rangle} : \mathbb{Z}^K \to \mathbb{Z}^{n \times n}$,

$$V_{\langle h \rangle}(x) := \sum_{j=1}^{\tilde{N}} E_j \; x_{j + \tilde{N}(\langle h \rangle - 1)} \quad \text{for all} \quad h \in W_{d_p}, \tag{4.27}$$

which maps each variable to E_j, where E_j, $j = 1, \cdots \tilde{N}$ define the canonical basis for $\mathbb{S}^n$ (subspace of symmetric matrices) as follows.

$$[E_j]_{i,k} := \begin{cases} 1 & \text{if } i = k = j \\ 0 & \text{otherwise} \end{cases}, \quad \text{for } j \leq n \quad \text{and} \tag{4.28}$$

$$[E_j]_{i,k} := [F_j]_{i,k} + [F_j]_{i,k}^T, \quad \text{for } j > n, \tag{4.29}$$

where

$$[F_j]_{i,k} := \begin{cases} 1 & \text{if } i = k - 1 = j - n \\ 0 & \text{otherwise.} \end{cases} \tag{4.30}$$

Note that a different choice of basis would require a different function $V_{\langle h \rangle}$. Then, for $i = 1, \cdots, K$, we define $B_{i,j}$ matrices as

$$B_{i,j} := \begin{cases} \sum_{h \in W_{d_p}} \beta_{\langle h \rangle, j} V_{\langle h \rangle}(e_i), & \text{for } 1 \leq j \leq L \qquad (I) \\ -\sum_{h \in W_{d_p}} \left(H_{\langle h \rangle, j-L}^T V_{\langle h \rangle}(e_i) + V_{\langle h \rangle}(e_i) H_{\langle h \rangle, j-L} \right), & \text{for } L+1 \leq j \leq L+M, \quad (II) \end{cases} \tag{4.31}$$

where we have denoted the canonical basis for $\mathbb{R}^n$ by $e_i = [0 \dots 0 \overbrace{1}^{i^{th}} 0 \dots 0]$, $i = 1, \cdots, n$. Finally, to complete the SDP problem associated with Polya's algorithm, we choose a as

$$a = \vec{1} \in \mathbb{R}^K. \tag{4.32}$$

4.3.4 A parallel algorithm for setting-up the SDP

In this section, we propose a decentralized, iterative algorithm for calculating the terms $\{\beta_{\langle h \rangle, \langle \gamma \rangle}\}$, $\{H_{\langle h \rangle, \langle \gamma \rangle}\}$, C and B_i as defined in (4.15), (4.17), (4.23) and (4.25). We have provided an MPI implementation of this algorithm in C++. The source code is available at `https://www.sites.google.com/a/asu.edu/kamyar/Software`. In Algorithm 5, we have presented a pseudo-code for this algorithm, wherein N is the number of available processors.

4.4 Complexity analysis of the set-up algorithm

Since verifying the positive definiteness of all representatives of a square matrix with entries on proper real intervals is intractable (Nemirovskii (1993)), the question of feasibility of (5.8) is also intractable. To solve the problem of inherent intractability we establish a trade off between accuracy and complexity. In fact, we develop a sequence of decentralized polynomial-time algorithms whose solutions converge to the exact solution of the NP-hard problem. In other words, the translation of a polynomial optimization problem to an LMI problem is the main source of complexity. This high complexity is unavoidable and, in fact, is the reason we seek parallel algorithms. Algorithm 5 distributes the computation and storage of coefficients $\{\beta_{\langle h \rangle, \langle \gamma \rangle}\}$ and $\{H_{\langle h \rangle, \langle \gamma \rangle}\}$ among the processors and their dedicated memories, respectively. In an ideal case, where the number of available processors is sufficiently large (equal to the

Inputs:

d_p: degree of $P(\alpha)$, d_a: degree of $A(\alpha)$, n: number of states, l: number of uncertain parameters, d_1, d_2: number of Polya's iterations, Coefficients of $A(\alpha)$.

Initialization:

Set $\hat{d}_1 = \hat{d}_2 = 0$ and $d_{pa} = d_p + d_a$.

Calculate L_0 as the number of monomials in $P(\alpha)$ using (4.18). Set $L = L_0$.

Calculate M as the number of monomials in $P(\alpha)A(\alpha)$ using (4.20).

Calculate $L' = \texttt{floor}(\frac{L}{N})$ as the number of monomials in $P(\alpha)$ assigned to each processor.

Calculate $M' = \texttt{floor}(\frac{M}{N})$ as the number of monomials in $P(\alpha)A(\alpha)$ assigned to each processor.

for $i = 1, \cdots, N$, *processor* i **do**
    Initialize $\beta_{k,j}$ for $j = (i-1)L' + 1, \cdots, iL'$ and $k = 1, \cdots L_0$ using (4.14).
    Initialize $H_{k,m}$ for $m = (i-1)M' + 1, \cdots, iM'$ & $k = 1, \cdots L_0$ using (4.16).
end

Calculating β and H coefficients:

while $\hat{d}_1 \le d_1$ *or* $\hat{d}_2 \le d_2$ **do**
    if $\hat{d}_1 \le d_1$ **then**
        for $i = 1, \cdots, N$, *processor* i **do**
            Set $d_p = d_p + 1$. Set $\hat{d}_1 = \hat{d}_1 + 1$.
            Update L using (4.19). Update L' as $L' = \texttt{floor}(\frac{L}{N})$.
            Calculate $\beta_{k,j}$, $j = (i-1)L' + 1, \cdots, iL'$ & $k = 1, \cdots L_0$ using (4.15).
        end
    end
    if $\hat{d}_2 \le d_2$ **then**
        for $i = 1, \cdots, N$, *processor* i **do**
            Set $d_{pa} = d_{pa} + 1$ and $\hat{d}_2 = \hat{d}_2 + 1$.
            Update M using (4.20). Update M' as $M' = \texttt{floor}(\frac{M}{N})$.
            Calculate $H_{k,m}$ for $m = (i-1)M' + 1, \cdots, iM'$ and $k = 1, \cdots L_0$.
            using (4.17).
        end
    end
end

***Calculating the SDP elements*:**

for $i = 1, \cdots, N$, *processor* i **do**

Calculate the number of dual variables K using (5.24).

Set $T' = \texttt{floor}(\frac{L+M}{N})$.

Calculate the blocks of the SDP element C as

$$\begin{cases} C_j \text{ using (4.24)} & \text{for } j = (i-1)L' + 1, \cdots, iL' \\ C_j = 0_n & \text{for } j = L + (i-1)M' + 1, \cdots, L + iM'. \end{cases}$$

Set the sub-blocks of the SDP element C as

$$\overline{\mathbf{C}}_i = \operatorname{diag}\left(C_{(i-1)T'+1}, \cdots, C_{iT'}\right). \tag{4.33}$$

for $j = 1, \cdots, K$ **do**

Calculate the blocks of the SDP elements B_j as

$$\begin{cases} B_{j,k} \text{ using (5.25)-}I & \text{for } k = (i-1)L' + 1, \cdots, iL' \\ B_{j,k} \text{ using (5.25)-}II & \text{for } k = L + (i-1)M' + 1, \cdots, L + iM'. \end{cases}$$

Set the sub-blocks of the SDP element B_j as

$$\overline{\mathbf{B}}_{j,i} = \operatorname{diag}\left(B_{j,(i-1)T'+1}, \cdots, B_{j,iT'}\right). \tag{4.34}$$

end

end

***Outputs*:**

Sub-blocks $\overline{\mathbf{C}}_i$ and $\overline{\mathbf{B}}_{j,i}$ of the SDP elements for $i = 1, \cdots, N$ and $j = 1, \cdots, K$.

Algorithm 5: A parallel set-up algorithm for robust stability analysis over the standard simplex number of monomials in $P(\alpha)A(\alpha)$, i.e. M) only one monomial (that corresponds to L_0 of coefficients $\beta_{\langle h\rangle,\langle\gamma\rangle}$ and L_0 of coefficients $H_{\langle h\rangle,\langle\gamma\rangle}$) is assigned to each processor.

4.4.1 Computational complexity analysis

The most computationally expensive part of the set-up algorithm is the calculation of the $B_{i,j}$ blocks in (5.25). Considering that the cost of matrix-matrix multiplication is $\sim n^3$, the cost of calculating each $B_{i,j}$ block is $\sim \text{card}(W_{d_p}) \cdot n^3$. According to (4.25) and (5.25), the total number of $B_{i,j}$ blocks is $K(L+M)$. Hence, as per Algorithm 5, each processor processes $K\left(\texttt{floor}(\frac{L}{N}) + \texttt{floor}(\frac{M}{N})\right)$ of the $B_{i,j}$ blocks, where N is the number of available processors. Therefore, the per processor computational cost of calculating the $B_{i,j}$ at each Polya's iteration is

$$\sim \text{card}(W_{d_p}) \cdot n^3 \cdot K \left(\texttt{floor}\left(\frac{L}{N}\right) + \texttt{floor}\left(\frac{M}{N}\right)\right). \tag{4.35}$$

By substituting for K from (5.24), $\text{card}(W_{d_p})$ from (4.18), L from (4.19) and M from (4.20), the per processor computation cost at each iteration is

$$\sim \left(\frac{(d_p+l-1)!}{d_p!(l-1)!}\right)^2 \frac{n^4(n+1)}{2} \left(\texttt{floor}\left(\frac{\frac{(d_p+d_1+l-1)!}{(d_p+d_1)!(l-1)!}}{N}\right) + \texttt{floor}\left(\frac{\frac{(d_{pa}+d_2+l-1)!}{(d_{pa}+d_2)!(l-1)!}}{N}\right)\right),$$

assuming that $l > 0$ and $N \leq M$, i.e., the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_2} P(\alpha)A(\alpha)$ is at least as large as the number of available processors. Under the assumption that the dynamical systems has large numbers of states and uncertain parameters (large n and l), Table (4.1) presents the computational cost per processor of each Polya's iteration for three different numbers of available processors. For the case where $d_p \geq 3$, the number of operations grows more slowly in n than in l.

Number of processors	L_0	L	M
Computational cost per processor	$\sim (l^{2d_p+d_1} + l^{2d_p+d_a+d_2})n^5$	$\sim (l^{2d_p+d_1} + l^{2d_p+d_a+d_2})n^5$	$\sim l^{2d_p+d_a+d_2-d_1}n^5$

Table 4.1: Per processor, per iteration computational complexity of the set-up algorithm. L_0 is the number of monomials is $P(\alpha)$; L is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_1} P(\alpha)$; M is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_2} P(\alpha)A(\alpha)$.

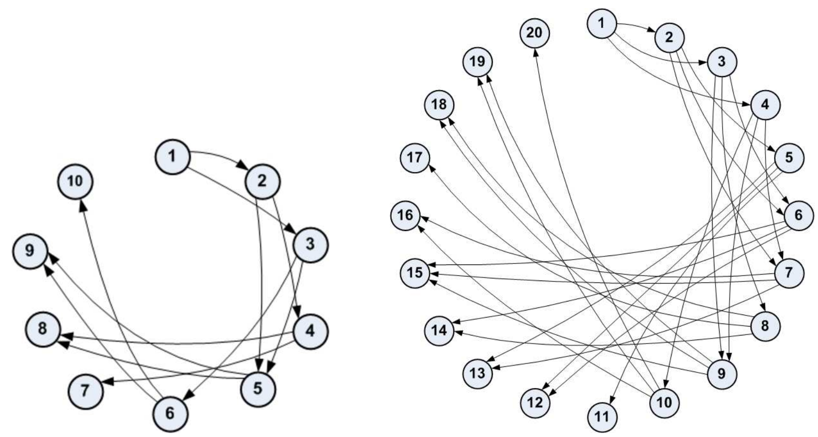
4.4.2 Communication complexity analysis

Communication between processors can be modeled by a directed graph $G(V, E)$, where the set of nodes $V = \{1, \cdots, N\}$ is the set of indices of the available processors and the set of edges $E = \{(i, j) : i, j \in V\}$ is the set of all pairs of processors that communicate with each other. For every directed graph, we can define an adjacency matrix T_G as follows. If processor i communicates with processor j, then $[T_G]_{i,j} = 1$, otherwise $[T_G]_{i,j} = 0$. Here we only define the adjacency matrix for the part of the algorithm that performs Polya's iterations on $P(\alpha)$. For Polya's iterations on $P(\alpha)A(\alpha)$, the adjacency matrix can be defined in a similar manner. For simplicity, we assume that at each iteration, the number of available processors is equal to the number of monomials in $(\sum_{i=1}^{l} \alpha_i)^{d_1} P(\alpha)$. Using (4.19), let us define r_{d_1} and r_{d_1+1} as the numbers of monomials in $(\sum_{i=1}^{l} \alpha_i)^{d_1} P(\alpha)$ and $(\sum_{i=1}^{l} \alpha_i)^{d_1+1} P(\alpha)$. For $I = 1, \cdots, r_{d_1}$, define

$$\mathcal{E}_I := \{\text{lexicographical indices of monomials in } \left(\sum_{i=1}^{l} \alpha_i\right) \alpha^\gamma : \gamma \in W_{d_p+d_1} \text{ and } \langle \gamma \rangle = I\}.$$

Then, for $i = 1, \cdots, r_{d_1+1}$ and $j = 1, \cdots, r_{d_1+1}$, the adjacency matrix of the communication graph is

$$[T_G]_{i,j} := \begin{cases} 1 & \text{if} \quad i \leq r_{d_1} \text{ and } j \in \mathcal{E}_i \text{ and } i \neq j \\ 0 & \text{otherwise.} \end{cases}$$



(a) (b)

Figure 4.5: Graph representation of the network communication of the set-up algorithm. (a) Communication directed graph for the case $\alpha \in \Delta^3, d_p = 2$. (b) Communication directed graph for the case $\alpha \in \Delta^4, d_p = 2$.

Note that this definition implies that the communication graph of the set-up algorithm changes at every iteration. To help visualize the graph, the adjacency matrix for the case where $\alpha \in \Delta^2$ is

$$T_G := \left[\begin{array}{cccccc|ccc} 0 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & & & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \ddots & 0 & & & \\ 0 & 0 & \cdots & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \hline 0 & & & \cdots & & 0 & 0 & \cdots & 0 \\ \vdots & & & \ddots & & \vdots & \vdots & \ddots & \vdots \\ 0 & & & \cdots & & 0 & 0 & \cdots & 0 \end{array}\right] \in \mathbb{R}^{r_{d_1+1} \times r_{d_1+1}},$$

where the nonzero sub-block of T_G lies in $\mathbb{R}^{r_{d_1} \times r_{d_1}}$. We can also illustrate the communication graphs for the cases $\alpha \in \Delta^3$ and $\alpha \in \Delta^4$ with $d_p = 2$ as seen in Figure 4.5.

For a given algorithm, the communication complexity is defined as the sum of the size of all communicated messages. For simplicity, let us consider the worst case

Number of processors	L_0	L	M
communication cost per processor	$\sim l^{d_{pa}+d_2} n^2$	$\sim l^{d_{pa}+d_2-d_1} n^2$	$\sim l^{d_p} n^2$

Table 4.2: Per processor, per iteration communication complexity of the set-up algorithm. L_0 is the number of monomials is $P(\alpha)$; L is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_1} P(\alpha)$; M is the number of monomials in $\left(\sum_{i=1}^{l} \alpha_i\right)^{d_2} P(\alpha)A(\alpha)$.

scenario, where each processor is assigned more than one monomial and sends all of its assigned coefficients $\beta_{\langle h \rangle, \langle \gamma \rangle}$ and $H_{\langle h \rangle, \langle \gamma \rangle}$ to other processors. In this case, the algorithm assigns $\texttt{floor}(\frac{L}{N}) \cdot \text{card}(W_{d_p})$ of coefficients $\beta_{\langle h \rangle, \langle \gamma \rangle}$, each of size 1, and $\big(\texttt{floor}(\frac{L}{N}) + \texttt{floor}(\frac{M}{N})\big) \cdot \text{card}(W_{d_p})$ of coefficients $H_{\langle h \rangle, \langle \gamma \rangle}$, each of size n^2, to each processor. Thus, the communication complexity of the algorithm per processor and per iteration becomes

$$\text{card}(W_{d_p}) \left(\texttt{floor} \left(\frac{L}{N} \right) + \texttt{floor} \left(\frac{M}{N} \right) n^2 \right). \tag{4.36}$$

This indicates that increasing the number of processors (up to M) actually leads to less communication overhead per processor and improves the scalability of the algorithm. By substituting for $\text{card}(W_{d_p})$ from (4.18), L from (4.19) and M from (4.20) and considering large l and n, the communication complexity per processor of each Polya's iteration can be represented as in Table 4.2.

4.5 A parallel SDP solver

Current state-of-the-art interior-point algorithms for solving linear and semi-definite programs are: dual-scaling, primal-dual, cutting-plane/spectral bundle method. Although we found it possible to use dual-scaling algorithms, we chose to pursue a central-path following primal-dual algorithm. One reason that we prefer primal-dual algorithms is because in general, primal-dual algorithms converge faster than dual-scaling algorithms. This assertion is motivated by experience as well as bounds on the convergence rate, such as those found in the literature (Helmberg *et al.* (1996); Benson *et al.* (2000)). More importantly, we prefer primal-dual algorithms because they have the property of preserving the structure (see (4.47)) of the solution at each iteration. We will elaborate on this property in Theorem 15.

We prefer primal-dual algorithms over cutting plane/spectral bundle algorithms because, as we show in Section 4.6, the centralized part of our primal-dual algorithm consists of solving a symmetric system of linear equations (see (4.61)), whereas for the cutting plane/spectral bundle algorithm, the centralized computation would consist of solving a constrained quadratic program (see Sivaramakrishnan (2010), Nayakkankuppam (2007)) with the number of variables equal to the size of the system of linear equations. Because centralized computation is a limiting factor in a parallel algorithm (Amdahl's law), and because solving symmetric linear equations is simpler than solving a quadratic programming problem, we chose the primal-dual approach.

The choice of a central path-following primal-dual algorithm as in Helmberg *et al.* (1996) and F. Alizadeh (1994) was motivated by results in Alizadeh *et al.* (1998) which demonstrated better convergence, accuracy and robustness over the other types of primal-dual algorithms. More specifically, we chose the approach in Helmberg *et al.* (1996) over F. Alizadeh (1994) because unlike the *Schur Complement Matrix* (SCM) approach of the algorithm in F. Alizadeh (1994), the SCM of Helmberg *et al.* (1996) is symmetric and only the upper-triangular elements need to be sent/received by the processors. This leads to less communication overhead. The other reason for choosing Helmberg *et al.* (1996) is that the symmetric SCM of the algorithm in Helmberg *et al.* (1996) can be factorized using Cholesky factorization, whereas the non-symmetric SCM of F. Alizadeh (1994) must be factorized by LU factorization (LU factorization is roughly twice as expensive as Cholesky factorization). Since factorization of SCM comprises the main portion of the centralized computation in our algorithm, it is crucial for us to use computationally-cheaper factorization methods to achieve a better scalability.

Recall from Section 3.5 that in the primal-dual algorithm, both primal and dual problems are solved by iteratively calculating primal and dual search directions and step sizes, and applying these to the primal and dual variables. Let X be the primal variable and y and Z be the dual variables. At each iteration, the variables are updated as

$$X_{k+1} = X_k + t_p \Delta X \tag{4.37}$$

$$y_{k+1} = y_k + t_d \Delta y \tag{4.38}$$

$$Z_{k+1} = Z_k + t_d \Delta Z, \tag{4.39}$$

where ΔX, Δy, and ΔZ are the search directions defined in (3.34) and t_p and t_d are primal and dual step sizes. For the SDPs associated with Polya's theorem (see (4.21) and (4.22)), because the map G (defined in (3.18)) is zero, the predictor search directions defined in (3.28)-(3.30) reduce to the following.

$$\Delta \widehat{y} = \Omega^{-1}\left(-a + B(Z^{-1}GX)\right) \tag{4.40}$$

$$\Delta \widehat{X} = -X + Z^{-1}G\left(\sum_{i=1}^{K} B_i \Delta \widehat{y}_i\right) X \tag{4.41}$$

$$\Delta \widehat{Z} = \left(\sum_{i=1}^{K} B_i y_i\right) - Z - C + \left(\sum_{i=1}^{K} B_i \Delta \widehat{y}_i\right), \tag{4.42}$$

where

$$G = -\sum_{i=1}^{K} B_i y_i + Z + C, \tag{4.43}$$

and $\Omega = [B(Z^{-1}B_1X) \ \cdots \ B(Z^{-1}B_KX)]$. Similarly, the corrector search directions defined in (3.31)-(3.33) reduce to

$$\Delta\overline{y} = \Omega^{-1}\left(B(\mu Z^{-1}) - B(Z^{-1}\Delta\widehat{Z}\Delta\widehat{X})\right) \tag{4.44}$$

$$\Delta\overline{X} = \mu Z^{-1} - Z^{-1}\Delta\widehat{Z}\Delta\widehat{X} - Z^{-1}\Delta\overline{Z}X \tag{4.45}$$

$$\Delta\overline{Z} = \sum_{i=1}^{K} B_i\Delta\overline{y}_i. \tag{4.46}$$

In the following section, we discuss the structure of the decision variables of the SDP defined by the Elements (4.23), (4.25) and (4.32).

4.5.1 Structure of the SDP variables

The key algorithmic insight of this study which allows us to use the primal-dual approach presented in Algorithm 4 is that by choosing an initial value for the primal variable with a certain block structure corresponding to the distributed structure of the processors, the algorithm will preserve this structure on the primal and dual variables at every iteration. Specifically, we define the following structured block-diagonal subspace, where each block corresponds to a single processor.

$$S_{l,m,n} := \left\{Y \subset \mathbb{R}^{(l+m)n\times(l+m)n} : Y = \operatorname{diag}(Y_1, \cdots Y_l, Y_{l+1}, \cdots Y_{l+m}) \text{ for } Y_i \in \mathbb{R}^{n\times n}\right\} \tag{4.47}$$

According to the following theorem, the subspace $S_{l,m,n}$ is invariant under the predictor and corrector iterations in the sense that when Algorithm 4 is applied to the SDP problem defined by the Elements (4.23), (4.25) and (4.32) with a primal starting point $X_0 \in S_{l,m,n}$, then the primal and dual variables remain in the subspace at every iteration.

Theorem 15. *Consider the SDP problem defined in* (4.21) *and* (4.22) *with elements given by* (4.23), (4.25) *and* (4.32). *Suppose L and M are the cardinalities of $W_{d_p+d_1}$ and $W_{d_{pa}+d_2}$ as defined in* (4.19) *and* (4.20). *If* (4.37), (4.38) *and* (4.39) *are initialized* *by*

$$X_0 \in S_{L,M,n}, \quad y_0 \in \mathbb{R}^K, \quad Z_0 \in S_{L,M,n},$$

then for all $k \in \mathbb{N}$*,*

$$X_k \in S_{L,M,n}, \quad Z_k \in S_{L,M,n}.$$

Proof. We proceed by induction. First, suppose for some $k \in \mathbb{N}$,

$$X_k \in S_{L,M,n} \quad \text{and} \quad Z_k \in S_{L,M,n}. \tag{4.48}$$

We would like to show that this implies $X_{k+1}, Z_{k+1} \in S_{L,M,n}$. To see this, observe that according to (4.37), $X_{k+1} = X_k + t_p \Delta X_k$ for all $k \in \mathbb{N}$. From (3.34), ΔX_k can be written as

$$\Delta X_k = \Delta \widehat{X}_k + \Delta \overline{X}_k \quad \text{for all } k \in \mathbb{N}. \tag{4.49}$$

To find the structure of ΔX_k, we focus on the structures of $\Delta \widehat{X}_k$ and $\Delta \overline{X}_k$ individually. Using (4.41), $\Delta \widehat{X}_k$ is

$$\Delta \widehat{X}_k = -X_k + Z_k^{-1} G_k \left(\sum_{i=1}^{K} B_i \Delta \widehat{y}_k \right) X_k \quad \text{for all } k \in \mathbb{N}, \tag{4.50}$$

where according to (4.43), G_k is

$$G_k = C - \sum_{i=1}^{K} B_i y_i + Z_k \quad \text{for all } k \in \mathbb{N}. \tag{4.51}$$

First, we examine the structure of G_k. According to the definition of C and B_i in (4.23) and (4.25), we know that

$$C \in S_{L,M,n} \quad \text{and} \quad \sum_{i=1}^{K} B_i y_i \in S_{L,M,n} \quad \text{for any } y \in \mathbb{R}^K. \tag{4.52}$$

Since all the terms on the right hand side of (4.51) are in $S_{L,M,n}$ and $S_{L,M,n}$ is a subspace, we conclude

$$G_k \in S_{L,M,n}. \tag{4.53}$$

Returning to (4.50), using our assumption in (4.48) and noting that the structure of the matrices in $S_{L,M,n}$ is also preserved through multiplication and inversion, we conclude

$$\Delta\widehat{X}_k \in S_{L,M,n}. \tag{4.54}$$

According to (4.45), the second term in (4.49) is

$$\Delta\overline{X}_k = \mu Z_k^{-1} - Z_k^{-1}\Delta\widehat{Z}_k\Delta\widehat{X}_k - Z_k^{-1}\Delta\overline{Z}_k X_k \quad \text{for all } k \in \mathbb{N}. \tag{4.55}$$

To determine the structure of $\Delta\overline{X}_k$, first we investigate the structure of $\Delta\widehat{Z}_k$ and $\Delta\overline{Z}_k$. According to (4.42) and (4.46) we have

$$\Delta\widehat{Z}_k = \sum_{i=1}^{K} B_i y_{k_i} - Z_k - C + \sum_{i=1}^{K} B_i \Delta\widehat{y}_{k_i} \qquad \text{for all } k \in \mathbb{N} \tag{4.56}$$

$$\Delta\overline{Z}_k = \sum_{i=1}^{K} B_i \Delta\overline{y}_{k_i} \qquad \text{for all } k \in \mathbb{N}. \tag{4.57}$$

Because all the terms in the right hand side of (4.56) and (4.57) are in $S_{L,M,n}$, it follows that

$$\Delta\widehat{Z}_k \in S_{L,M,n}, \quad \Delta\overline{Z}_k \in S_{L,M,n}. \tag{4.58}$$

Recalling (4.54), (4.55) and our assumption in (4.48), we have

$$\Delta\overline{X}_k \in S_{L,M,n}. \tag{4.59}$$

According to (4.54), (4.58) and (4.59), the total step directions are in $S_{L,M,n}$,

$$\Delta X_k = \Delta\widehat{X}_k + \Delta\overline{X}_k \in S_{L,M,n}$$

$$\Delta Z_k = \Delta\widehat{Z}_k + \Delta\overline{Z}_k \in S_{L,M,n},$$

and it follows that

$$X_{k+1} = X_k + t_p \Delta X_k \in S_{L,M,n}$$

$$Z_{k+1} = Z_k + t_p \Delta Z_k \in S_{L,M,n}.$$

Thus, for any $y \in \mathbb{R}^K$ and $k \in \mathbb{N}$, if $X_k, Z_k \in S_{L,M,n}$, we have $X_{k+1}, Z_{k+1} \in S_{L,M,n}$. Since we have assumed that the initial values $X_0, Z_0 \in S_{L,M,n}$, we conclude by induction that $X_k \in S_{L,M,n}$ and $Z_k \in S_{L,M,n}$ for all $k \in \mathbb{N}$. $\square$

4.5.2 A parallel implementation for the SDP solver

In this section, we propose a parallel algorithm for solving the SDP problems associated with Polya's algorithm. In particular, we show how to map the block-diagonal structure of the primal variable and the primal-dual search directions described in Section 4.5 to a parallel computing structure consisting of a central root processor with N slave processors. Note that processor steps are simultaneous and transitions between root and processor steps are synchronous. Processors are idle when root is active and vice-versa. A C++ implementation of this algorithm using MPI and numerical linear algebra libraries CBLAS and CLAPACK is provided at: `www.sites.google.com/a/asu.edu/kamyar/Software`. Let N be the number of available processors and $J := \texttt{floor}\left(\frac{L+M}{N}\right)$. As per Algorithm 6, we assume processor i has access to the sub-blocks $\overline{\mathbf{C}}_i$ and $\overline{\mathbf{B}}_{j,i}$ defined in (4.33) and (4.34) for $j = 1, \cdots, K$. Be aware that minor parts of Algorithm 6 have been abridged in order to simplify the presentation.

4.6 Computational complexity analysis of the SDP algorithm

Complexity theory for parallel computation has been studied in some depth (Greenlaw *et al.* (1995)). The class NC $\subset$ P is often considered to be the class of problems that can be parallelized efficiently. More precisely, a problem is in NC if there exist integers c and d such that the problem can be solved in $\mathcal{O}(\log(n)^c)$ steps using $\mathcal{O}(n^d)$ processors. On the other hand, the class P-complete is a class of problems which are equivalent up to an NC reduction, but contains no problem in NC and

Inputs: $\overline{\mathbf{C}}_i, \overline{\mathbf{B}}_{j,i}$ for $i = 1, \cdots, N$ and $j = 1, \cdots, K$: the sub-blocks of the SDP elements provided to processor i by the set-up algorithm; Stopping criterion ϵ.

Processors Initialization step:

for $i = 1, \cdots, N$, *processor* i **do**

Initialize primal and dual variables $\mathbf{X}_i^0$, $\mathbf{Z}_i^0$ and y^0 as

$$\mathbf{X}_i^0 = \begin{cases} I_{(J+1)n}, & 0 \leq i < L + M - NJ \\ I_{Jn}, & L + M - NJ \leq i < N, \end{cases},$$

$$\mathbf{Z}_i^0 = \mathbf{X}_i^0 \quad \text{and} \quad y^0 = \vec{0} \in \mathbb{R}^K,$$

Calculate the complementary slackness as $S_i = tr(\mathbf{Z}_i^0 \mathbf{X}_i^0)$.

Send S_i to the processor root.

end

Root Initialization step:

Root processor **do**

Calculate the barrier parameter as $\mu = \frac{1}{3} \sum_{i=1}^{N} S_i$.

Set the SDP element $a = \vec{1} \in \mathbb{R}^K$.

Processors step 1:

for $i = 1, \cdots, N$, *processor* i **do**

for $k = 1, \cdots, K$ **do**

Calculate the elements of Ω_1 (R-H-S of System (4.61))

$$\omega_{i,k} = \mathrm{tr}\left(\overline{\mathbf{B}}_{k,i}(\mathbf{Z}_i)^{-1}\left(-\sum_{j=1}^{K} y_j \overline{\mathbf{B}}_{j,i} + \mathbf{Z}_i + \overline{\mathbf{C}}_i\right)\mathbf{X}_i\right)$$

for $l = 1, \cdots, K$ **do**

Calculate the elements of the SCM as

$$\lambda_{i,k,l} = \mathrm{tr}\left(\overline{\mathbf{B}}_{k,i}(\mathbf{Z}_i)^{-1}\overline{\mathbf{B}}_{l,i}\mathbf{X}_i\right) \tag{4.60}$$

end

end

Send $\omega_{i,k}$ and $\lambda_{i,k,l}$, $k = 1, \cdots, K$ and $l = 1, \cdots, K$ to the root processor.

end

Root step 1:

Root processor **do**

Construct the R-H-S of System (4.61) and the SCM as

$$\Omega_1 = \begin{pmatrix} \sum_{i=1}^N \omega_{i,1} \\ \sum_{i=1}^N \omega_{i,2} \\ \vdots \\ \sum_{i=1}^N \omega_{i,K} \end{pmatrix} - a \quad \text{and } \Lambda = \left[\begin{pmatrix} \sum_{i=1}^N \lambda_{i,1,1} \\ \sum_{i=1}^N \lambda_{i,2,1} \\ \vdots \\ \sum_{i=1}^N \lambda_{i,K,1} \end{pmatrix}, \cdots, \begin{pmatrix} \sum_{i=1}^N \lambda_{i,1,K} \\ \sum_{i=1}^N \lambda_{i,2,K} \\ \vdots \\ \sum_{i=1}^N \lambda_{i,K,K} \end{pmatrix} \right]$$

Solve the following system of equations for the predictor dual step $\Delta\widehat{y} \in \mathbb{R}^K$.

$$\Lambda \Delta \widehat{y} = \Omega_1 \tag{4.61}$$

Send $\Delta\widehat{y}$ to all processors.

Processors step 2:

for $i = 1, \cdots, N$, *processor* i **do**

Calculate the predictor step directions

$$\Delta\widehat{\mathbf{X}}_i = -\mathbf{X}_i + (\mathbf{Z}_i)^{-1} \left(-\sum_{j=1}^K y_j \overline{\mathbf{B}}_{j,i} + \mathbf{Z}_i + \overline{\mathbf{C}}_i \right) \sum_{j=1}^K \Delta\widehat{y}_j \, \overline{\mathbf{B}}_{j,i} \, \mathbf{X}_i,$$

$$\Delta\widehat{\mathbf{Z}}_i = \sum_{j=1}^K y_j \overline{\mathbf{B}}_{j,i} - \mathbf{Z}_i - \overline{\mathbf{C}}_i + \sum_{j=1}^K \Delta\widehat{y}_j \overline{\mathbf{B}}_{j,i}.$$

for $k = 1, \cdots, K$ **do**

Calculate the elements of Ω_2 (R-H-S of (4.62))

$$\delta_{i,k} = \mathrm{tr}(\overline{\mathbf{B}}_{k,i}(\mathbf{Z}_i)^{-1}), \tau_{i,k} = tr(\overline{\mathbf{B}}_{k,i}(\mathbf{Z}_i)^{-1}\Delta\widehat{\mathbf{Z}}_i\Delta\widehat{\mathbf{X}}_i)$$

end

Send $\delta_{i,k}$ and $\tau_{i,k}$, $k = 1, \cdots, K$ to the root processor.

end

Root step 2:

Construct the R-H-S of (4.62) as

$$\Omega_2 = \mu \begin{bmatrix} \sum_{i=1}^N \delta_{i,1} & \sum_{i=1}^N \delta_{i,2} & \cdots & \sum_{i=1}^N \delta_{i,K} \end{bmatrix}^T - \begin{bmatrix} \sum_{i=1}^N \tau_{i,1} & \sum_{i=1}^N \tau_{i,2} & \cdots & \sum_{i=1}^N \tau_{i,K} \end{bmatrix}^T$$

Solve the following system of equations for the corrector dual variable $\Delta\overline{y}$.

$$\Lambda \Delta \overline{y} = \Omega_2 \tag{4.62}$$

Send $\Delta\overline{y}$ to all processors.

Processors step 3:

for $i = 1, \cdots, N$, *processor* i **do**

Calculate the corrector step directions as follows.

$$\Delta\overline{\mathbf{Z}}_i = \sum_{j=1}^{K} \Delta\overline{y}_j \overline{\mathbf{B}}_{j,i}$$

$$\Delta\overline{\mathbf{X}}_i = -(\mathbf{Z}_i)^{-1}(\Delta\overline{\mathbf{Z}}_i\mathbf{X}_i + \Delta\widehat{\mathbf{Z}}_i\Delta\widehat{\mathbf{X}}_i) + \mu(\mathbf{Z}_i)^{-1}$$

Calculate the primal and dual total search directions as

$$\Delta\mathbf{X}_i = \Delta\widehat{\mathbf{X}}_i + \Delta\overline{\mathbf{X}}_i, \quad \Delta\mathbf{Z}_i = \Delta\widehat{\mathbf{Z}}_i + \Delta\overline{\mathbf{Z}}_i, \quad \Delta y = \Delta\widehat{y} + \Delta\overline{y}.$$

Set the primal step size t_p and dual step size t_d using an appropriate line search method.

Update the primal and dual variables as

$$\mathbf{X}_i \equiv \mathbf{X}_i + t_p\Delta\mathbf{X}_i, \quad \mathbf{Z}_i \equiv \mathbf{Z}_i + t_d\Delta\mathbf{Z}_i, \quad y \equiv y + t_d\Delta y$$

end

Processors step 4:

for $i = 1, \cdots, N$, *processor* i **do**

Calculate the contribution of X_i to primal cost and complementary slackness as

$$\tilde{\phi}_i = tr\left(\overline{\mathbf{C}}_i\mathbf{X}_i\right) \quad \text{and} \quad S_i = tr\left(\mathbf{Z}_i\mathbf{X}_i\right).$$

Send S_i and $\tilde{\phi}_i$ to the root processor.

end

Root step 4:

Update the barrier parameter as $\mu = \frac{1}{3}\sum_{i=1}^{N} S_i$.

Calculate the primal and dual costs as $\phi = \sum_{i=1}^{N} \tilde{\phi}_i$ and $\psi = a^T y$.

if $|\phi - \psi| > \epsilon$ **then**

go to Processors step 1

end

else

Calculate the coefficients of $P(\alpha)$ as $P_i = \sum_{j=1}^{\tilde{N}} E_j y_{(j+\tilde{N}i-1))}$ for $i = 1, \cdots, L_0$.

end

Output:

Coefficients P_i of a polynomial $P(\alpha)$ such that $P(\alpha) > 0$ for all $\alpha \in \Delta^l$ and satisfies the Lyapunov inequalities in (5.8).

Algorithm 6: A parallel SDP algorithm is thought to be the simplest class of "inherently sequential" problems. It has been proven that Linear Programming (LP) is P-complete Greenlaw *et al.* (1995) and SDP is P-hard (at least as hard as any P-complete problem) and thus is unlikely to admit a general-purpose parallel solution. Given this fact and given the observation that the problem we are trying to solve is NP-hard, it is important to thoroughly understand the complexity of the algorithms we are proposing and how this complexity scales with various parameters which define the size of the stability analysis problem. To better understand these issues, we have broken our complexity analysis down into several cases which should be of interest to the control community. Note that the cases below do not discuss memory complexity. This is because in the cases when a sufficient number of processors are available, for a system with n states, the memory requirements per block are simply proportional to n^2.

4.6.1 Complexity analysis for systems with large number of states

Suppose we are considering a problem with n states. For this case, the most computationally expensive part of the algorithm is the calculation of the Schur complement matrix Λ by the processors in Processors step 1 (and summed by the root in Root step 1, although we neglect this part). In particular, the computational complexity of the algorithm is determined by the number of operations required to calculate (4.60), restated here.

$$\lambda_{i,k,l} = tr\left(\overline{\mathbf{B}}_{k,i}(\mathbf{Z}_i)^{-1}\overline{\mathbf{B}}_{l,i}\mathbf{X}_i\right) \quad \text{for} \quad k = 1, \cdots, K \text{ and } \quad l = 1, \cdots, K.$$

Since the cost of $n \times n$ matrix-matrix multiplication requires $\mathcal{O}(n^3)$ steps and each of $\mathbf{X}_i, \mathbf{Z}_i, \overline{\mathbf{B}}_{l,i}$ has $\mathtt{floor}(\frac{L+M}{N})$ number of blocks in $\mathbb{R}^{n\times n}$, the number of operations performed by the i^{th} processor to calculate $\lambda_{i,k,l}$ for $k = 1, \cdots, K$ and $l = 1, \cdots, K$

is proportional to

$$\begin{cases} \texttt{floor}\left(\dfrac{L+M}{N}\right)K^2n^3 & N < L+M \\ K^2n^3 & N \geq L+M \end{cases} \tag{4.63}$$

at each iteration, where $i = 1, \cdots, N$. By substituting K in (4.63) from (5.24), for $N \geq L+M$, each processor performs

$$\sim \frac{((d_p + l - 1)!)^2}{(d_p!)^2((l-1)!)^2} n^7 \tag{4.64}$$

operations per iteration. Therefore, for systems with large number n of states and fixed degree d_p of $P(\alpha)$ and number l of uncertain parameters, the number of operations per processor required to solve the SDP associated with parameter-dependent feasibility problem $A(\alpha)^T P(\alpha) + P(\alpha)A(\alpha) < 0$, is proportional to n^7. Solving the LMI associated with the parameter-independent problem $\mathrm{A}^T P + P\mathrm{A} < 0$ using our algorithm or most of the SDP solvers such as Sturm (1999); Borchers and Young (2007); Yamashita *et al.* (2003) also requires $O(n^7)$ operations per processor. Therefore, if we have a sufficient number of available processors (at least $L+M$), the proposed algorithm solves both the stability and robust stability problems by performing $O(n^7)$ operations per processor.

4.6.2 Complexity of increasing accuracy/decreasing Conservativeness

We now consider the effect of raising Polya's exponent. Consider the definition of simplex as follows.

$$\tilde{\Delta}^l_r = \left\{ \alpha \in \mathbb{R}^l : \sum_{i=1}^{l} \alpha_i = r, \alpha_i \geqslant 0 \right\}$$

Suppose we now define the accuracy of the algorithm as the largest value of r found by the algorithm (if it exists) such that if the uncertain parameters lie inside the corresponding simplex, the stability of the system is verified. Typically, increasing Polya's exponent d in (4.7) improves the accuracy of the algorithm. If we again only consider Processor step 1, according to (4.64), the number of processor operations is independent of the Polya's exponent d_1 and d_2. Because this part of the algorithm does not vary with Polya's exponent, we look at the root processing requirements associated with solving the systems of equations in (4.61) and (4.62) in Root step 1 using Cholesky factorization. Each of these systems consists of K equations. The computational complexity of Cholesky factorization is $O(K^3)$. Thus, the number of operations performed by the root processor is proportional to

$$K^3 = \frac{((d_p + l - 1)!)^3}{(d_p!)^3((l-1)!)^3} n^6. \tag{4.65}$$

In terms of communication complexity, the most significant operation between the root and other processors is sending and receiving $\lambda_{i,k,l}$ for $i = 1, \cdots, N$, $k = 1, \cdots, K$ and $l = 1, \cdots, K$ in Processors step 1 and Root step 1. Thus, the total communication cost for N processors per iteration is

$$\sim N \cdot K^2 = N \frac{((d_p + l - 1)!)^2}{(d_p!)^2((l-1)!)^2} n^4. \tag{4.66}$$

From (4.64), (4.65) and (4.66) it is observed that the number of processors operations, root operations and communication operations are independent of Polya's exponent d_1 and d_2. Therefore, we conclude that for a fixed d_p and sufficiently large number of processors N ($N \geq L + M$), improving the accuracy by increasing d_1 and d_2 does not add any computation per processor or communication overhead.

4.6.3 Analysis of scalability/speed-up

The speed-up of a parallel algorithm is defined as $SP_N = \dfrac{T_s}{T_N}$, where T_s is the execution time of the algorithm on a single processor and T_N is the execution time of the parallel algorithm using N processors. The speed-up is governed by

$$SP_N = \frac{N}{D + NS}, \tag{4.67}$$

where D is the *decentralization ratio* and is defined as the ratio of the total operations performed by all processors except the root to total operations performed by all processors and root. S is the *centralization ratio* and is defined as the ratio of the operations performed by the root processor to total operations performed by all processors and the root. Suppose that the number of available processors is equal to the number of sub-blocks in C defined in (4.23), i.e, equal to $L+M$. Using the above definitions for D and S, Equation (4.64) as the decentralized computation and (4.65) as the centralized computation, D and S can be approximated as

$$D \simeq \frac{N\dfrac{((d_p+l-1)!)^2}{(d_p!)^2((l-1)!)^2}n^7}{N\dfrac{((d_p+l-1)!)^2}{(d_p!)^2((l-1)!)^2}n^7+\dfrac{((d_p+l-1)!)^3}{(d_p!)^3((l-1)!)^3}n^6} \tag{4.68}$$

and

$$S \simeq \frac{\dfrac{((d_p+l-1)!)^3}{(d_p!)^3((l-1)!)^3}n^6}{N\dfrac{((d_p+l-1)!)^2}{(d_p!)^2((l-1)!)^2}n^7+\dfrac{((d_p+l-1)!)^3}{(d_p!)^3((l-1)!)^3}n^6}. \tag{4.69}$$

According to (4.19) and (4.20) the number of processors $N = L + M$ is independent of n. Therefore,

$$\lim_{n\to\infty} D = 1 \quad \text{and} \quad \lim_{n\to\infty} S = 0.$$

By substituting D and S in (4.67) with their limit values, we have $\lim_{n\to\infty} SP_N = N$. Thus, for large n, by using $L+M$ processors, the presented decentralized algorithm solves large robust stability problems $L+M$ times faster than the sequential algorithms. For different values of the state-space dimension n, the theoretical speed-up of the algorithm versus the number of processors is illustrated in Figure 4.6. As shown in Figure 4.6, for problems with large n, by using $N \leq L+M$ processors the parallel algorithm solves the robust stability problems approximately N times faster than the sequential algorithm. As n increases, the trend of speed-up becomes increasingly

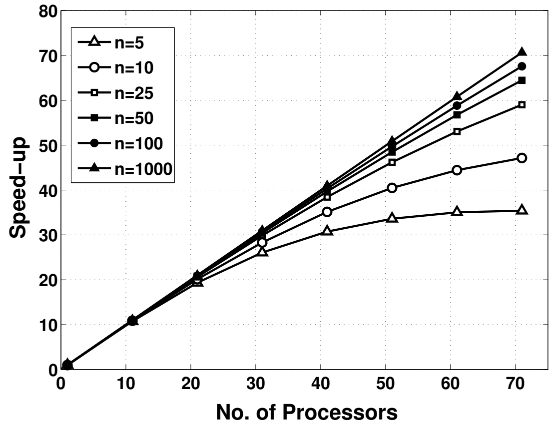


Figure 4.6: Theoretical speed-up vs. No. of processors for different system dimensions n for $l = 10$, $d_p = 2$, $d_a = 3$ and $d_1 = d_2 = 4$, where $L + M = 53625$

linear. Therefore, for problems with a large number of states, our algorithm becomes increasingly efficient in terms of processor utilization.

4.6.4 Synchronization and load balancing analysis

The proposed algorithm is synchronous in that all processors must return values before the centralized step can proceed. However, in the case where we have fewer processors than blocks, some processors may be assigned one block more than other processors. In this case, some processors may remain idle while waiting for the more heavily loaded blocks to complete. In the worst case, this can result in a 50% decrease in the execution speed. We have addressed this issue in the following manner:

1. We allocate almost the same number (± 1) of blocks of the SDP elements C and B_i to all processors, i.e., $\texttt{floor}(\frac{L+M}{N})+1$ blocks to r processors and $\texttt{floor}(\frac{L+M}{N})$ blocks to the other $N-r$ processors, where r is the remainder of dividing $L+M$ by N.

2. We assign the same routine to all of the processors in the Processors steps of Algorithm 6.

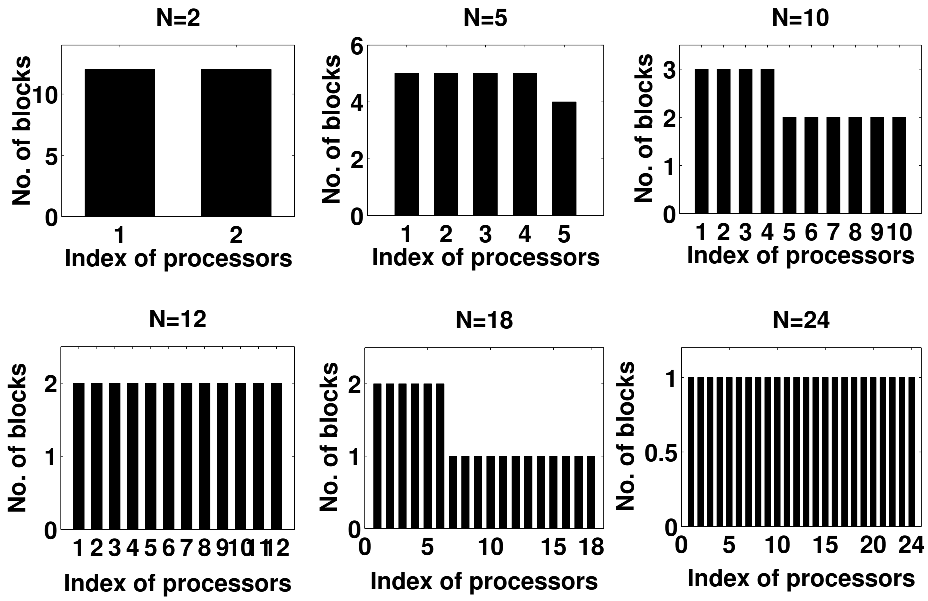


Figure 4.7: The number of blocks of the SDP elements assigned to each processor. An illustration of load balancing.

If $L + M$ is a multiple of N, then the algorithm assigns the same amount of data, i.e., $\frac{L+M}{N}$ blocks of C and B_i to each processor. In this case, the processors are perfectly synchronized. If $L + M$ is not a multiple of N, then according to (4.63), r of the N processors perform K^2n^3 extra operations per iteration. This fraction is $\dfrac{1}{1+\mathtt{floor}(\frac{L+M}{N})} \leq 0.5$ of the operations per iteration performed by each of r processors. Thus in the worst case, we have a 50% reduction, although this situation is rare. As an example, the load balancing (distribution of data and calculation) for the case of solving an SDP of the size $L+M = 24$ using different numbers of available processors N is demonstrated in Figure 4.7. This figure shows the number of blocks that are allocated to each processor. According to this figure, for $N = 2, 12$ and 24, the processors are perfectly balanced, whereas for the case where $N = 18$, twelve processors perform 50% fewer calculations.

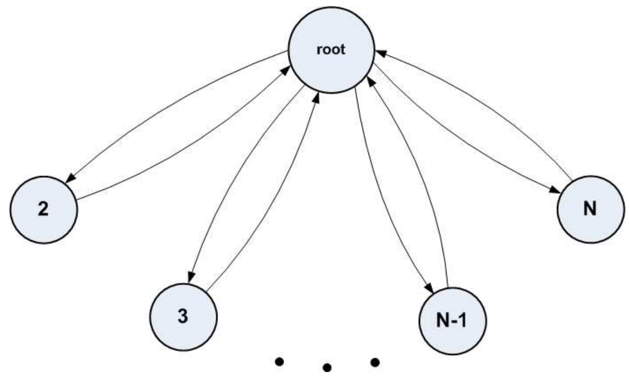


Figure 4.8: The communication graph of the SDP algorithm

4.6.5 Communication graph of the algorithm

The communication directed graph of the SDP algorithm (see Figure 4.8) is static (fixed for all iterations). At each iteration, root sends messages (dual predictor and corrector search directions $\Delta\widehat{y}$ and $\Delta\overline{y}$) to all of the processors and receives messages (elements $\lambda_{i,k,l}$ of the SCM defined in (4.60)) from all of the processors. The adjacency matrix of the communication directed graph is defined as follows. For $i = 1, \cdots, N$ and $j = 1, \cdots, N$,

$$[T_G]_{i,j} := \begin{cases} 1 & \text{if } \big(i = 1 \text{ or } j = 1\big) \text{ and } \big(i \neq j\big) \\ 0 & \text{Otherwise.} \end{cases}$$

4.7 Testing and validation

In this section, we present validation data in 4 key areas. First, we present analysis results for a realistic large-scale model of Tokamak operation using a discretized PDE model. Next, we present accuracy and convergence data and compare our algorithm to the SOS approach. Next, we analyze scalability and speed-up of our algorithm as we increase the number of processors and compare our results to the state-of-the-art general-purpose parallel SDP solver SDPARA. Finally, we explore the limits of the algorithm in terms of the size of the problem, when implemented on a moderately powerful cluster computer and using a moderate processor allocation on the IBM Blue Gene supercomputer at Argonne National Laboratory.

4.7.1 Example 1: Application to control of a discretized PDE model in fusion research

The goal of this example is to use the proposed algorithm to solve a real-world stability problem. A simplified model for the poloidal magnetic flux gradient in a Tokamak reactor (Witrant *et al.* (2007)) is

$$\frac{\partial \psi_x(x,t)}{\partial t} = \frac{1}{\mu_0 a^2} \frac{\partial}{\partial x} \left(\frac{\eta(x)}{x} \frac{\partial}{\partial x} \left(x\psi_x(x,t) \right) \right) \tag{4.70}$$

with Dirichlet boundary conditions $\psi_x(0,t) = 0$ and $\psi_x(1,t) = 0$ for all $t \in \mathbb{R}^+$, where ψ_x is the deviation of the flux gradient from a reference flux gradient profile, μ_0 is the permeability of free space, $\eta(x)$ is the plasma resistivity and a is the radius of the Last Closed Magnetic Surface (LCMS). To obtain the finite-dimensional state-space representation of the PDE, we discretize the PDE in the spatial domain $[0,1]$ at $N = 7$ points. The state-space model is then

$$\dot{\psi}_x(t) = A(\eta(x))\psi_x(t), \tag{4.71}$$

j	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$	$\frac{9}{2}$	$\frac{11}{2}$	$\frac{13}{2}$	$\frac{15}{2}$
x_j	0.036	0.143	0.286	0.429	0.571	0.714	0.857	0.964
$\widehat{\eta}(x_j)$	$1.775e-8$	$2.703e-8$	$5.676e-8$	$1.182e-7$	$2.058e-7$	$3.655e-7$	$1.076e-6$	$8.419e-6$

Table 4.3: Data for Example 1: Nominal values of the plasma resistivity

where $A(\eta(x)) \in \mathbb{R}^{N \times N}$ has the following non-zero entries.

$$
\begin{aligned}
a_{11} &= \frac{-4}{3\mu_0 \Delta x^2 a^2} \left(\frac{\eta(x_{\frac{3}{2}})}{x_{\frac{3}{2}}} + \frac{2\eta(x_{\frac{3}{4}})}{x_{\frac{3}{4}}} \right), \\
a_{12} &= \frac{4}{3\mu_0 \Delta x^2 a^2} \left(\frac{\eta(x_{\frac{3}{2}}) x_2}{x_{\frac{3}{2}}} \right), \\
a_{j,j-1} &= \frac{1}{\Delta x^2 \mu_0 a^2} \left(\frac{\eta(x_{j-\frac{1}{2}})}{x_{j-\frac{1}{2}}} x_{j-1} \right) && \text{for } j = 2, \cdots, N-1, \\
a_{j,j} &= \frac{-1}{\Delta x^2 \mu_0 a^2} \left(\frac{\eta(x_{j+\frac{1}{2}})}{x_{j+\frac{1}{2}}} + \frac{\eta(x_{j-\frac{1}{2}})}{x_{j-\frac{1}{2}}} \right) x_j && \text{for } j = 2, \cdots, N-1, \\
a_{j,j+1} &= \frac{1}{\Delta x^2 \mu_0 a^2} \left(\frac{\eta(x_{j+\frac{1}{2}})}{x_{j+\frac{1}{2}}} x_{j+1} \right) && \text{for } j = 2, \cdots, N-1, \\
a_{N,N-1} &= \frac{4}{3\Delta x \mu_0 a^2} \frac{\eta(x_{N-\frac{1}{2}}) x_{N-1}}{x_{N-\frac{1}{2}} \Delta x}, \\
a_{N,N} &= \frac{-4}{3\Delta x \mu_0 a^2} \left(\frac{2\eta(x_{N+\frac{1}{4}}) x_N}{x_{N+\frac{1}{4}} \Delta x} + \frac{\eta(x_{N-\frac{1}{2}}) x_N}{x_{N-\frac{1}{2}} \Delta x} \right),
\end{aligned}
$$

where $\Delta x = \dfrac{1}{N}$ and $x_j := (j - \frac{1}{2})\Delta x$. Typically $\eta(x_j)$ are not precisely known (they depend on other state variables), so we substitute for $\eta(x_j)$ in (4.71) with $\widehat{\eta}(x_j) + \alpha_k$, where $\widehat{\eta}(x_j)$ are the nominal values of $\eta(x_j)$ and α_k for $k = 1, \cdots, 8$ are the uncertain parameters. The values for $x_1, \cdots, x_j$ and their corresponding values of $\widehat{\eta}(x_j)$ are presented in Table 4.3. Note that we have used data from the Tore Supra reactor to estimate the nominal values $\widehat{\eta}(x_j)$.

The uncertain system is then written as

$$\dot{\psi}_x(t) = A(\alpha)\psi_x(t), \tag{4.72}$$

where A is affine, $A(\alpha) = A_0 + \sum_{i=1}^{8} A_i \alpha_i$, where

$$A_1 = \begin{bmatrix} -14.09 & 6.47 & 0 & 0 & 0 & 0 & 0 \\ 0.41 & -3.54 & 3.83 & 0 & 0 & 0 & 0 \\ 0 & 2.30 & -10.24 & 8.97 & 0 & 0 & 0 \\ 0 & 0 & 6.41 & -21.46 & 16.06 & 0 & 0 \\ 0 & 0 & 0 & 12.49 & -39.71 & 28.91 & 0 \\ 0 & 0 & 0 & 0 & 23.66 & -101.96 & 86.33 \\ 0 & 0 & 0 & 0 & 0 & 97.40 & -1.74e3 \end{bmatrix} \quad A_2 = \begin{bmatrix} -2.86 & 1.66 & 0 & 0 & 0 & 0 & 0 \\ 1.62 & -7.15 & 3.83 & 0 & 0 & 0 & 0 \\ 0 & 2.30 & -10.24 & 8.97 & 0 & 0 & 0 \\ 0 & 0 & 6.41 & -21.46 & 16.06 & 0 & 0 \\ 0 & 0 & 0 & 12.49 & -39.72 & 28.91 & 0 \\ 0 & 0 & 0 & 0 & 23.66 & -101.96 & 86.33 \\ 0 & 0 & 0 & 0 & 0 & 97.40 & -1.74e3 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} -1.25 & 6.47 & 0 & 0 & 0 & 0 & 0 \\ 0.41 & -5.35 & 6.84 & 0 & 0 & 0 & 0 \\ 0 & 4.10 & -13.25 & 8.97 & 0 & 0 & 0 \\ 0 & 0 & 6.41 & -21.46 & 16.06 & 0 & 0 \\ 0 & 0 & 0 & 12.49 & -39.71 & 28.91 & 0 \\ 0 & 0 & 0 & 0 & 23.66 & -101.96 & 86.33 \\ 0 & 0 & 0 & 0 & 0 & 97.40 & -1.74e3 \end{bmatrix} \quad A_4 = \begin{bmatrix} -1.25 & 6.47 & 0 & 0 & 0 & 0 & 0 \\ 0.41 & -3.54 & 3.83 & 0 & 0 & 0 & 0 \\ 0 & 2.30 & -12.25 & 11.77 & 0 & 0 & 0 \\ 0 & 0 & 8.41 & -24.27 & 16.06 & 0 & 0 \\ 0 & 0 & 0 & 12.49 & -39.72 & 28.91 & 0 \\ 0 & 0 & 0 & 0 & 23.66 & -101.96 & 86.33 \\ 0 & 0 & 0 & 0 & 0 & 97.40 & -1.74e3 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} -1.25 & 6.47 & 0 & 0 & 0 & 0 & 0 \\ 0.41 & -3.54 & 3.83 & 0 & 0 & 0 & 0 \\ 0 & 2.30 & -10.24 & 8.97 & 0 & 0 & 0 \\ 0 & 0 & 6.41 & -23.57 & 18.76 & 0 & 0 \\ 0 & 0 & 0 & 14.59 & -42.42 & 28.91 & 0 \\ 0 & 0 & 0 & 0 & 23.66 & -101.96 & 86.33 \\ 0 & 0 & 0 & 0 & 0 & 97.40 & -1.74e3 \end{bmatrix} \quad A_6 = \begin{bmatrix} -1.25 & 6.47 & 0 & 0 & 0 & 0 & 0 \\ 0.41 & -3.54 & 3.83 & 0 & 0 & 0 & 0 \\ 0 & 2.30 & -10.24 & 8.97 & 0 & 0 & 0 \\ 0 & 0 & 6.41 & -21.46 & 16.06 & 0 & 0 \\ 0 & 0 & 0 & 12.49 & -41.88 & 31.56 & 0 \\ 0 & 0 & 0 & 0 & 25.82 & -104.61 & 86.33 \\ 0 & 0 & 0 & 0 & 0 & 97.40 & -1.74e3 \end{bmatrix}$$

$$A_7 = \begin{bmatrix} -1.25 & 6.47 & 0 & 0 & 0 & 0 & 0 \\ 0.41 & -3.54 & 3.83 & 0 & 0 & 0 & 0 \\ 0 & 2.30 & -10.24 & 8.97 & 0 & 0 & 0 \\ 0 & 0 & 6.41 & -21.46 & 16.06 & 0 & 0 \\ 0 & 0 & 0 & 12.49 & -39.71 & 28.91 & 0 \\ 0 & 0 & 0 & 0 & 23.66 & -104.17 & 88.94 \\ 0 & 0 & 0 & 0 & 0 & 100.34 & -1.74e3 \end{bmatrix} \quad A_8 = \begin{bmatrix} -1.25 & 6.47 & 0 & 0 & 0 & 0 & 0 \\ 0.41 & -3.54 & 3.83 & 0 & 0 & 0 & 0 \\ 0 & 2.30 & -10.24 & 8.97 & 0 & 0 & 0 \\ 0 & 0 & 6.41 & -21.46 & 16.06 & 0 & 0 \\ 0 & 0 & 0 & 12.49 & -39.71 & 28.91 & 0 \\ 0 & 0 & 0 & 0 & 23.66 & -101.96 & 86.33 \\ 0 & 0 & 0 & 0 & 0 & 97.40 & -1.74e3 \end{bmatrix}.$$

For a given ρ, we restrict the uncertain parameters α_k to S_ρ, defined as

$$S_\rho := \{\alpha \in \mathbb{R}^8 : \sum_{i=1}^{8} \alpha_i = -6|\rho|, -|\rho| \leq \alpha_i \leq |\rho|\},$$

which is a simplex translated to the origin. We would like to determine the maximum value of ρ such that the system is stable by solving the following optimization problem.

$$\begin{aligned} \rho^* := \max \quad & \rho \\ \text{subject to} \quad & \text{System (4.72) is stable for all } \alpha \in S_\rho. \end{aligned} \tag{4.73}$$

To represent S_ρ using the standard unit simplex defined in (2.15), we define the invertible map $g : \Delta^8 \to S_\rho$ as

$$g(\alpha) = [g_1(\alpha) \ \cdots \ g_8(\alpha)], \ g_i(\alpha) := 2|\rho|(\alpha_i - 0.5). \tag{4.74}$$

Then, if we let $A'(\alpha) = A(g(\alpha))$, since g is one-to-one,

$$\{A(\alpha') : \alpha' \in S_\rho\} = \{A(g(\alpha)) : \alpha \in \Delta^8\} = \{A'(\alpha) : \alpha \in \Delta^8\}.$$

Thus, stability of $\dot{\psi}_x(t) = A'(\alpha)\psi_x(t)$, for all $\alpha \in \Delta^l$ is equivalent to stability of Equation (4.72) for all $\alpha \in S_\rho$.

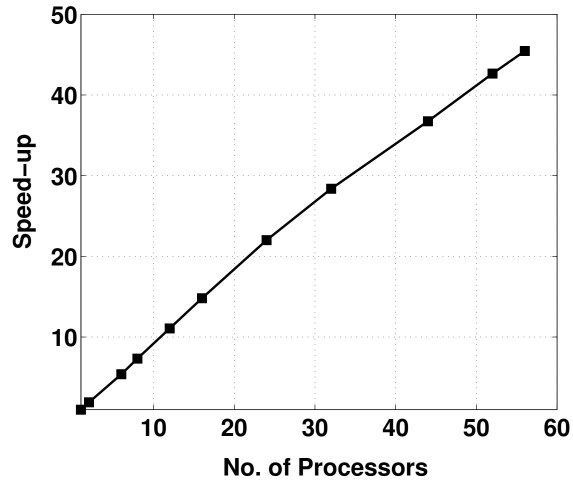


Figure 4.9: Speed-up of set-up and SDP algorithms vs. number of processors for a discretized model of magnetic flux in Tokamak

We solve the optimization problem in (4.73) using bisection. For each trial value of ρ, we use the proposed parallel SDP solver in Algorithm 6 to solve the associated SDP obtained by our parallel set-up Algorithm 5. The SDP problems have 224 constraints with the primal variable $X \in \mathbb{R}^{1092\times1092}$. The normalized value of ρ^*, i.e., $\frac{\rho^*}{\widehat{\eta}(x_{15/2})}$ is found to be 0.0019, where $\widehat{\eta}(x_{15/2}) = 8.419 \cdot 10^{-6}$ from Table 4.3. In this particular example, the optimal value of ρ does not change with the degrees of $P(\alpha)$ and Polya's exponents d_1 and d_2, primarily because the model is affine. The SDPs are constructed and solved on a parallel Linux-based cluster Cosmea at Argonne National Laboratory. Figure 4.9 shows the algorithm speed-up vs. the number of processors. Note that solving this problem by SOSTOOLS (Papachristodoulou *et al.* (2013)) on the same machine is impossible due to the lack of unallocated memory.

4.7.2 Example 2: Accuracy and convergence

The goal of this example is to investigate the effect of the degree d_p of $P(\alpha)$ and the Polya's exponents, d_1, d_2 on the accuracy of our algorithms. Given a computer with a fixed amount of RAM, we compare the accuracy (as we defined in Section 4.6.2) of the proposed algorithms to the SOS algorithm. Consider the system $\dot{x}(t) = A(\alpha)x(t)$ where A is a polynomial degree 3 defined as

$$A(\alpha) = A_1\alpha_1^3 + A_2\alpha_1^2\alpha_2 + A_3\alpha_1\alpha_2\alpha_3 + A_4\alpha_1\alpha_3^2 + A_5\alpha_2^3 + A_6\alpha_3^3 \tag{4.75}$$

with the constraint

$$\alpha \in S_L := \left\{ \alpha \in \mathbb{R}^3 : \sum_{i=1}^{3} \alpha_i = 2L+1, L \leq \alpha_i \leq 1 \right\},$$

where A_i matrices are defined as

$$A_1 = \begin{bmatrix} -0.61 & -0.56 & 0.402 \\ -0.48 & -0.550 & 0.671 \\ -1.01 & -0.918 & 0.029 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -0.484 & -0.86 & 1.5 \\ -0.732 & -0.841 & -0.126 \\ 0.685 & 0.305 & 0.106 \end{bmatrix}, \quad A_3 = \begin{bmatrix} -0.357 & 0.344 & -0.661 \\ -0.210 & -0.505 & 0.588 \\ 0.268 & 0.487 & -0.846 \end{bmatrix},$$
$$A_4 = \begin{bmatrix} -0.881 & -0.436 & 0.228 \\ 0.503 & -0.812 & 0.249 \\ -0.012 & 0.542 & -0.536 \end{bmatrix}, A_5 = \begin{bmatrix} -0.703 & -0.298 & -0.178 \\ 0.402 & -0.761 & -0.300 \\ -0.010 & 0.461 & -0.588 \end{bmatrix}, A_6 = \begin{bmatrix} -0.201 & -0.182 & -0.557 \\ 0.803 & -0.412 & -0.203 \\ -0.440 & 0.011 & -0.881 \end{bmatrix}.$$

Defining g as in Example 1, the problem is

$$\begin{aligned} &\min \quad L \\ &\text{s.t.} \quad \dot{x}(t) = A(g(\alpha))x(t) \text{ is stable for all } \alpha \in \Delta^3. \end{aligned} \tag{4.76}$$

Using bisection in L, as in Example 1, we varied the parameters d_p, d_1 and d_2. The cluster computer Karlin at Illinois Institute of Technology with 24 Gbytes/node of RAM (216 Gbytes total memory) was used to run our algorithm. The upper bounds on the optimal L are shown in Figure 4.10 in terms of d_1 and d_2 and for different d_p. Considering the optimal value of L to be $L_{\text{opt}} = -0.111$, Figure 4.10 shows how increasing d_p and/or d_1, d_2 - when they are still relatively small - improves the accuracy of the algorithm. Figure 4.11 demonstrates how the error in our upper bound for L_{opt} decreases by increasing d_p and/or d_1, d_2.

Table 4.4: Upper bounds found for L_{opt} by the SOS algorithm using different degrees for x and α (inf: infeasible, O.M.: Out of Memory)

Degree in x \ Degree in α	0	1	2
1	Infeasible	Infeasible	Infeasible
2	Infeasible	-0.102	Out of Memory
3	Infeasible	Out of Memory	Out of Memory

For comparison, we solved the same stability problem using the SOS algorithm (Papachristodoulou *et al.* (2013)) using only a single node of the same cluster computer and 24 Gbytes of RAM. We used Putinar's Positivstellensatz (see Section 2.3.4) to impose the constraints $\sum_{i=1}^{3} \alpha_i = 2L+1$ and $L \leq \alpha_i \leq 1$. Table 4.4 shows the upper bounds on L given by the SOS algorithm using different degrees for x and α. By considering a Lyapunov function of degree two in x and degree one in α, the SOS algorithm gives -0.102 as an upper bound on L_{opt} as compared with our value of -0.111. Increasing the degree of α in the Lyapunov function beyond two resulted in a failure due to lack of memory.

4.7.3 Example 3: Evaluating speed-up

In this example, we evaluate the efficiency of the algorithm in using additional processors to decrease computation time. As mentioned in Section 4.6 on computational complexity, the measure of this efficiency is termed speed-up and in Section 4.6.3, we gave a formula for this number. To evaluate the true speed-up, we first ran the set-up algorithm on the Blue Gene supercomputer at Argonne National Laboratory using three random linear systems with different state-space dimensions and numbers of uncertain parameters. Figure 4.12 shows a log-log plot of the computation time

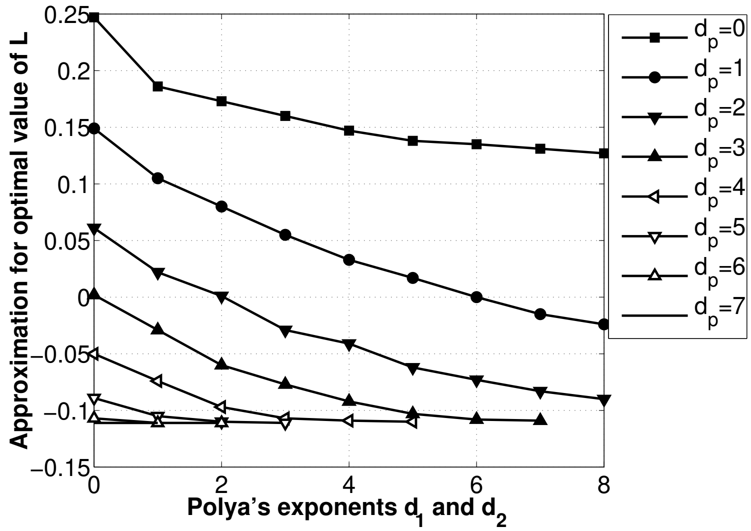


Figure 4.10: Upper bound on optimal L vs. Polya's exponents d_1 and d_2, for different degrees of $P(\alpha)$. ($d_1 = d_2$).

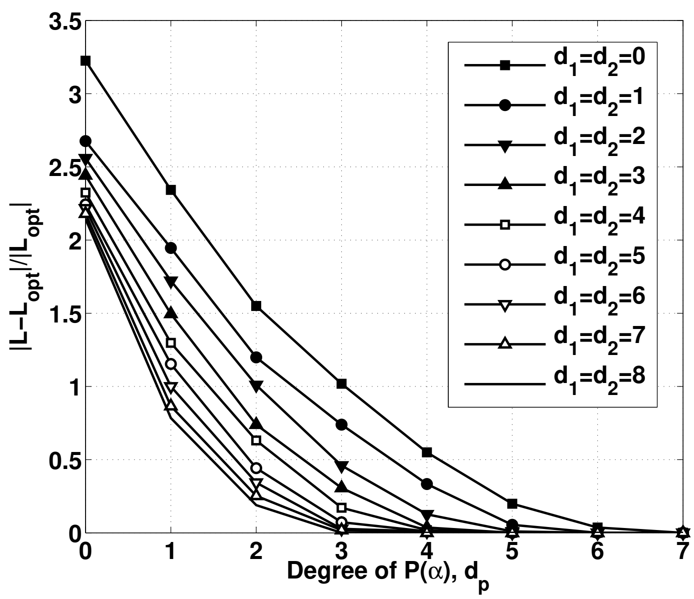


Figure 4.11: Error of the approximation for the optimal value of L vs. degrees of $P(\alpha)$, for different Polya's exponents

of the set-up algorithm vs. the number of processors. One can be observed that the scalability of the algorithm is practically ideal for several different state-space dimensions and numbers of uncertain parameters.

To evaluate the speed-up of the SDP portion of the algorithm, we solved three random SDP problems with different dimensions using the Karlin cluster computer. Figure 4.13 gives a log-log plot of the computation time of the SDP algorithm vs. the number of processors for three different dimensions of the primal variable X and the dual variable y. As indicated in the figure, the three dimensions of the primal variable X are 200, 385 and 1092, and the dimensions of the dual variable y are $K = 50$, 90 and 224, respectively. In all cases, $d_p = 2$ and $d_1 = d_2 = 1$. The linearity of the Time vs. Number of Processors curves in all three cases demonstrates the scalability of the SDP algorithm.

For comparison, we plot the speed-up of our algorithm vs. that of the general-purpose parallel SDP solver SDPARA 7.3.1 as illustrated in Figure 4.14. Although similar for a small number of processors, for a larger number of processors, SDPARA saturates, while our algorithm remains approximately linear.

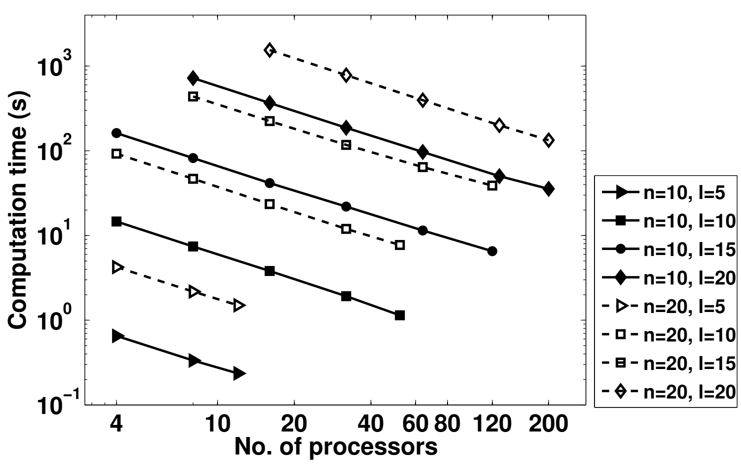


Figure 4.12: Computation time of the parallel set-up algorithm vs. number of processors for different dimensions of linear system n and numbers of uncertain parameters l- executed on Blue Gene supercomputer of Argonne National Labratory

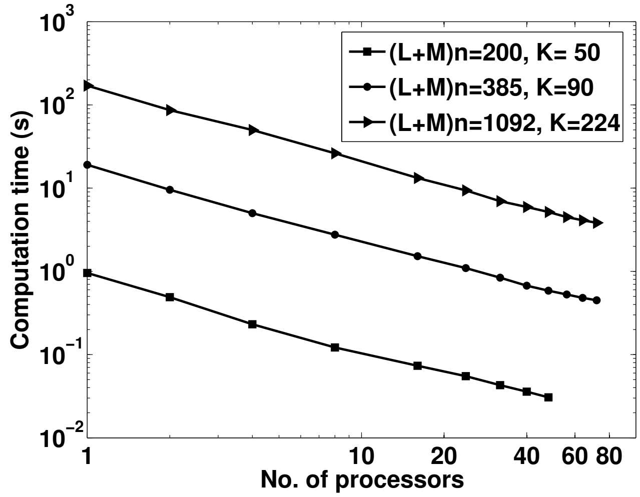


Figure 4.13: Computation time of the parallel SDP algorithm vs. number of processors for different dimensions of primal variable $(L + M)n$ and of dual variable K-executed on Karlin cluster computer

4.7.4 Example 4: Maximum state-space and parameter dimensions for a 9-node Linux-based cluster computer

The goal of this example is to show that given moderate computational resources, the proposed decentralized algorithms can solve robust stability problems for systems with 100+ states. We used the Karlin cluster computer with 24 Gbytes/node of RAM and nine nodes. We ran the set-up and the SDP algorithms to solve the robust stability problem with dimension n and l uncertain parameters on one and nine nodes of Karlin cluster computer. Thus, the total accessible memory was 24 Gbytes and 216 Gbytes, respectively. Using trial and error, for different n and d_1, d_2 we found the largest l for which the algorithms do not terminate due to insufficient memory (Figure 4.15). In all of the runs $d_a = d_p = 1$. Figure 4.15 shows that by using 216 Gbytes of RAM, the algorithms can solve the stability problem of size $n = 100$ with 4 uncertain parameters in $d_1 = d_2 = 1$ Polya's iteration and with 3 uncertain parameters in $d_1 = d_2 = 4$ Polya's iterations.

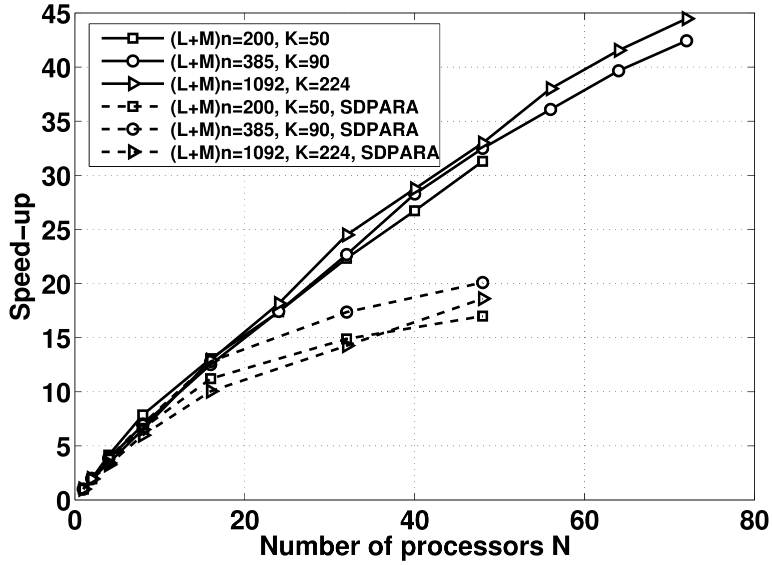


Figure 4.14: Comparison between the speed-up of the present SDP solver and SDPARA 7.3.1, executed on Karlin cluster computer

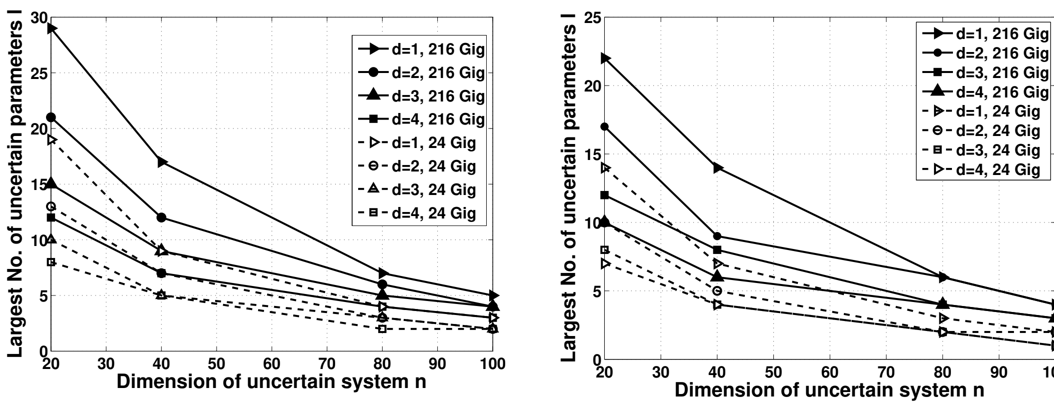


Figure 4.15: Largest number of uncertain parameters of n-dimensional systems for which the set-up algorithm (Left) and SDP solver (Right) can solve the robust stability problem of the system using 24 and 216 Gig of RAM

Chapter 5

PARALLEL ALGORITHMS FOR ROBUST STABILITY ANALYSIS OVER HYPERCUBES

5.1 Background and motivation

In Chapter 4, we proposed a distributed parallel algorithm for stability analysis over a simplex. Unfortunately, simplices are rather restrictive forms of uncertainty set in that they do not allow for parameters which take values on intervals or polytopes. Additionally, we hope to eventually extend our algorithms to the problem of nonlinear stability, which requires search over positive polynomials defined over a set which contains the origin. Since simplicies do not include the origin, our algorithms cannot be readily applied to such problems.

In this chapter, our goal is to generalize our algorithms in Chapter 4 in order to perform robust stability analysis on linear systems with uncertain parameters defined over hypercubes. Several recent papers such as Chesi *et al.* (2005) and Bliman (2004a), have proposed LMI-based techniques to construct parameter-dependent quadratic-in-state Lyapunov functions for this class of systems. In particular, researchers (Chesi (2005)) have recently turned to SOS methods and the Positivstellensatz results (see Section 2.1) to construct increasingly accurate and increasingly complex LMI-based tests for stability over hypercubes. Unfortunately, due to the inherent intractability of the problem of polynomial optimization, SOS based algorithms typically run out of memory for even relatively small-sized problems; see *e.g.,* Table 4.4 of Section 4.7. This makes it difficult to solve SOS-based algorithms on desktop computers. In this chapter, we seek for a parallel methodology to distribute the required memory and computation among hundreds of processors - each processor possessing a dedicated memory.

5.1.1 Our Contributions

We start by proposing an extension to Polya's theorem. This new result parameterizes every multi-homogeneous polynomial which is positive over a given multi-simplex/hypercube. Based on this result, we propose a parallel algorithm to set-up a sequence of block-structured LMIs (similar to the case of a single simplex). Solutions to these LMIs define parameter-dependent Lyapunov functions for the system. Finally, we use our parallel SDP solver in Section 4.5 to efficiently solve these structured LMIs. Similar to Algorithm 7, the proposed set-up algorithm in this chapter has no centralized computation, memory or communication, hence resulting in a near-ideal speed-up. Specifically, we show that the communication operations per processor is proportional to $\frac{1}{N_c}$, where N_c is the number of processors used by the algorithm. This implies that by increasing the number of processors, we actually decrease the communication overhead per processor and improve the speed-up. Naturally, there exists an upper-bound for the number of processors which can be used by the algorithm, beyond which, no speed-up is gained. This upper-bound is proportional to the number of uncertain parameters in the system and for practical problems will be far larger than the number of available processors.

5.2 Notation and preliminaries on multi-homogeneous polynomials

Recall from Section 4.2 that we denote a monomial by $\alpha^\gamma = \prod_{i=1}^{l} \alpha_i^{\gamma_i}$, where $\alpha \in \mathbb{R}^l$ is the vector of variables and $\gamma \in W_d$ is the vector of exponents, were W_d is the set of exponents defined in (4.2). Now consider the case where $\alpha = [\alpha_1, \cdots, \alpha_n]$ with $\alpha_i \in \mathbb{R}^{l_i}$, and $h = [h_1, \cdots, h_n]$, where $h_i \in W_{d_{p_i}}$. Then, we define the set of

n-variate *multi-homogeneous* polynomials of degree vector $D = [d_1, \cdots, d_n] \in \mathbb{N}^n$ as (a generalization of (2.19))

$$\left\{ P \in \mathbb{R}[\alpha_1, \cdots, \alpha_n] : P(\alpha) = \sum_{h_1 \in W_{d_1}} \cdots \sum_{h \in W_{d_n}} P_{\{h_1, \cdots, h_n\}} \alpha_1^{h_1} \cdots \alpha_n^{h_n} \right\}. \tag{5.1}$$

Note that for any $i \in \{1, \cdots, n\}$, the element d_i of the degree vector D is the degree of $\alpha_i^{h_i}$ in P. For brevity, we denote the index set $\{h_1, \cdots, h_n\}$ by $\mathcal{H}_n$ and $\{h_{1_j}, \cdots, h_{n_j}\}$ by $\mathcal{H}_{n,j}$, where h_{i_j} is defined as the j^{th} element of $h_i \in W_{d_i}$ using lexicographical ordering. We define the unit multi-simplex $\tilde{\Delta}^{\{l_1, \cdots, l_N\}}$ as the Cartesian product of N unit simplexes; i.e., $\tilde{\Delta}^{\{l_1, \cdots, l_N\}} := \Delta^{l_1} \times \cdots \times \Delta^{l_N}$. Given $r_i \in \mathbb{R}$, let us define the hypercube $\Phi^n \subset \mathbb{R}^n$ as

$$\Phi^n := \{\alpha \in \mathbb{R}^n : |\alpha_i| \leq r_i,\, i = 1, \cdots, n\}.$$

Claim 1: *For every non-homogeneous polynomial $F(\alpha)$ with $\alpha \in \tilde{\Delta}^{\{l_1, \cdots, l_n\}}$, there exists a multi-homogeneous polynomial P such that*

$$\left\{ F(\alpha) \in \mathbb{R} : \alpha \in \tilde{\Delta}^{\{l_1, \cdots, l_n\}} \right\} = \left\{ P(\beta) \in \mathbb{R} : \beta \in \tilde{\Delta}^{\{l_1, \cdots, l_n\}} \right\}.$$

To construct P, first let N_F be the number of monomials in F. Define $t^{(k)} := \left[t_1^{(k)}, \cdots, t_n^{(k)} \right]$ for $k = 1, \cdots, N_F$, where $t_i^{(k)}$ is the sum of the exponents of the variables inside Δ^{l_i}, in the k^{th} monomial of F. Then, one can construct P by multiplying the k^{th} monomial of F (according to lexicographical ordering) for $k = 1, \cdots, N_F$ by

$$\prod_{i=1}^{n} \left(\sum_{j=1}^{l_i} \alpha_{i_j} \right)^{T_i - t_i^{(k)}}, \quad T_i := \max_{k \in \{1, \cdots, N_F\}} t_i^{(k)}.$$

For more clarification, we provide the following example of constructing the multi-homogeneous polynomial P.

Example: Consider the non-homogeneous polynomial

$$F(\alpha) = F_1(\alpha_{1,1} + \alpha_{12})\alpha_{2,1} + F_2 \alpha_{1,2}^2 + F_3 \alpha_{2,2},$$

where $(\alpha_{1_1}, \alpha_{1_2}), (\alpha_{2_1}, \alpha_{2_2}) \in \Delta^2$, $t^{(1)} = t^{(2)} = [1,1], t^{(3)} = [2,0]$ and $t^{(4)} = [0,1]$. Then, the multi-homogeneous polynomial $P(\alpha)$ is

$$\begin{aligned} P(\alpha) &= F_1(\alpha_{1_1} + \alpha_{1_2})^2 \alpha_{2_1} + F_2 \alpha_{1_2}^2 (\alpha_{2_1} + \alpha_{2_2}) + F_3 (\alpha_{1_1} + \alpha_{1_2})^2 \alpha_{2_2} \\ &= F_{\{(2,0),(1,0)\}} \alpha_{1_1}^2 \alpha_{2_1} + F_{\{(2,0),(0,1)\}} \alpha_{1_1}^2 \alpha_{2_2} + F_{\{(1,1),(1,0)\}} \alpha_{1_1} \alpha_{1_2} \alpha_{2_1} \\ &+ F_{\{(1,1),(0,1)\}} \alpha_{1_1} \alpha_{1_2} \alpha_{2_2} + F_{\{(0,2),(1,0)\}} \alpha_{1_2}^2 \alpha_{2_1} + F_{\{(0,2),(0,1)\}} \alpha_{1_2}^2 \alpha_{2_2}. \end{aligned}$$

Thus, the coefficients of the multi-homogeneous polynomial P are

$$\begin{aligned} &P_{\{(2,0),(1,0)\}} = F_1, \qquad &&P_{\{(2,0),(0,1)\}} = F_3, \qquad &&P_{\{(1,1),(1,0)\}} = 2F_1 \\ &P_{\{(1,1),(0,1)\}} = 2F_3, \qquad &&P_{\{(0,2),(1,0)\}} = F_1 + F_2, \qquad &&P_{\{(0,2),(0,1)\}} = F_2 + F_3. \end{aligned}$$

Claim 2: *For every polynomial $F(x)$ with $x \in \Phi^n$, there exists a multi-homogeneous polynomial P such that*

$$\{F(x) \in \mathbb{R} : x \in \Phi^n\} = \left\{P(\alpha, \beta) \in \mathbb{R} : \alpha, \beta \in \mathbb{R}^n \text{ and } (\alpha_i, \beta_i) \in \Delta^2 \text{ for } i = 1, \cdots, n\right\}. \tag{5.2}$$

To construct P, we propose the following steps.

1. Define new variables $\alpha_i := \frac{x_i + r_i}{2\, r_i} \in [0,1]$ for $i = 1, \cdots, n$.

2. Define $Q(\alpha_1, \cdots, \alpha_n) := F(2\, r_1 \alpha_1 - r_1, \cdots, 2\, r_n \alpha_n - r_n)$.

3. Define a new set of variables $\beta_i := 1 - \alpha_i$ for $i = 1, \cdots, n$.

4. Let N_Q be the number of monomials in Q. Define $t^{(k)} := \left[t_1^{(k)}, \cdots, t_n^{(k)}\right]$ for $k = 1, \cdots, N_Q$, where $t_i^{(k)}$ is the sum of the exponents of the variables inside Δ^2, in the k^{th} monomial of Q. Then, for $k = 1, \cdots, N_Q$, multiply the k^{th} monomial of Q (according to lexicographical ordering) by

$$\prod_{i=1}^{n} (\alpha_i + \beta_i)^{T_i - t_i^{(k)}}, \quad T_i := \max_{k \in \{1, \cdots, N_Q\}} t_i^{(k)}.$$

We provide the following example to further clarify this procedure.

Example: Suppose $F(x_1, x_2) = x_1^2 + x_2$, with $x_1 \in [-2, 2]$ and $x_2 \in [-1, 1]$. Define $\alpha_1 := \frac{x_1+2}{4} \in [0, 1]$ and $\alpha_2 := \frac{x_2+1}{2} \in [0, 1]$. Then, define

$$Q(\alpha_1, \alpha_2) := f(4\alpha_1 - 2, 2\alpha_2 - 1) = 16\alpha_1^2 - 16\alpha_1 + 2\alpha_2 + 3$$

By homogenizing Q we obtain the multi-homogeneous polynomial

$$\begin{aligned} P(\alpha, \beta) =& 16\alpha_1^2(\alpha_2 + \beta_2) - 16\alpha_1(\alpha_1 + \beta_1)(\alpha_2 + \beta_2) + 2\alpha_2(\alpha_1 + \beta_1)^2 \\ &+ 3(\alpha_1 + \beta_1)^2(\alpha_2 + \beta_2), \quad (\alpha_1, \beta_1), (\alpha_2, \beta_2) \in \Delta^2 \end{aligned}$$

with the degree vector $D = [2, 1]$, where $d_1 = 2$ is the sum of exponents of α_1 and β_1 in every monomial of P, and $d_2 = 1$ is the sum of exponents of α_2 and β_2 in every monomial of P.

In the following theorem (Kamyar and Peet (2012b)), we parameterize all of the multi-homogeneous polynomials which are positive over a multi-simplex.

Theorem 16. *(Polya's theorem, multi-simplex version) A matrix-valued multi-homogeneous polynomial F satisfies $F(\alpha, \beta) > 0$ for all $(\alpha_i, \beta_i) \in \Delta^2, i = 1, \cdots, n$, if there exist $e \geq 0$ such that all the coefficients of*

$$\left(\prod_{i=1}^{n} (\alpha_i + \beta_i)^e \right) F(\alpha, \beta)$$

are positive definite.

Proof. We use induction as follows.

Basis step: Suppose $n = 1$. Then, from the simplex version of Polya's theorem (Theorem 2) it follows that for every $F(\alpha, \beta) > 0$ with $(\alpha, \beta) \in \Delta^2$, there exists some $e \geq 0$ such that all of the coefficients of $(\alpha_1 + \beta_1)^e F(\alpha, \beta)$ are positive definite.

Induction hypothesis: Suppose for every $F(\alpha, \beta) > 0$ with $(\alpha_i, \beta_i) \in \Delta^2$, $i = 1, \cdots, k$ there exists some $e \geq 0$ such that all of the coefficients of

$$\left(\prod_{i=1}^{k} (\alpha_i + \beta_i)^e\right) F(\alpha, \beta)$$

are positive definite.

We need to prove that for every $F(\alpha, \beta) > 0$ with $(\alpha_i, \beta_i) \in \Delta^2$, $i = 1, \cdots, k+1$ there exists some $e^* \geq 0$ such that all of the coefficients of

$$\left(\prod_{i=1}^{k+1} (\alpha_i + \beta_i)^{e^*}\right) F(\alpha, \beta)$$

are positive definite. From the induction hypothesis it follows that for any fixed $(\hat{\alpha}, \hat{\beta}) \in \Delta^2$, if $F(\alpha_1, \cdots, \alpha_k, \hat{\alpha}, \beta_1, \cdots, \beta_k, \hat{\beta}) > 0$ for all $(\alpha_i, \beta_i) \in \Delta^2$, $i = 1, \cdots, k$, then there exists some $e \geq 0$ such that all of the coefficients of

$$\left(\prod_{i=1}^{k} (\alpha_i + \beta_i)^e\right) F(\alpha_1, \cdots, \alpha_k, \hat{\alpha}, \beta_1, \cdots, \beta_k, \hat{\beta}) \tag{5.3}$$

are positive definite. Using our notation in (2.19), we can expand (5.3) as

$$\left(\prod_{i=1}^{k} (\alpha_i + \beta_i)^e\right) F(\alpha_1, \cdots, \alpha_k, \hat{\alpha}, \beta_1, \cdots, \beta_k, \hat{\beta}) = \sum_{\substack{h,g \in \mathbb{N}^k \\ h+g=d+e\cdot\mathbf{1}_k}} f_{h,g}(\hat{\alpha}, \hat{\beta}) \alpha_1^{h_1} \beta_1^{g_1} \cdots \alpha_k^{h_k} \beta_k^{g_k}, \tag{5.4}$$

in which we have denoted the coefficients of Product (5.3) by $f_{h,g}(\hat{\alpha}, \hat{\beta})$ and we have denoted the degree vector of F by d. Also $\mathbf{1}_k \in \mathbb{N}^k$ denotes the vector of ones. Because F is a homogeneous polynomial, $f_{h,g}$ are also homogeneous polynomials. Since $f_{h,g}(\hat{\alpha}, \hat{\beta}) > 0$ for all $(h, g) \in M_{d,e} := \{(h, g) \in \mathbb{N}^k \times \mathbb{N}^k : h + g = d + e \cdot \mathbf{1}\}$, Polya's theorem implies that there exist $l_{g,h} \geq 0$ for any $h, g \in M_{d,e}$ such that all of the coefficients of $(\hat{\alpha} + \hat{\beta})^{l_{g,h}} f_{g,h}(\hat{\alpha}, \hat{\beta})$ are positive definite. Let us define

$$e^* := \max\left\{\max_{h,g \in M_{d,e}} \{l_{g,h}\}, e\right\}.$$

Then, clearly all of the coefficients in $(\hat{\alpha}+\hat{\beta})^{e^*} f_{g,h}(\hat{\alpha},\hat{\beta})$ are also positive definite. By multiplying both sides of (5.4) by $(\hat{\alpha}+\hat{\beta})^{e^*}$ we have

$$\left(\prod_{i=1}^{k}(\alpha_i+\beta_i)^e\right)(\hat{\alpha}+\hat{\beta})^{e^*}F(\alpha_1,\cdots,\alpha_k,\hat{\alpha},\beta_1,\cdots,\beta_k,\hat{\beta})$$
$$=\sum_{\substack{h,g\in\mathbb{N}^k\\ h+g=d+e\cdot\mathbf{1}_k}}(\hat{\alpha}+\hat{\beta})^{e^*}f_{h,g}(\hat{\alpha},\hat{\beta})\alpha_1^{h_1}\beta_1^{g_1}\cdots\alpha_k^{h_k}\beta_k^{g_k}. \tag{5.5}$$

Since all of the coefficients of $(\hat{\alpha}+\hat{\beta})^{e^*} f_{h,g}(\hat{\alpha},\hat{\beta})$ are positive definite, all of the coefficients of the monomials on the right hand side of (5.5) are positive definite. Moreover, because $e^* \geq e$

$$\left(\prod_{i=1}^{k}(\alpha_i+\beta_i)^{e^*}\right)(\hat{\alpha}+\hat{\beta})^{e^*}F(\alpha_1,\cdots,\alpha_k,\hat{\alpha},\beta_1,\cdots,\beta_k,\hat{\beta}) \tag{5.6}$$

will also have all positive definite coefficients. Since we chose $(\hat{\alpha},\hat{\beta})$ arbitrarily from the simplex Δ^2, by replacing $\hat{\alpha}$ and $\hat{\beta}$ with α_{k+1} and β_{k+1} in (5.6),

$$\left(\prod_{i=1}^{k+1}(\alpha_i+\beta_i)^{e^*}\right)F(\alpha,\beta)\ \text{ with }\ (\alpha_i,\beta_i)\in\Delta^2,\ i=1,\cdots,k+1$$

will have all positive definite coefficients. $\square$

5.3 Setting up the problem of robust stability analysis over multi-simplex

In this section, we focus on the problem of robust stability of a system the form

$$\dot{x}(t)=A(\alpha)x(t), \tag{5.7}$$

where $A(\alpha)\in\mathbb{R}^{n\times n}$ is a multi-homogeneous polynomial of degree vector D_a and $\alpha\in\tilde{\Delta}^{\{l_1,\cdots,l_N\}}$ denotes the parametric uncertainty in the system. Note that if A is not homogeneous, one can use Claim 1 to find a multi-homogeneous representation for A over the multi-simplex. Furthermore, if $\alpha\in\Phi^N$, then one can use Claim 2 to find an equivalent representation for A over the multi-simplex $\tilde{\Delta}^{\{l_1,\cdots,l_N\}}$.

The following theorem gives necessary and sufficient conditions for asymptotic stability of System (5.7).

Theorem 17. *The linear system* (5.7) *is stable if and only if there exists a polynomial matrix* $P(\alpha)$ *such that* $P(\alpha) > 0$ *and*

$$\mathrm{A}^T(\alpha)P(\alpha) + P(\alpha)\mathrm{A}(\alpha) < 0 \quad \textit{for all} \;\; \alpha \in \tilde{\Delta}^{\{l_1,\cdots,l_N\}}. \tag{5.8}$$

Unfortunately, the question of feasibility of the inequalities in Theorem 17 is NP-hard. In this section, we show that applying Theorem 16 yields a sequence of SDPs of increasing size (and precision) whose solutions converge to a solution of the inequalities in Theorem 17. Motivated by the result in Bliman *et al.* (2006), we consider $P(\alpha)$ to be homogeneous. In particular, let P be a multi-homogeneous matrix-valued polynomial of form

$$P(\alpha) = \sum_{h_N \in W_{d_{p_N}}} \cdots \sum_{h_1 \in W_{d_{p_1}}} P_{\mathcal{H}_N} \alpha_1^{h_1} \cdots \alpha_N^{h_N}, \tag{5.9}$$

with degree $d_p = \sum_{i=1}^N d_{p_i}$ and unknown coefficients $P_{\mathcal{H}_N} \in \mathbb{S}^n$. Moreover, let $A(\alpha)$ be of the form

$$A(\alpha) = \sum_{h_1 \in W_{d_{a_1}}} \cdots \sum_{h_N \in W_{d_{a_N}}} A_{\mathcal{H}_N} \alpha_1^{h_1} \cdots \alpha_N^{h_N}, \tag{5.10}$$

with degree $d_a = \sum_{i=1}^N d_{a_i}$. It follows from Theorem 16 that the Lyapunov inequalities in Theorem 17 hold for all $\alpha \in \tilde{\Delta}^{\{l_1,\cdots,l_N\}}$ if there exist some $d_1 \geq 0$ and $d_2 \geq 0$ such that

$$\prod_{i=1}^N \left(\sum_{j=1}^{l_i} \alpha_{i_j} \right)^{d_1} P(\alpha) \quad \text{and} \tag{5.11}$$

$$-\prod_{i=1}^N \left(\sum_{j=1}^{l_i} \alpha_{i_j} \right)^{d_2} \left(\mathrm{A}^T(\alpha)P(\alpha) + P(\alpha)\mathrm{A}(\alpha) \right) \tag{5.12}$$

have all positive definite coefficients. By substituting for $A(\alpha)$ and $P(\alpha)$ in (5.11) and (5.12) from (5.10) and (5.9), we find that the inequalities of Theorem 17 hold if there exists $d_1, d_2 \geq 0$ such that

$$\sum_{h_1 \in W_{d_1}} \cdots \sum_{h_N \in W_{d_N}} \beta_{\{\mathcal{H}_N, \Gamma_N\}} P_{\mathcal{H}_N} > 0 \tag{5.13}$$

for all $\gamma_1 \in W_{d_{p_1}+d_1}, \cdots, \gamma_N \in W_{d_{p_N}+d_1}$, and

$$\sum_{h_1 \in W_{d_1}} \cdots \sum_{h_N \in W_{d_N}} \left(H^T_{\{\mathcal{H}_N, \Gamma_N\}} P_{\mathcal{H}_N} + P_{\mathcal{H}_N} H_{\{\mathcal{H}_N, \Gamma_N\}} \right) < 0 \tag{5.14}$$

for all $\gamma_1 \in W_{d_{pa_1}+d_2}, \cdots, \gamma_N \in W_{d_{pa_N}+d_2}$, where recall that $\mathcal{H}_N$ denotes $\{h_1, \cdots, h_N\}$, Γ_N denotes $\{\gamma_1, \cdots, \gamma_N\}$ and $d_{pa_i} = d_{p_i} + d_{a_i}$ for $i = 1, \cdots, N$.

5.3.1 General formulae for calculating coefficients β and H

To calculate the $\left\{\beta_{\{\mathcal{H}_N, \Gamma_N\}}\right\}$ coefficients and $\left\{H_{\{\mathcal{H}_N, \Gamma_N\}}\right\}$ we provide the following recursive formulae. These formulae are generalization of the recursive formulae in 4.3.1 for the case of a single simplex. First, for all $\gamma_1 \in W_{d_{p_1}}, \cdots, \gamma_N \in W_{d_{p_N}}$, and for all $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ set

$$\beta^{(0)}_{\{\mathcal{H}_N, \Gamma_N\}} = \begin{cases} 1 & \text{if } h_1 = \gamma_1, \cdots, h_N = \gamma_N \\ 0 & \text{otherwise.} \end{cases} \tag{5.15}$$

Then, for $i = 1, \cdots, d_1$, for all $\gamma_1 \in W_{d_{p_1}} + i, \cdots, \gamma_N \in W_{d_{p_N}} + i$ and for all $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$, $\beta^{(i)}_{\{\mathcal{H}_N, \Gamma_N\}}$ can be calculated using

$$\beta^{(i)}_{\{\mathcal{H}_N, \Gamma_N\}} = \sum_{\lambda_N \in W_1} \cdots \sum_{\lambda_1 \in W_1} \beta^{(i-1)}_{\{\mathcal{H}_N, \{\gamma_1 - \lambda_1, \cdots, \gamma_N - \lambda_N\}\}}. \tag{5.16}$$

Finally, set $\beta_{\{\mathcal{H}_N, \Gamma_N\}} = \beta^{(d_1)}_{\{\mathcal{H}_N, \Gamma_N\}}$, where $\gamma \in W_{d_p+d_1}$.

To calculate $\left\{H_{\{\mathcal{H}_N, \Gamma_N\}}\right\}$, first let

$$H^{(0)}_{\{\mathcal{H}_N, \Gamma_N\}} = \sum_{\substack{\lambda_N \in W_{d_{a_N}}: \\ \lambda_N + h_N = \gamma_N}} \cdots \sum_{\substack{\lambda_1 \in W_{d_{a_1}}: \\ \lambda_1 + h_1 = \gamma_1}} A_{\{\lambda_1, \cdots, \lambda_N\}}. \tag{5.17}$$

for $\gamma_1 \in W_{d_{pa_1}}, \cdots, \gamma_N \in W_{d_{pa_N}}$ and $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$. Then, for $i = 1, \ldots, d$, $\gamma_1 \in W_{d_{pa_1}+i}, \cdots, \gamma_1 \in W_{d_{pa_N}+i}$ and $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ we have

$$H^{(i)}_{\{\mathcal{H}_N,\Gamma_N\}} = \sum_{\lambda_N \in W_1} \cdots \sum_{\lambda_1 \in W_1} H^{(i-1)}_{\{\mathcal{H}_N,\{\gamma_1-\lambda_1,\cdots,\gamma_N-\lambda_N\}\}}. \tag{5.18}$$

Finally, set $H_{\{\mathcal{H}_N,\Gamma_N\}} = H^{(d_2)}_{\{\mathcal{H}_N,\Gamma_N\}}$, where $\gamma_1 \in W_{d_{pa_1}+d_2}, \cdots, \gamma_N \in W_{d_{pa_N}+d_2}$.

5.3.2 The SDP elements associated with the multi-simplex version of Polya's theorem

To solve the LMI conditions in (5.13) and (5.14), we express them in the form of a dual Semi-Definite Programming (SDP) problem with a block-diagonal structure that is suitable for parallel computation. Define the element C of the SDP formulation of Conditions (5.13) and (5.14) as

$$C := \text{diag}(C_1, \cdots C_L, C_{L+1}, \cdots C_{L+M}), \tag{5.19}$$

where for given Polya's exponents d_1 and d_2,

$$L = \prod_{i=1}^{N} \frac{(d_{p_i} + d_1 + l_i - 1)!}{(d_{p_i} + d_1)!(l_i - 1)!} \tag{5.20}$$

is the number of monomials in $\prod_{i=1}^{N} \left(\sum_{j=1}^{l_i} \alpha_{i,j} \right)^{d_1} P(\alpha)$ and

$$M = \prod_{i=1}^{N} \frac{(d_{p_i} + d_{a_i} + d_2 + l_i - 1)!}{(d_{p_i} + d_{a_i} + d_2)!(l_i - 1)!} \tag{5.21}$$

is the number of monomials in

$$\prod_{i=1}^{N} \left(\sum_{j=1}^{l_i} \alpha_{i,j} \right)^{d_2} (A^T(\alpha)P(\alpha) + P(\alpha)A(\alpha)),$$

and for $j = 1, \cdots, L + M$,

$$C_j := \begin{cases} \epsilon I_n \zeta^{(j)}, & \text{if } 1 \le j \le L \\ 0_n, & \text{if } L + 1 \le j \le L + M, \end{cases} \tag{5.22}$$

where $\epsilon > 0$. In (5.22), we define $\zeta^{(j)} \in \mathbb{N}^L$ recursively as follows. First, let

$$\zeta^{(0)} = \left[\frac{(d_{p_N} + d_1)!}{\prod_{i=1}^{l_N} h_{(N,i,1)}!}, \cdots, \frac{(d_{p_N} + d_1)!}{\prod_{i=1}^{l_N} h_{(N,i,f(l_N, d_{P_N} + d_1))}!} \right],$$

where we have denoted the exponent of the i^{th} variable in the j^{th} (according to lexicographical ordering) element of $W_{d_{P_N}}$ by $h_{(N,i,j)}$. Recall that

$$f(l,g) := \frac{(l+g-1)!}{g!(l-1)!}$$

is the number of monomials in a polynomial of degree g with l variables. Then, for $k = 1, \cdots, N$, define

$$\zeta^{(k)} := \zeta^{(k-1)} \otimes \left[\frac{(d_{p_{r(k)}} + d_1)!}{\prod_{i=1}^{l_{r(k)}} h_{(r(k),i,1)}!}, \cdots, \frac{(d_{p_{r(k)}} + d_1)!}{\prod_{i=1}^{l_{r(k)}} h_{(r(k),i,s(k))}!} \right],$$

where $r(k) := N - k + 1$ and $s(k) := f(l_{r(k)}, d_{p_{r(k)}} + d_1)$. Finally, set $\zeta = \zeta^{(N)}$.

For $i = 1, \cdots, K$, define the elements B_i of the SDP as

$$B_i = \operatorname{diag}\left(B_{i,1}, \cdots, B_{i,L}, B_{i,L+1}, \cdots, B_{i,L+M}\right), \tag{5.23}$$

where

$$K = \frac{n(n+1)}{2} \prod_{i=1}^{N} \frac{(d_{p_i} + l_i - 1)!}{d_{p_i}!(l_i - 1)!}, \tag{5.24}$$

is the total number of dual variables in the SDP problem (i.e., the total number of upper-triangular elements in all of the coefficients of $P(\alpha)$) and where

$$B_{i,j} = \begin{cases} \sum\limits_{h_N \in W_{d_{p_N}}} \cdots \sum\limits_{h_1 \in W_{d_{p_1}}} \beta_{\{\mathcal{H}_N, \Gamma_{N,j}\}} V_{\mathcal{H}_N}(e_i), & \text{if } 1 \leq j \leq L \\ -\sum\limits_{h_N \in W_{d_{p_N}}} \cdots \sum\limits_{h_1 \in W_{d_{p_1}}} H^T_{\{\mathcal{H}_N, \Gamma_{N,j-L}\}} V_{\mathcal{H}_N}(e_i) + V_{\mathcal{H}_N}(e_i) H_{\{\mathcal{H}_N, \Gamma_{N,j-L}\}} & \text{if } L+1 \leq j \leq L+M, \end{cases} \tag{5.25}$$

where recall from Section 5.2 that $\Gamma_{N,j} = \{\gamma_{1_j}, \cdots, \gamma_{N_j}\}$, where γ_{i_j} is the j^{th} element of $W_{d_{p_i}+d_1}$ using lexicographical ordering, and

$$V_{\mathcal{H}_N}(x) = \sum_{k=1}^{\tilde{N}} E_k \; x_{k+N(I_{\mathcal{H}_N}-1)},$$

where E_k is the canonical basis for $\mathbb{S}^n$ defined in (4.28), $I_{\mathcal{H}_N}$ is the lexicographical index of monomial $\alpha_1^{h_1} \cdots \alpha_N^{h_N}$, and $\tilde{N} := \frac{n(n+1)}{2}$. Finally, we complete the definition of the SDP problem by setting $a = \vec{1} \in \mathbb{R}^K$. In the following section, we propose a parallel set-up algorithm to calculate the SDP elements defined in this section.

5.3.3 A parallel algorithm for setting up the SDP

In this section, we propose a parallel set-up algorithm for computing the SDP elements in (5.19) and (5.23). An abridged description of the algorithm is presented in Algorithm 7, wherein we suppose the algorithm is executed on N_c number of processors. A C++ parallel implementation of the algorithm is available at:
`www.sites.google.com/a/asu.edu/kamyar/software`.

5.4 Computational complexity analysis of the set-up algorithm

In this section, we discuss the performance of Algorithm 7 in terms of speed-up, computation cost, communication cost and memory requirement.

5.4.1 Computational cost of the set-up algorithm:

The most computationally expensive part of the algorithm is calculation of the elements $B_{i,j}$ elements for $i = 1, \cdots, K$ and $j = 1, \cdots, L + M$. If the number of available processors is

$$N_c = L_0 := \prod_{i=1}^{N} \frac{(d_{p_i} + l_i - 1)!}{(d_{p_i})!(l_i - 1)!},$$

then the number of operations per processor at each Polya's iteration of Algorithm 7 is

$$\sim K \cdot L_0 \left(\texttt{floor}\left(\frac{L}{N_c}\right) + \texttt{floor}\left(\frac{M}{N_c}\right)\right) n^3 \sim n^5 \prod_{i=1}^{N} l_i^{2d_{p_i}+d_{a_i}+d_2}, \tag{5.27}$$

Inputs:

N : dimension of multi-simplex; $l_1, \cdots, l_N$: dimensions of simplexes; D_p, D_a : degree vectors of P and A; coefficients of A; $\hat{d}_1, \hat{d}_2$: Polya's exponents.

Initialization:

for $i = 1, \cdots, N_c$, *processor* i **do**

Set $d_1 = d_2 = 0$ and $d_{pa} = d_p + d_a$.

Calculate the number of monomials in $P(\alpha)$, i.e., L using (5.20).

Calculate the number of monomials in $P(\alpha)A(\alpha)$, i.e., M using (5.21).

Calculate the per-processor number of monomials in $P(\alpha)$ and $P(\alpha)A(\alpha)$, i.e.,

$$L' = \texttt{floor}\,(L/N_c) \quad \text{and} \quad M' = \texttt{floor}\,(M/N_c)\,. \tag{5.26}$$

for $\gamma_1, h_1 \in W_{d_{p_1}}, \cdots, \gamma_N, h_N \in W_{d_{p_N}}$ **do**

Calculate $\beta_{\{\mathcal{H}_N,\Gamma_N\}}$ using (5.15).

end

for $\gamma_1 \in W_{d_{pa_1}}, \cdots, \gamma_N \in W_{d_{pa_N}}$ *and* $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ **do**

Calculate $H_{\{\mathcal{H}_N,\Gamma_N\}}$ using (5.17).

end

end

Polya's iterations:

for $i = 1, \cdots, N_c$, *processor* i **do**

for $d_1 = 1, \cdots, \hat{d}_1$ **do**

Set $d_p = d_p + 1$. Update L and L' according to (5.20) and (5.26).

for $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ **do**

Update $\beta_{\{\mathcal{H}_N,\Gamma_N\}}$ for $\gamma_{(i-1)L'+1} \in W_{d_{p_{(i-1)L'+1}}}, \cdots, \gamma_{iL'} \in W_{d_{p_{iL'}}}$ as in (5.16).

end

end

for $d_2 = 1, \cdots, \hat{d}_2$ **do**

Set $d_{pa} = d_{pa} + 1$. Update M and M' according to (5.21) and (5.26).

for $h_1 \in W_{d_{p_1}}, \cdots, h_N \in W_{d_{p_N}}$ **do**

Update $H_{\{\mathcal{H}_N,\Gamma_N\}}$ for $\gamma_{(i-1)M'+1} \in W_{d_{pa_{(i-1)M'+1}}}, \cdots, \gamma_{iM'} \in W_{d_{pa_{iM'}}}$ using (5.18).

end

end

end

Calculating the SDP elements:

for $i = 1, \cdots, N_c$, *processor* i **do**

  for $j = (i-1)L' + 1, \cdots, iL', L + (i-1)M' + 1, \cdots, iM'$ **do**

    Calculate C_j using (5.22).

    Calculate B_j for using (5.23).

  end

end

Outputs:

The SDP elements C and B_i for $i = 1, \cdots, K$.

Algorithm 7: A parallel set-up algorithm for robust stability analysis over the multi-simplex

where recall that d_{p_i} is the degree of $\alpha_i^{h_i}$ in polynomial $P(\alpha)$ (see (5.9)), d_{a_i} is the degree of the variable $\alpha_i^{h_i}$ in the polynomial $A(\alpha)$ (see (5.10)), and d_2 is the Polya's exponent. Note that for the case of systems with uncertain parameters inside a simplex, (5.27) reduces to our results in Table 4.2. The number of operations versus the dimension of hypercube N is plotted in Figure 5.1 for different Polya's exponents $d := d_1 = d_2$. The figure shows that for the case of analysis over a hypercube, the number of operations grows exponentially with the dimension of the hypercube, whereas in analysis over a simplex, the number of operations grows polynomially. This is due to the fact that an N-dimensional hypercube is represented by the Cartesian product of N two-dimensional simplices.

5.4.2 Communication cost of the set-up algorithm:

In the worst case scenario, where each processor sends all of its assigned coefficients $\{H_{\{\mathcal{H}_N, \gamma_N\}}\}$ to other processors (a very rare situation), the communication cost per

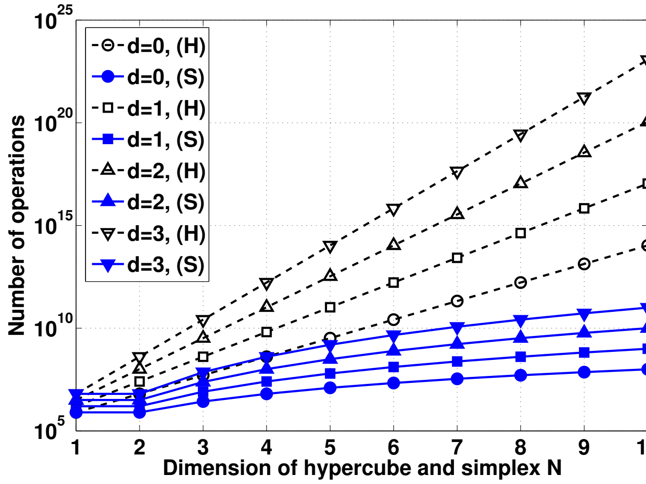


Figure 5.1: Number of operations versus dimension of the hypercube, for different Polya's exponents $d_1 = d_2 = d$. (H): hypercube and (S): simplex.

processor at each Polya's iteration is

$$\sim L_0 \left(\texttt{floor}\left(\frac{L}{N_c}\right) + \texttt{floor}\left(\frac{M}{N_c}\right) n^2 \right) \sim n^2 \prod_{i=1}^{N} l_i^{d_{pa_i}+d_2}, \tag{5.28}$$

assuming the number of processors $N_c = L_0$. Therefore, in this case, by increasing the number of processors, the communication cost per processor decreases and the scalability of the algorithm improves. for the case where the uncertain parameters belong to a simplex, (5.28) reduces to our results in Table 4.2. Again, it can be shown that the communication cost increases exponentially with the dimension of the hypercube, whereas in analysis over a simplex, the communication cost increases polynomially.

5.4.3 Speed-up and memory requirement of the set-up algorithm:

In the proposed set-up algorithm (Algorithm 7), calculation of the coefficients $\{\beta\}$ and $\{H\}$ is distributed among all of the available processors such that there exists no centralized computation. As a result, the algorithm can theoretically achieve ideal

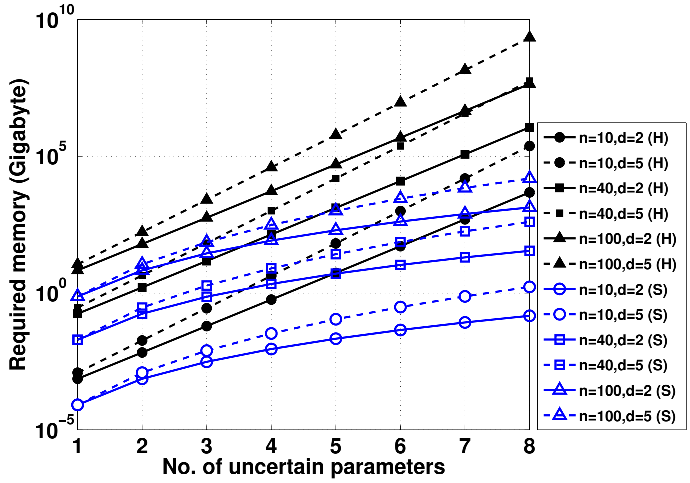


Figure 5.2: Required memory for the calculation of SDP elements vs. number of uncertain parameters in hypercube and simplex, for different state-space dimensions and Polya's exponents $d_1 = d_2$. (H): hypercube, (S): simplex.

(linear) speed-up. In other words, the speed-up

$$SP_N = \frac{N}{D + NS} = \frac{N}{1+0} = N,$$

Where $D = 1$ is the ratio of the operations performed by all processors simultaneously to the total operations performed simultaneously and sequentially, and S is the ratio of the operations performed sequentially to the total operations performed simultaneously and sequentially.

In Figure 5.2, we have shown the amount of memory required for storing the SDP elements versus the number of uncertain parameters in the unit hypercube and the unit simplex. The figure shows the required memory for different dimensions of the state-space n and Polya's exponents d. In all of the cases, we use $d_{p_i} = d_{a_i} = 1$ for $i = 1, \cdots, N$. The figure shows that for the case analysis over the hypercube, the required memory increases exponentially with the number of uncertain parameters, whereas for the case of the simplex the required memory increases polynomially. This is again because an N-dimensional hypercube is the Cartesian product of N

two-dimensional simplices, i.e., $\underbrace{\Delta^2 \times \cdots \times \Delta^2}_{\text{N times}}$.

5.5 Testing and validation

In this section, we evaluate the scalability and accuracy of our algorithm through numerical examples. In example 1, we evaluate the speed-up of our algorithm through numerical experiments. In examples 2 and 3, we evaluate the conservativeness of our algorithm and compare it to other methods in the literature.

5.5.1 Example 1: Evaluating speed-up

A parallel algorithm is scalable, if by using N_c processors it can solve a problem N_c times faster than solving the same problem using one processor. Thus, the speed-up of the ideal scalable algorithm is linear. To test the scalability of our algorithm, we run the algorithm using two random uncertain systems with state-space dimensions $n = 5$ and $n = 10$. The tests were performed on a linux-based Karlin cluster computer at Illinois Institute of Technology. In all of the runs, $D_p = [2, 2, 2, 2], D_a = [1, 1, 1, 1]$ and $\alpha \in \Phi^4$. Figure 5.3 shows the computation time of the algorithm versus the number of processors, for two different state-space dimensions and two different number of Polya's iterations (Polya's exponents $d = d_1 = d_2$). The linearity of the curves in all of the executions implies near-perfect scalability of the algorithm.

5.5.2 Example 2: Verifying robust stability over a hypercube

Consider the system $\dot{x}(t) = A(\alpha)x(t)$, where

$$A(\alpha) = A_0 + A_1\alpha_1^2 + A_2\alpha_1\alpha_2\alpha_3 + A_3\alpha_1^2\alpha_2\alpha_3^2,$$

$$\alpha_1 \in [-1, 1],\ \alpha_2 \in [-0.5, 0.5],\ \alpha_3 \in [-0.1, 0.1],$$

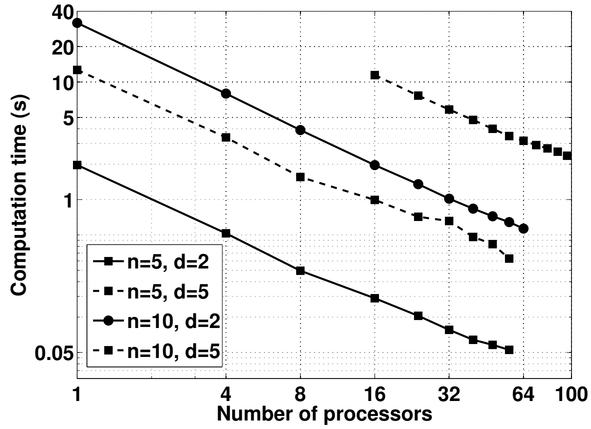


Figure 5.3: Execution time of the set-up algorithm vs. number of processors, for different state-space dimensions n and Polya's exponents

where

$$A_0 = \begin{bmatrix} -3.0 & 0 & -1.7 & 3.0 \\ -0.2 & -2.9 & -1.7 & -2.60 \\ 0.6 & 2.6 & -5.8 & -2.60 \\ -0.7 & 2.9 & -3.3 & -2.10 \end{bmatrix} \qquad A_1 = \begin{bmatrix} 2.2 & -5.4 & -0.8 & -2.2 \\ 4.4 & 1.4 & -3.0 & 0.8 \\ -2.4 & -2.2 & 1.4 & 6.0 \\ -2.4 & -4.4 & -6.4 & 0.18 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -8.0 & -13.5 & -0.5 & -3.0 \\ 18.0 & -2.0 & 0.5 & -11.5 \\ 5.5 & -10.0 & 3.5 & 9.0 \\ 13.0 & 7.5 & 5.0 & -4.0 \end{bmatrix} \qquad A_3 = \begin{bmatrix} 3.0 & 7.5 & 2.5 & -8.0 \\ 1.0 & 0.5 & 1.0 & 1.5 \\ -0.5 & -1.0 & 1.0 & 6.0 \\ -2.5 & -6.0 & 8.5 & 14.25 \end{bmatrix}.$$

The problem is to investigate asymptotic stability of this system for all α in the given intervals using Algorithm 7 and our solver in Algorithm 6. We first represented $A(\alpha)$ defined over the hypercube $[-1,1] \times [-0.5, 0.5] \times [-0.1, 0.1]$ by a multi-homogeneous polynomial $B(\beta, \eta)$ with $(\beta_i, \eta_i) \in \Delta^2$ and with the degree vector $D_b = [2, 1, 2]$. Then, in one Polya's iteration (i.e., $d_1 = d_2 = 1$) our algorithm found the following Lyapunov function as a certificate for asymptotic stability of the system.

$$V(x, \beta, \eta) = x^T P(\beta, \eta) x = x^T (\beta_1 (P_1 \beta_2 \beta_3 + P_2 \beta_2 \eta_3 + P_3 \eta_2 \beta_3 + P_4 \eta_2 \eta_3) \\ + \eta_1 (P_5 \beta_2 \beta_3 + P_6 \beta_2 \eta_3 + P_7 \eta_2 \beta_3 + P_8 \eta_2 \eta_3)) x,$$

where $\beta_1 = 0.5\alpha_1+0.5, \beta_2 = \alpha_2+0.5, \beta_3 = 5\alpha_3+0.5, \eta_1 = 1-\beta_1, \eta_2 = 1-\beta_2, \eta_3 = 1-\beta_3$ and

$$P_1 = \begin{bmatrix} 5.807 & 0.010 & -0.187 & -1.186 \\ 0.010 & 5.042 & -0.369 & 0.227 \\ -0.187 & -0.369 & 8.227 & -1.824 \\ -1.186 & 0.227 & -1.824 & 8.127 \end{bmatrix} P_2 = \begin{bmatrix} 7.409 & -0.803 & 1.804 & -1.594 \\ -0.803 & 6.016 & 0.042 & -0.538 \\ 1.804 & 0.042 & 7.894 & -1.118 \\ -1.594 & -0.538 & -1.118 & 8.590 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 6.095 & -0.873 & 0.512 & -1.125 \\ -0.873 & 5.934 & -0.161 & 0.503 \\ 0.512 & -0.161 & 7.417 & -0.538 \\ -1.125 & 0.503 & -0.538 & 6.896 \end{bmatrix} P_4 = \begin{bmatrix} 5.388 & 0.130 & -0.363 & -0.333 \\ 0.130 & 5.044 & -0.113 & -0.117 \\ -0.363 & -0.113 & 6.156 & -0.236 \\ -0.333 & -0.117 & -0.236 & 5.653 \end{bmatrix}$$

$$P_5 = \begin{bmatrix} 7.410 & -0.803 & 1.804 & -1.594 \\ -0.803 & 6.016 & 0.042 & -0.538 \\ 1.804 & 0.042 & 7.894 & -1.118 \\ -1.594 & -0.538 & -1.118 & 8.590 \end{bmatrix} P_6 = \begin{bmatrix} 5.807 & 0.010 & -0.187 & -1.186 \\ 0.010 & 5.042 & -0.369 & 0.227 \\ -0.187 & -0.369 & 8.227 & -1.824 \\ -1.186 & 0.227 & -1.824 & 8.127 \end{bmatrix}$$

$$P_7 = \begin{bmatrix} 5.388 & 0.130 & -0.363 & -0.333 \\ 0.130 & 5.044 & -0.113 & -0.117 \\ -0.363 & -0.113 & 6.156 & -0.236 \\ -0.333 & -0.117 & -0.236 & 5.653 \end{bmatrix} P_8 = \begin{bmatrix} 6.095 & -0.873 & 0.512 & -1.125 \\ -0.873 & 5.934 & -0.161 & 0.503 \\ 0.512 & -0.161 & 7.417 & -0.538 \\ -1.125 & 0.503 & -0.538 & 6.896 \end{bmatrix}.$$

5.5.3 Example 2: Evaluating accuracy

In this example, we used our algorithm to find lower bounds on $r^* = \max r$ such that $\dot{x}(t) = A(\alpha)x(t)$ with

$$A(\alpha) = A_0 + \sum_{i=1}^{4} A_i\alpha_i,$$

$$A_0 = \begin{bmatrix} -3.0 & 0 & -1.7 & 3.0 \\ -0.2 & -2.9 & -1.7 & -2.6 \\ 0.6 & 2.6 & -5.8 & -2.6 \\ -0.7 & 2.9 & -3.3 & -2.4 \end{bmatrix}, A_1 = \begin{bmatrix} 1.1 & -2.7 & -0.4 & -1.1 \\ 2.2 & 0.7 & -1.5 & 0.4 \\ -1.2 & -1.1 & 0.7 & 3.0 \\ -1.2 & -2.2 & -3.2 & -1.4 \end{bmatrix}, A_2 = \begin{bmatrix} 1.6 & 2.7 & 0.1 & 0.6 \\ -3.6 & 0.4 & -0.1 & 2.3 \\ -1.1 & 2 & -0.7 & -1.8 \\ -2.6 & -1.5 & -1.0 & 0.8 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} -0.6 & 1.5 & 0.5 & -1.6 \\ 0.2 & -0.1 & 0.2 & 0.3 \\ -0.1 & -0.2 & -0.2 & 1.2 \\ -0.5 & -1.2 & 1.7 & -0.1 \end{bmatrix}, A_4 = \begin{bmatrix} -0.4 & -0.1 & -0.3 & 0.1 \\ 0.1 & 0.3 & 0.2 & 0.0 \\ 0.0 & 0.2 & -0.3 & 0.1 \\ 0.1 & -0.2 & -0.2 & 0.0 \end{bmatrix}$$

is asymptotically stable for all $\alpha \in \{\alpha \in \mathbb{R}^4 : |\alpha_i| \leq r\}$. In Table 5.1, we have shown the computed lower bounds on r^* for different degree vectors D_p (degree vector of polynomial P in Theorem 17). In all of the cases, we set the Polya's exponents $d_1 = d_2 = 0$. For comparison, we have also included the lower-bounds computed by the methods in Bliman (2004a) and Chesi (2005) in Table 5.1.

Table 5.1: The lower-bounds on r^* computed by Algorithm 7 using different degree vector D_p and using methods in Bliman (2004a) and Chesi (2005).

	D_p =[0,0,0,0]	D_p =[0,1,0,1]	D_p =[1,0,1,0]	D_p =[1,1,1,1]	D_p =[2,2,2,2]	Bliman (2004a)	Chesi (2005)
Bound on r^*	0.494	0.508	0.615	0.731	0.840	0.4494	0.8739

Chapter 6

PARALLEL ALGORITHMS FOR NONLINEAR STABILITY ANALYSIS

6.1 Background and Motivation

One approach to stability analysis of nonlinear systems is the search for a decreasing Lyapunov function. For those systems with polynomial vector fields, Peet (2009) has shown that searching for polynomial Lyapunov functions is necessary and sufficient for stability on any bounded set. However, searching for a polynomial Lyapunov function which proves local stability requires enforcing positivity on a neighborhood of the equilibrium. Unfortunately, while we do have necessary and sufficient conditions for positivity of a polynomial (e.g. Tarski-Seidenberg's algorithm in Tarski (1951) and Artin's theorem in Artin (1927)), it has been shown that the general problem of determining whether a polynomial is positive is NP-hard (L. Blum and Smale (1998)).

Based on Artin's theorem, non-negativity of a polynomial is equivalent to existence of a representation in the form of sum of quotients of squared polynomials. If we leave off the quotient, the search for a Sum-of-Squares (SOS) is a common sufficient condition for positivity of a polynomial. The advantage of the SOS approach is that verifying the existence of an SOS representation is a semidefinite programming problem. This approach was first articulated in Parrilo (2000). SOS programming has been used extensively in stability analysis and control including stability analysis of nonlinear systems (Tan and Packard (2008)), robust stability analysis of switched and hybrid systems Prajna and Papachristodoulou (2003), and stability analysis of time-delay systems (Papachristodoulou *et al.* (2009)).

The downside to the use of SOS (with Positivstellensatz multipliers) for stability analysis of nonlinear systems with many states is computational complexity. Specifically, this approach requires us to set up and solve large SDPs. As an example, applying SOS method to find a degree 8 Lyapunov function for a nonlinear system with 10 states requires at least 900 GB of memory and more than 116 days as computation time on a single-core 2.5 GHz processor. Although Polya's algorithm implies similar complexity to SOS, as we showed in Section 4.3.3, the SDPs associated with Polya's algorithm possess a block-diagonal structure. This allowed us to develop parallel algorithms (see Algorithms 5, 6, and 7) for robust stability analysis of linear systems. Unfortunately, Polya's theorem cannot be used to represent polynomials which have zeros in the interior of the unit simplex (see Powers and Reznick (2006) for an elementary proof of this). From the same reasoning as in Powers and Reznick (2006) it follows that our multi-simplex version of Polya's theorem (See theorem 16) cannot be used to represent polynomials which have zeros in the interior of a multi-simplex/hypercube. Our proposed solution to this problem is to reformulate the nonlinear stability problem using only strictly positive forms. Specifically, we consider Lyapunov functions of the form $V(x) = x^T P(x) x$, where P is a strictly positive matrix-valued polynomial on the hypercube. This way, we can use our multi-simplex version of Polya's theorem to search for a polynomial $P(x)$ such that $P(x) > 0$ for all $x \in \Phi \setminus \{0\}$ and $\langle \nabla x^T P(x) x, f(x) \rangle < 0$ for all $x \in \Phi$ - hence proving asymptotic local stability of $\dot{x}(t) = f(x(t))$ for some $f \in \mathbb{R}[x]$.

Although Polya's algorithm has been generalized to positivity over simplices and hypercubes; as yet no further generalization to arbitrary convex polytopes exists. In order to perform analysis on more complicated geometries such as arbitrary convex polytopes, in this chapter, we look into Handelman's theorem (see Theorem 19). Some preliminary work on the use of Handelman's theorem and interval evaluation for Lyapunov functions on the hypercube has been suggested in Sankaranarayanan *et al.* (2013) and has also been applied to robust stability of positive linear systems in Briat (2013). One difficulty in using Handelman's theorem in stability analysis is that then theorem cannot be readily used to represent polynomials which have zeros in the interior of a given polytope. To see this, suppose a polynomial g (g is not identically zero) is zero at $x = a$, where a is in the interior of a polytope

$$\Gamma^K := \{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \cdots, K\}.$$

Suppose there exist $b_\alpha \geq 0, \alpha \in \mathbb{N}^K$ such that for some $d \in \mathbb{N}$,

$$g(x) = \sum_{\alpha \in \mathbb{N}^K : \|\alpha_i\|_1 \leq d} b_\alpha (w_1^T x + u_1)^{\alpha_1} \cdots (w_K^T x + u_K)^{\alpha_K}.$$

Then,

$$g(a) = \sum_{\alpha \in \mathbb{N}^K : \|\alpha_i\|_1 \leq d} b_\alpha (w_1^T a + u_1)^{\alpha_1} \cdots (w_K^T a + u_K)^{\alpha_K} = 0.$$

From the assumption $a \in \text{int}(\Gamma^K)$ it follows that $w_i^T a + u_i > 0$ for $i = 1, \cdots, K$. Hence $b_\alpha < 0$ for at least one $\alpha \in \{\alpha \in \mathbb{N}^K : \|\alpha\|_1 \leq d\}$. This contradicts with the assumption that all $b_\alpha \geq 0$. Based on the this reasoning, one cannot readily use Handelman's theorem to search for a polynomial V such that $V(x) > 0$ for all $x \in \Gamma^K \setminus \{0\}$ and $V(0) = 0$.

6.1.1 Our contributions

In this chapter, we consider a new approach to the use of Handelman's theorem for computing regions of attraction of stable equilibria by constructing piecewise-polynomial Lyapunov functions over arbitrary convex polytopes. Specifically, we decompose a given convex polytope into a set of convex sub-polytopes that share a common vertex at the origin. Then, on each sub-polytope, we use Handelman's conditions to define linear programming constraints. Additional constraints are then proposed which ensure continuity of the Lyapunov function over the entire polytope.

We then show the resulting algorithm has polynomial complexity in the number of states and compare this complexity with algorithms based on SOS and Polya's theorem. Finally, we evaluate the accuracy of our algorithm by numerically approximating the domain of attraction of two nonlinear dynamical systems.

6.2 Definitions and Notation

In this section, we present/review notations and definitions of convex polytopes, facets of polytopes, decompositions and Handelman bases.

Definition 1. *(Convex Polytope) Given the set of vertices* $P := \{p_i \in \mathbb{R}^n, i = 1, \cdots, K\}$*, define the* ***convex polytope*** Γ_P *as*

$$\Gamma_P := \{x \in \mathbb{R}^n : x = \sum_{i=1}^{K} \mu_i p_i : \mu_i \in [0,1] \text{ and } \sum_{i=1}^{K} \mu_i = 1\}.$$

Every convex polytope can be represented as

$$\Gamma^K := \{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \cdots, K\},$$

for some $w_i \in \mathbb{R}^n, u_i \in \mathbb{R}, i = 1, \cdots, K$. Throughout the chapter, every polytope that we use contains the origin. Moreover, for brevity, we will drop the superscript K in Γ^K.

Definition 2. *Given a bounded polytope of the form* $\Gamma := \{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \cdots, K\}$*, we call*

$$\zeta^i(\Gamma) := \left\{x \in \mathbb{R}^n : w_i^T x + u_i = 0 \text{ and } w_j^T x + u_j \geq 0 \quad \text{for } j \in \{1, \cdots, K\}\right\}$$

the $i-$***th facet*** *of* Γ *if* $\zeta^i(\Gamma) \neq \emptyset$*.*

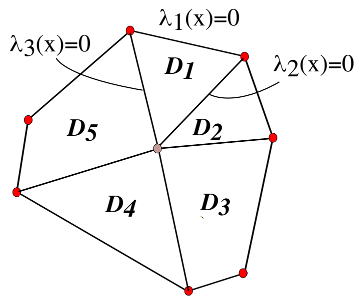


Figure 6.1: An illustration of a D-decomposition of a 2D polytope. $\lambda_i(x) := h_{i,j}^T x + g_{i,j}$ for $j = 1, \cdots, m_i$.

Definition 3. *($D-$decomposition) Given a bounded polytope of the form* $\Gamma := \{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \cdots, K\}$*, we call* $D_\Gamma := \{D_i\}_{i=1,\cdots,L}$ *a $D-$**decomposition** of* Γ *if*

$$D_i := \left\{x \in \mathbb{R}^n : h_{i,j}^T x + g_{i,j} \geq 0, j = 1, \cdots, m_i\right\}$$

for some $h_{i,j} \in \mathbb{R}^n$*,* $g_{i,j} \in \mathbb{R}$*, such that* $\cup_{i=1}^L D_i = \Gamma$*,* $\cap_{i=1}^L D_i = \{0\}$ *and* $int(D_i) \cap int(D_j) = \emptyset$*.*

In Figure 6.1, we have illustrated a D-decomposition of a two-dimensional polytope.

Definition 4. *(The Handelman basis associated with a polytope) Given a polytope of the form*

$$\Gamma := \left\{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0,\ i = 1, \cdots, K\right\},$$

*we define the set of **Handelman bases**, indexed by*

$$\alpha \in E_{d,K} := \left\{\alpha \in \mathbb{N}^K : |\alpha|_1 \leq d\right\} \tag{6.1}$$

as

$$\Theta_d(\Gamma) := \left\{\rho_\alpha(x) : \rho_\alpha(x) = \prod_{i=1}^K (w_i^T x + u_i)^{\alpha_i},\ \alpha \in E_{d,K}\right\}.$$

Definition 5. *(Restriction of a polynomial to a facet) Given a polytope of the form* $\Gamma := \{x \in \mathbb{R}^n : w_i^T x + u_i,\ i = 1, \cdots, K\}$*, and a polynomial* $P(x)$ *of the form*

$$P(x) = \sum_{\alpha \in E_{d,K}} b_\alpha \prod_{i=1}^{K} (w_i^T x + u_i)^{\alpha_i},$$

define the ***restriction of*** $P(x)$ ***to the*** k***-th facet of*** Γ *as the function*

$$P|_k(x) := \sum_{\alpha \in E_d : \alpha_k = 0} b_\alpha \prod_{i=1}^{K} (w_i^T x + u_i)^{\alpha_i}.$$

We will use the maps defined below in future sections.

Definition 6. *Given* $w_i, h_{i,j} \in \mathbb{R}^n$ *and* $u_i, g_{i,j} \in \mathbb{R}$*, let* Γ *be a convex polytope as defined in Definition 1 with* $D-$*decomposition* $D_\Gamma := \{D_i\}_{i=1,\cdots,L}$ *as defined in Definition 3, and let* $\lambda^{(k)}$, $k = 1, \cdots, B$ *be the elements of* $E_{d,n}$*, as defined in* (6.1)*, for some* $d, n, \in \mathbb{N}$*. For any* $\lambda^{(k)} \in E_{d,n}$*, let* $p_{\{\lambda^{(k)},\alpha,i\}}$ *be the coefficient of* $b_{i,\alpha} x^{\lambda^{(k)}}$ *in*

$$P_i(x) := \sum_{\alpha \in E_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_j}. \tag{6.2}$$

Let N_i *be the cardinality of* E_{d,m_i}*, and denote by* $b_i \in \mathbb{R}^{N_i}$ *the vector of all coefficients* $b_{i,\alpha}$.

Define $F_i : \mathbb{R}^{N_i} \times \mathbb{N} \to \mathbb{R}^B$ *as*

$$F_i(b_i, d) := \left[\sum_{\alpha \in E_{d,m_i}} p_{\{\lambda^{(1)},\alpha,i\}} b_{i,\alpha}, \ \cdots \ , \sum_{\alpha \in E_{d,m_i}} p_{\{\lambda^{(B)},\alpha,i\}} b_{i,\alpha} \right]^T \tag{6.3}$$

for $i = 1, \cdots, L$*. In other words,* $F_i(b_i, d)$ *is the vector of the coefficients of* $P_i(x)$ *after expansion.*

Define $H_i : \mathbb{R}^{N_i} \times \mathbb{N} \to \mathbb{R}^Q$ *as*

$$H_i(b_i, d) := \left[\sum_{\alpha \in E_{d,m_i}} p_{\{\delta^{(1)},\alpha,i\}} b_{i,\alpha} \ , \ \cdots \ , \sum_{\alpha \in E_{d,m_i}} p_{\{\delta^{(Q)},\alpha,i\}} b_{i,\alpha} \right]^T \tag{6.4}$$

for $i = 1, \cdots, L$, where we have denoted the elements of $\{\delta \in \mathbb{N}^n : \delta = 2e_j$ for $j = 1, \cdots, n\}$ by $\delta^{(k)}, k = 1, \cdots, Q$, where e_j are the canonical basis for $\mathbb{N}^n$. In other words, $H_i(b_i, d)$ is the vector of coefficients of square terms of $P_i(x)$ after expansion.

Define $J_i : \mathbb{R}^{N_i} \times \mathbb{N} \times \{1, \cdots, m_i\} \to \mathbb{R}^B$ as

$$J_i(b_i, d, k) := \left[\sum_{\substack{\alpha \in E_{d,m_i} \\ \alpha_k = 0}} p_{\{\lambda^{(1)},\alpha,i\}} b_{i,\alpha} \ \cdots \ , \sum_{\substack{\alpha \in E_{d,m_i} \\ \alpha_k = 0}} p_{\{\lambda^{(B)},\alpha,i\}} b_{i,\alpha} \right]^T \tag{6.5}$$

for $i = 1, \cdots, L$. In other words, $J_i(b_i, d, k)$ is the vector of coefficients of $P_i|_k(x)$ after expansion.

Given a polynomial vector field $f(x)$ of degree d_f, define $G_i : \mathbb{R}^{N_i} \times \mathbb{N} \to \mathbb{R}^Z$ as

$$G_i(b_i, d) := \left[\sum_{\alpha \in E_{d,m_i}} s_{\{\eta^{(1)},\alpha,i\}} b_{i,\alpha} \ , \ \cdots \ , \sum_{\alpha \in E_{d,m_i}} s_{\{\eta^{(P)},\alpha,i\}} b_{i,\alpha} \right]^T \tag{6.6}$$

for $i = 1, \cdots, L$, and where we have denoted the elements of $E_{d+d_f-1,n}$ by $\eta^{(k)}$, $k = 1, \cdots, Z$. For any $\eta^{(k)} \in E_{d+d_f-1,n}$, we define $s_{\{\eta^{(k)},\alpha,i\}}$ as the coefficient of $b_{i,\alpha} x^{\eta^{(k)}}$ in $\langle \nabla P_i(x), f(x) \rangle$, where $P_i(x)$ is defined in (6.2). In other words, $G_i(b_i, d)$ is the vector of coefficients of $\langle \nabla P_i(x), f(x) \rangle$.

Define $R_i(b_i, d) : \mathbb{R}^{N_i} \times \mathbb{N} \to \mathbb{R}^C$ as

$$R_i(b_i, d) := \left[b_{i,\beta^{(1)}} \ , \ \cdots \ , \ b_{i,\beta^{(C)}} \right]^T, \tag{6.7}$$

for $i = 1, \cdots, L$, where we have denoted the elements of

$$S_{d,m_i} := \{\beta \in E_{d,m_i} : \beta_j = 0 \text{ for } j \in \{j \in \mathbb{N} : g_{i,j} = 0\}\}$$

by $\beta^{(k)}$, $k = 1, \cdots, C$. Consider P_i in the Handelman basis $\Theta_d(\Gamma)$. Then, $R_i(b_i, d)$ is the vector of coefficients of monomials of P_i which are nonzero at the origin.

It can be shown that the maps F_i, H_i, J_i, G_i and R_i are affine in b_i.

Definition 7. *(Upper Dini Derivative) Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a continuous map. Then, define the upper Dini derivative of a function $V : \mathbb{R}^n \to \mathbb{R}$ in the direction $f(x)$ as*

$$D^+(V(x), f(x)) = \limsup_{h \to 0^+} \frac{V(x + hf(x)) - V(x)}{h}.$$

It can be shown that for a continuously differentiable $V(x)$,

$$D^+(V(x), f(x)) = \langle \nabla V(x), f(x) \rangle.$$

6.3 Statement of the stability problem

We address the problem of local stability of nonlinear systems of the form

$$\dot{x}(t) = f(x(t)), \tag{6.8}$$

about the zero equilibrium, where $f : \mathbb{R}^n \to \mathbb{R}^n$. We use the following Lyapunov stability condition.

Theorem 18. *For any $\Omega \subset \mathbb{R}^n$ with $0 \in \Omega$, suppose there exists a continuous function $V : \mathbb{R}^n \to \mathbb{R}$ and continuous positive definite functions W_1, W_2, W_3,*

$$W_1(x) \leq V(x) \leq W_2(x) \text{ for } x \in \Omega \text{ and}$$

$$D^+(V(x), f(x)) \leq -W_3(x) \text{ for } x \in \Omega,$$

then System (6.8) is asymptotically stable on $\{x : \{y : V(y) \leq V(x)\} \subset \Omega\}$.

In this paper, we construct piecewise-polynomial Lyapunov functions which may not have classical derivatives. As such, we use Dini derivatives which are known to exist for piecewise-polynomial functions.

Problem statement: Given the vertices $p_i \in \mathbb{R}^n, i = 1, \cdots, K$, we would like to find the largest positive s such that there exists a polynomial $V(x)$ where $V(x)$ satisfies the conditions of Theorem 18 on the convex polytope

$$\left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^{K} \mu_i p_i : \mu_i \in [0, s] \text{ and } \sum_{i=1}^{K} \mu_i = s \right\}.$$

Given a convex polytope, the following result (Handelman (1988a)) parameterizes the set of polynomials which are positive on that polytope using the positive orthant.

Theorem 19. *(Handelman's Theorem) Given $w_i \in \mathbb{R}^n, u_i \in \mathbb{R}, i = 1, \cdots, K$, let Γ be a convex polytope as defined in definition 1. If polynomial $P(x) > 0$ for all $x \in \Gamma$, then there exist $b_\alpha \geq 0$, $\alpha \in \mathbb{N}^K$ such that for some $d \in \mathbb{N}$,*

$$P(x) := \sum_{\alpha \in E_{d,K}} b_\alpha \prod_{ji=1}^{K} (w_i^T x + u_i)^{\alpha_i}.$$

Given a D-decomposition $D_\Gamma := \{D_i\}_{i=1,\cdots,L}$ of the form

$$D_i := \left\{ x \in \mathbb{R}^n : h_{i,j}^T x + g_{i,j} \geq 0, j = 1, \cdots, m_i \right\}$$

of some polytope Γ, we parameterize a cone of piecewise-polynomial Lyapunov functions which are positive on Γ as

$$V(x) = V_i(x) := \sum_{\alpha \in E_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_j} \quad \text{for } x \in D_i \text{ and } i = 1, \cdots, L.$$

We will use a similar parameterization of piecewise-polynomials which are negative on Γ in order to enforce negativity of the derivative of the Lyapunov function. We will also use linear equality constraints to enforce continuity of the Lyapunov function.

6.4 Expressing the stability problem as a linear program

We first present some lemmas necessary for the proof of our main result. The following lemma provides a sufficient condition for a polynomial represented in the Handelman basis to vanish at the origin ($V(0) = 0$).

Lemma 1. *Let $D_\Gamma := \{D_i\}_{i=1,\cdots,L}$ be a D-decomposition of a convex polytope Γ, where*

$$D_i := \left\{ x \in \mathbb{R}^n : h_{i,j}^T x + g_{i,j} \geq 0, j = 1, \cdots, m_i \right\}.$$

For each $i \in \{1 \cdots, L\}$*, let*

$$P_i(x) := \sum_{\alpha \in E_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_j},$$

N_i *be the cardinality* E_{d,m_i} *as defined in* (6.1)*, and let* $b_i \in \mathbb{R}^{N_i}$ *be the vector of the coefficients* $b_{i,\alpha}$ *. Consider* $R_i : \mathbb{R}^{N_i} \times \mathbb{N} \to \mathbb{R}^C$ *as defined in* (6.7)*. If* $R_i(b_i, d) = \mathbf{0}$*, then* $P_i(x) = 0$ *for all* $i \in \{1 \cdots, L\}$*.*

Proof. We can write

$$P_i(x) = \sum_{\alpha \in E_{d,m_i} \setminus S_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_i} + \sum_{\alpha \in S_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_i},$$

where

$$S_{d,m_i} := \{\alpha \in E_{d,m_i} : \alpha_j = 0 \text{ for } j \in \{j \in \mathbb{N} : g_{i,j} = 0\}\}.$$

By the definitions of E_{d,m_i} and S_{d,m_i}, we know that for each $\alpha \in \ E_{d,m_i} \setminus S_{d,m_i}$ for $i \in \{1, \cdots, L\}$, there exists at least one $j \in \{1, \cdots, m_i\}$ such that $g_{i,j} = 0$ and $\alpha_k > 0$. Thus, at $x = 0$,

$$\sum_{\alpha \in E_{d,m_i} \setminus S_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_i} = 0 \quad \text{for all } i \in \{1, \cdots, L\}.$$

Recall the definition of the map R_i from (6.7). Since $R_i(b_i, d) = \mathbf{0}$ for each $i \in \{1, \cdots, L\}$, it follows from that $b_{i,\alpha} = 0$ for each $\alpha \in S_{d,m_i}$ and $i \in \{1, \cdots, L\}$. Thus,

$$\sum_{\alpha \in S_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_i} = 0 \quad \text{for all } i \in \{1, \cdots, L\}.$$

Thus, $P_i(0) = 0$ for all $i \in \{1, \cdots, L\}$. $\square$

This Lemma provides a condition which ensures that a piecewise-polynomial function on a D-decomposition is continuous.

Lemma 2. *Let $D_\Gamma := \{D_i\}_{i=1,\cdots,L}$ be a D-decomposition of a polytope Γ, where*

$$D_i := \{x \in \mathbb{R}^n : h_{i,j}^T x + g_{i,j} \geq 0, j = 1, \cdots, m_i\}.$$

For each $i \in \{1 \cdots, L\}$, let

$$P_i(x) := \sum_{\alpha \in E_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_j},$$

N_i be the cardinality of E_{d,m_i} as defined in (6.1), and let $b_i \in \mathbb{R}^{N_i}$ be the vector of the coefficients $b_{i,\alpha}$. Given $i, j \in \{1, \cdots, L\}, i \neq j$, let

$$\Lambda_{i,j}(D_\Gamma) := \left\{k, l \in \mathbb{N} : k \in \{1, \cdots, m_i\}, l \in \{1, \cdots, m_j\} : \zeta^k(D_i) \neq \emptyset \text{ and } \zeta^k(D_i) = \zeta^l(D_j)\right\}. \tag{6.9}$$

Consider $J_i : \mathbb{R}^{N_i} \times \mathbb{N} \times \{1 \cdots, m_i\} \to \mathbb{R}^B$ as defined in (6.5). If

$$J_i(b_i, d, k) = J_j(b_j, d, l)$$

for all $i, j \in \{1, \cdots, L\}$, $i \neq j$ and $k, l \in \Lambda_{i,j}(D_\Gamma)$, then the piecewise-polynomial function

$$P(x) = P_i(x), \quad \text{for } x \in D_i, \ i = 1, \cdots, L$$

is continuous for all $x \in \Gamma$.

Proof. From (6.5), $J_i(b_i, d, k)$ is the vector of coefficients of $P_i|_k(x)$ after expansion. Therefore, if

$$J_i(b_i, d, k) = J_j(b_j, d, l) \text{ for all } i, j \in \{1, \cdots, L\}, \ i \neq j$$

and $(k, l) \in \Lambda_{i,j}(D_\Gamma)$, then

$$P_i|_k(x) = P_j|_l(x) \text{ for all } i, j \in \{1, \cdots, L\}, \ i \neq j \text{ and } (k, l) \in \Lambda_{i,j}(D_\Gamma). \tag{6.10}$$

On the other hand, from definition 5, it follows that for any $i \in \{1, \cdots, L\}$ and $k \in \{1, \cdots, m_i\}$,

$$P_i|_k(x) = P_i(x) \text{ for all } x \in \zeta^k(D_i). \tag{6.11}$$

Furthermore, from the definition of $\Lambda_{i,j}(D_\Gamma)$, we know that

$$\zeta^k(D_i) = \zeta^l(D_j) \subset D_i \cap D_j \tag{6.12}$$

for any $i, j \in \{1 \cdots, L\}$, $i \neq j$ and any $(k,l) \in \Lambda_{i,j}(D_\Gamma)$. Thus, from (6.10), (6.11) and (6.12), it follows that for any $i, j \in \{1, \cdots, L\}$, $i \neq j$, we have $P_i(x) = P_j(x)$ for all $x \in D_i \cap D_j$. Since for each $i \in \{1, \cdots, L\}$, $P_i(x)$ is continuous on D_i and for any $i, j \in \{1 \cdots, L\}$, $i \neq j$, $P_i(x) = P_j(x)$ for all $x \in D_i \cap D_j$, we conclude that the piecewise polynomial function

$$P(x) = P_i(x) \quad x \in D_i, i = 1, \cdots, L$$

is continuous for all $x \in \Gamma$. $\square$

Theorem 20. *(Main Result) Let d_f be the degree of the polynomial vector field $f(x)$ of System* (6.8)*. Given w_i, $h_{i,j} \in \mathbb{R}^n$ and u_i, $g_{i,j} \in \mathbb{R}$, define the polytope*

$$\Gamma := \{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i = 1, \cdots, K\},$$

with D-decomposition $D_\Gamma := \{D_i\}_{i=1,\cdots,L}$, where

$$D_i := \{x \in \mathbb{R}^n : h_{i,j}^T x + g_{i,j} \geq 0, j = 1, \cdots, m_i\}.$$

Let N_i be the cardinality of E_{d,m_i}, as defined in (6.1) *and let M_i be the cardinality of E_{d+d_f-1,m_i}. Consider the maps R_i, H_i, F_i, G_i, and J_i as defined in definition 6, and $\Lambda_{i,j}(D_\Gamma)$ as defined in* (6.9) *for $i, j \in \{1, \cdots, L\}$. If there exists $d \in \mathbb{N}$ such that*

$\max \gamma$ *in the linear program (LP),*

$$
\begin{aligned}
&\max_{\gamma \in \mathbb{R}, b_i \in \mathbb{R}^{N_i}, c_i \in \mathbb{R}^{M_i}} \quad \gamma \\
&\textit{subject to} \\
&b_i \geq \mathbf{0} && \textit{for } i = 1, \cdots, L \\
&c_i \leq \mathbf{0} && \textit{for } i = 1, \cdots, L \\
&R_i(b_i, d) = \mathbf{0} && \textit{for } i = 1, \cdots, L \\
&H_i(b_i, d) \geq \mathbf{1} && \textit{for } i = 1, \cdots, L \\
&H_i(c_i, d + d_f - 1) \leq -\gamma \cdot \mathbf{1} && \textit{for } i = 1, \cdots, L \\
&G_i(b_i, d) = F_i(c_i, d + d_f - 1) && \textit{for } i = 1, \cdots, L \\
&J_i(b_i, d, k) = J_j(b_j, d, l) && \textit{for } i, j = 1, \cdots, L \textit{ and } k, l \in \Lambda_{i,j}(D_\Gamma)
\end{aligned} \tag{6.13}
$$

is positive, then the origin is an asymptotically stable equilibrium for System 6.8. Furthermore,

$$
V(x) = V_i(x) = \sum_{\alpha \in E_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_j} \textit{ for } x \in D_i,\ i = 1, \cdots, L
$$

with $b_{i,\alpha}$ as the elements of b_i, is a piecewise polynomial Lyapunov function proving stability of System (6.8)*.*

Proof. Let us choose

$$
V(x) = V_i(x) = \sum_{\alpha \in E_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_j} \text{ for } x \in D_i,\ i = 1, \cdots, L
$$

In order to show that $V(x)$ is a Lyapunov function for system 6.8, we need to prove the following:

1. $V_i(x) \geq x^T x$ for all $x \in D_i,\ i = 1, \cdots, L$,

2. $D^+(V_i(x), f(x)) \leq -\gamma\, x^T x$ for all $x \in D_i,\ i = 1, \cdots, L$ and for some $\gamma > 0$,

3. $V(0) = 0$,

4. $V(x)$ is continuous on Γ.

Then, by Theorem 18, it follows that System (6.8) is asymptotically stable at the origin. Now, let us prove items (1)-(4). For some $d \in \mathbb{N}$, suppose $\gamma > 0$, b_i and c_i for $i = 1, \cdots, L$ is a solution to linear program (6.13).

Item 1. First, we show that $V_i(x) \geq x^T x$ for all $x \in D_i$, $i = 1, \cdots, L$. From the definition of the D-decomposition in the theorem statement, $h_{i,j}^T x + g_{i,j} \geq 0$, for all $x \in D_i$, $j = 1, \cdots, m_i$. Furthermore, $b_i \geq \mathbf{0}$. Thus,

$$V_i(x) := \sum_{\alpha \in E_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_j} \geq 0 \tag{6.14}$$

for all $x \in D_i \backslash$, $i = 1, \cdots, L$. From (6.4), $H_i(b_i, d) \geq \mathbf{1}$ for each $i = 1, \cdots, L$ implies that all the coefficients of the expansion of $x^T x$ in $V_i(x)$ are greater than 1 for $i = 1, \cdots, L$. This, together with (6.14), prove that $V_i(x) \geq x^T x$ for all $x \in D_i$, $i = 1, \cdots, L$.

Item 2. Next, we show that $D^+(V_i(x), f(x)) \leq -\gamma x^T x$ for all $x \in D_i$, $i = 1, \cdots, L$. For $i = 1, \cdots, L$, let us refer the elements of c_i as $c_{i,\beta}$, where $\beta \in E_{d+d_f-1,m_i}$. From (6.13), $c_i \leq \mathbf{0}$ for $i = 1, \cdots, L$. Furthermore, since $h_{i,j}^T x + g_{i,j} \geq 0$ for all $x \in D_i$, it follows that

$$Z_i(x) = \sum_{\beta \in E_{d+d_f-1}} c_{\beta,i} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\beta_j} \leq 0 \tag{6.15}$$

for all $x \in D_i$, $i = 1, \cdots, L$. From (6.4), $H_i(c_i, d + d_f - 1) \leq -\gamma \cdot \mathbf{1}$ for $i = 1, \cdots, L$ implies that all the coefficients of the expansion of $x^T x$ in $Z_i(x)$ are less than $-\gamma$ for $i = 1, \cdots, L$. This, together with (6.15), prove that $Z_i(x) \leq -\gamma x^T x$ for all $x \in D_i$, for $i = 1, \cdots, L$. Lastly, by the definitions of the maps G_i and F_i in (6.6) and (6.3), if $G_i(b_i, d) = F_i(c_i, d + d_f - 1)$, then

$$\langle \nabla V_i(x), f(x) \rangle = Z_i(x) \leq -\gamma x^T x \;\text{ for all } x \in D_i \;\text{ and }\; i \in \{1 \cdots, L\}.$$

Since $D^+(V_i(x), f(x)) = \langle \nabla V_i(x), f(x) \rangle$ for all $x \in D_i$, it follows that $D^+(V_i(x), f(x)) \leq -\gamma x^T x$ for all $x \in D_i,\ i \in \{1 \cdots, L\}$.

Item 3. Now, we show that $V(0) = 0$. By Lemma 1, $R_i(b_i, d) = \mathbf{0}$ implies $V_i(0) = 0$ for each $i \in \{1, \cdots, L\}$.

Item 4. Finally, we show that $V(x)$ is continuous for $x \in \Gamma$. By Lemma 2, $J_i(b_i, d, k) = J_j(b_j, d, l)$ for all $i, j \in \{1, \cdots, L\},\ k, l \in \Lambda_{i,j}(D_\Gamma)$ implies that $V(x)$ is continuous for all $x \in \Gamma$. □

Using Theorem 20, we define Algorithm 8 to search for piecewise-polynomial Lyapunov functions to verify local stability of system (6.8) on convex polytopes. We have provided a Matlab implementation for Algorithm 8 at: `www.sites.google.com/a/asu.edu/kamyar/Software`.

6.5 Computational complexity analysis

In this section, we analyze and compare the complexity of the LP in (6.13) with the complexity of the SDPs associated with Polya's algorithm in Kamyar and Peet (2013) and an SOS approach using Positivstellensatz multipliers. For simplicity, we consider Lyapunov functions defined on a hypercube centered at the origin. Note that we make frequent use of the formula

$$N_{vars} := \sum_{i=0}^{d} \frac{(i + K - 1)!}{i!(K - 1)!},$$

which gives the number of basis functions in $\Theta_d(\Gamma)$ for a convex polytope Γ with K facets.

Inputs:

Vertices of the polytope: p_i for $i = 1, \cdots, K$.

$h_{i,j}$ and $g_{i,j}$ for $i = 1, \cdots, K$ and $j = 1, \cdots, m_i$.

Coefficients and degree of the polynomial vector field in (6.8).

Maximum degree of the Lyapunov function: d_{max}

while $d < d_{max}$ **do**

  if *the LP defined in* (6.13) *is feasible* **then**

    Break the while loop.

  else

    Set $d = d + 1$

  end

end

Outputs:

If the LP in (6.13) is feasible, then the output is the coefficients $b_{i,\alpha}$ of the Lyapunov function

$$V(x) = V_i(x) = \sum_{\alpha \in E_{d,m_i}} b_{i,\alpha} \prod_{j=1}^{m_i} (h_{i,j}^T x + g_{i,j})^{\alpha_j} \text{ for } x \in D_i, \ i = 1, \cdots, L$$

Algorithm 8: Search for piecewise polynomial Lyapunov functions using Handelman's theorem

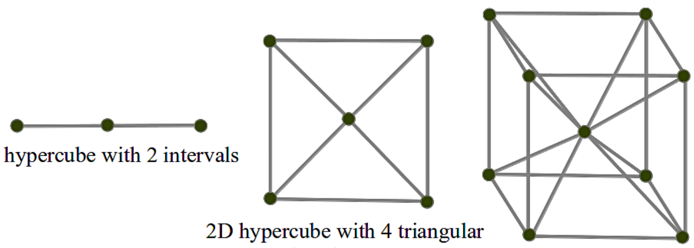


Figure 6.2: Decomposition of the hypercube in $1-$,$2-$ and $3-$dimensions

6.5.1 Complexity of the LP associated with Handelman's representation

Consider the following assumption on our $D-$decomposition.

Assumption 1. *We perform the analysis on an $n-$dimensional hypercube, centered at the origin. The hypercube is decomposed into $L = 2n$ sub-polytopes such that the i-th sub-polytope has $m = 2n - 1$ facets. Figure 6.2 shows the $1-$, $2-$ and $3-$dimensional decomposed hypercube.*

Let n be the number of states in System (6.8). Let d_f be the degree of the polynomial vector field in System (6.8). Suppose we use Algorithm 1 to search for a Lyapunov function of degree d_V. Then, the number of decision variables in the LP is

$$N^H_{vars} = L\left(\sum_{d=0}^{d_V}\frac{(d+m-1)!}{d!(m-1)!} + \sum_{d=0}^{d_V+d_f-1}\frac{(d+m-1)!}{d!(m-1)!} - (d_V+1)\right) \tag{6.16}$$

where the first term is the number of $b_{i,\alpha}$ coefficients, the second term is the number of $c_{i,\beta}$ coefficients and the third term is the dimension of $R_i(b_i, d)$ in (6.13). By substituting for L and m in (6.16), from Assumption 1 we have

$$N^H_{vars} = 2n\left(\sum_{d=0}^{d_V}\frac{(d+2n-2)!}{d!(2n-2)!} + \sum_{d=0}^{d_V+d_f-1}\frac{(d+2n-2)!}{d!(2n-2)!} - d_V - 1\right).$$

Then, for large number of states, i.e., large n,

$$N^H_{vars} \sim 2n\left((2n-2)^{d_V} + (2n-2)^{d_V+d_f-1}\right) \sim n^{d_V+d_f}.$$

Meanwhile, the number of constraints in the LP is

$$N_{cons}^H = N_{vars}^H + L\left(\sum_{d=0}^{d_V} \frac{(d+n-1)!}{d!(n-1)!} + \sum_{d=0}^{d_V+d_f-1} \frac{(d+n-1)!}{d!(n-1)!}\right), \tag{6.17}$$

where the first term is the total number of inequality constraints associated with the positivity of b_i and negativity of c_i, the second term is the number of equality constraints on the coefficients of the Lyapunov function required to ensure continuity ($J_i(b_i, d, k) = J_j(b_j, d, l)$ in the LP (6.13)) and the third term is the number of equality constraints associated with negativity of the Lie derivative of the Lyapunov function ($G_i(b_i, d) = F_i(c_i, d + d_f - 1)$ in the LP (6.13)). By substituting for L in (6.17), from Assumption 1 for large n we get

$$N_{cons}^H \sim n^{d_V+d_f} + 2n(n^{d_V} + n^{d_V+d_f-1}) \sim n^{d_V+d_f}.$$

The complexity of an LP using interior-point algorithms is approximately $O(N_{vars}^2 N_{cons})$ (Boyd and Vandenberghe (2004)). Therefore, the computational cost of solving the LP (6.13) is

$$\sim n^{3(d_V+d_f)}.$$

6.5.2 Complexity of the SDP associated with Polya's algorithm

Recall our approach in Section 5.3 for applying Polya's algorithm to analyze stability over hypercubes. In Kamyar and Peet (2013), we used the same approach to construct Lyapunov functions for nonlinear ODEs with polynomial vector fields. In particular, this approach uses semi-definite programming to search for the coefficients of a matrix-valued polynomial $P(x)$ which defines a Lyapunov function as $V(x) = x^T P(x) x$. Using a similar complexity analysis as in 5.4, we determine that the number of decision variables in the associated SDP is

$$N_{vars}^P = \frac{n(n+1)}{2} \sum_{d=0}^{d_V-2} \frac{(d+n-1)!}{d!(n-1)!}.$$

The number of constraints in the SDP is

$$N_{cons}^{P} = \frac{n(n+1)}{2}\left((d_V+e-1)^n + (d_V+d_f+e-2)^n\right),$$

where here we have denoted Polya's exponent by e. Then, for large n, $N_{vars}^{P} \sim n^{d_V}$ and $N_{cons}^{P} \sim (d_V+d_f+e-2)^n$. Since solving an SDP with an interior-point algorithm typically requires $O(N_{cons}^3 + N_{var}^3 N_{cons} + N_{var}^2 N_{cons}^2)$ operations (Boyd and Vandenberghe (2004)), the computational cost of solving the SDP associated with Polya's algorithm is estimated as

$$\sim (d_V+d_f+e-2)^{3n}.$$

6.5.3 Complexity of the SDP associated with SOS algorithm

To find a Lyapunov function for (6.8) over the polytope

$$\Gamma = \left\{x \in \mathbb{R}^n : w_i^T x + u_i \geq 0, i \in \{1,\cdots,K\}\right\}$$

using the SOS approach with Positivstellensatz multipliers Stengle (1974), we search for a polynomial $V(x)$ and SOS polynomials $s_i(x)$ and $t_i(x)$ such that for any $\epsilon > 0$

$$\begin{aligned} V(x) - \epsilon x^T x - \sum_{i=1}^{K} s_i(x)(w_i^T x + u_i) \quad &\text{is SOS} \\ -\langle \nabla V(x), f(x)\rangle - \epsilon x^T x - \sum_{i=1}^{K} t_i(x)(w_i^T x + u_i) \quad &\text{is SOS.} \end{aligned}$$

Suppose we choose the degree of the $s_i(x)$ to be $d_V - 2$ and the degree of the $t_i(x)$ to be $d_V + d_f - 2$. Then, it can be shown that the total number of decision variables in the SDP associated with the SOS approach is

$$N_{vars}^{S} = \frac{N_1(N_1+1)}{2} + K\frac{N_2(N_2+1)}{2} + K\frac{N_3(N_3+1)}{2}, \tag{6.18}$$

where N_1 is the number of monomials in a polynomial of degree $d_V/2$, N_2 is the number of monomials in a polynomial of degree $(d_V-2)/2$ and N_3 is the number of monomials in a polynomial of degree $(d_V + d_f - 2)/2$ calculated as

$$N_1 = \sum_{d=1}^{d_V/2} \frac{(d+n-1)!}{(d)!(n-1)!}, \quad N_2 = \sum_{d=0}^{(d_V-2)/2} \frac{(d+n-1)!}{(d)!(n-1)!} \quad \text{and} \quad N_3 = \sum_{d=0}^{(d_V+d_f-2)/2} \frac{(d+n-1)!}{(d)!(n-1)!}.$$

The first terms in (6.18) is the number of scalar decision variables associated with the polynomial $V(x)$. The second and third terms are the number of scalar variables in the polynomials s_i and t_i, respectively. The number of constraints in the SDP is

$$N^S_{cons} = N_1 + K\, N_2 + K\, N_3 + N_4, \tag{6.19}$$

where

$$N_4 = \sum_{d=0}^{(d_V+d_f)/2} \frac{(d+n-1)!}{(d)!(n-1)!}.$$

The first term in (6.19) is the number of constraints associated with positivity of $V(x)$, the second and third terms are the number of constraints associated with positivity of the polynomials s_i and t_i, respectively. The fourth term is the number of constraints associated with negativity of the Lie derivative. By substituting $K = 2n$ (For the case of a hypercube), for large n we have $N^S_{vars} \sim N_3^2 \sim n^{d_V+d_f-1}$ and

$$N^S_{cons} \sim K N_3 + N_4 \sim n\, N_3 + N_4 \sim n^{0.5(d_V+d_f)}.$$

Finally, using an interior-point algorithm with complexity $O(N^3_{cons} + N^3_{var} N_{cons} + N^2_{var} N^2_{cons})$ to solve the SDP associated the SOS algorithm requires $\sim n^{3.5(d_V+d_f)-3}$ operations. As an additional comparison, we also consider the SOS algorithm for global stability analysis, which does not use Positivstellensatz multipliers. For a large number of states, we have $N^S_{vars} \sim n^{0.5 d_V}$ and $N^S_{cons} \sim n^{0.5(d_V+d_f)}$. In this case, the complexity of the SDP is

$$\sim n^{1.5(d_V+d_f)} + n^{2d_V+d_f}.$$

6.5.4 Comparison of the Complexities

We draw the following conclusions from our complexity analysis.

1. For large number of states, the complexity of the LP defined in (6.13) and the SDP associated with SOS are both **polynomial** in the number of states, whereas the complexity of the SDP associated with Polya's algorithm grows **exponentially** in the number of states. For a large number of states and large degree of the Lyapunov polynomial, the LP has the least computational complexity.

2. The complexity of the LP defined in (6.13) scales linearly with the number of sub-polytopes L.

3. In Figure 6.3, we show the number of decision variables and constraints for the LP and SDPs using different degrees of the Lyapunov function and different degrees of the vector field. The figure shows that in general, the SDP associated with Polya's algorithm has the least number of variables and the greatest number of constraints, whereas the SDP associated with SOS has the greatest number of variables and the least number of constraints.

6.6 Numerical results

In this section, we first use our algorithm to construct a Lyapunov function for a nonlinear system. we then assess the accuracy of our algorithm in estimating the region of attraction of the equilibrium point using different types of convex polytopes.

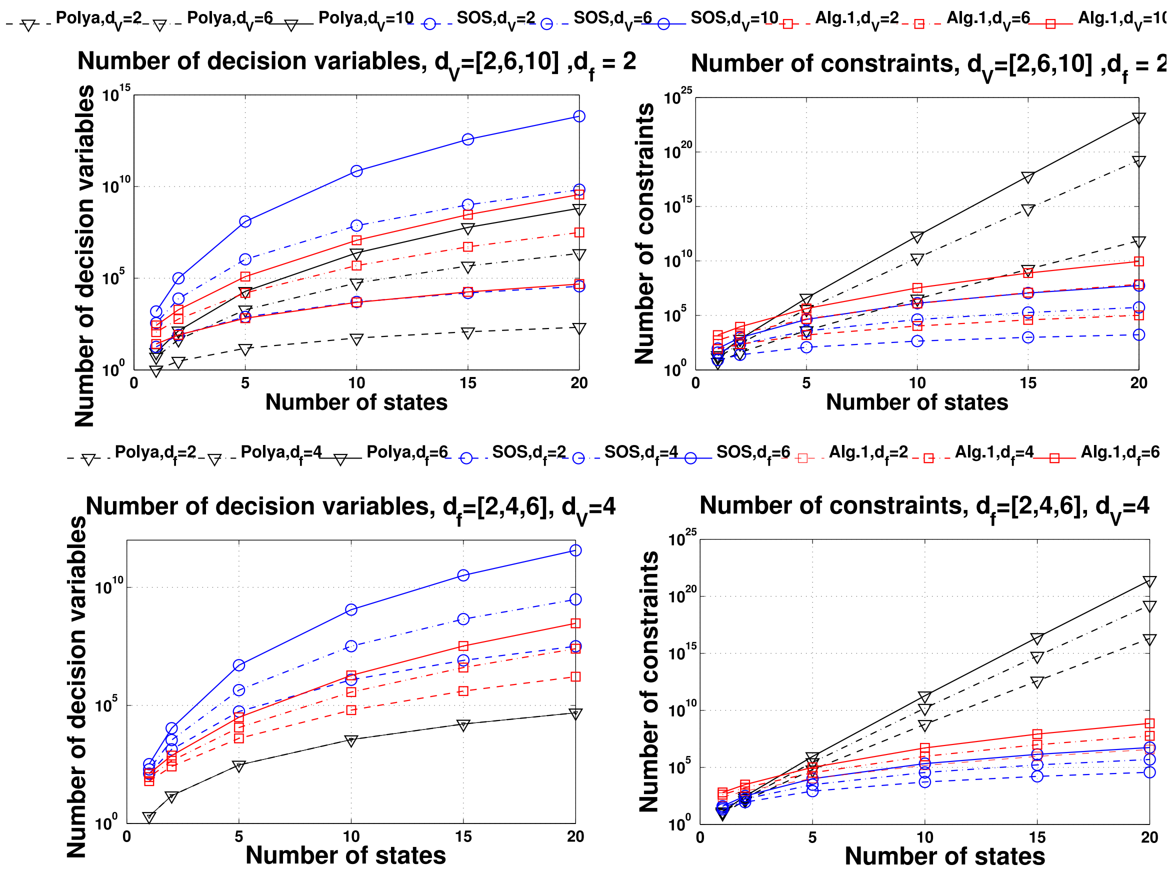


Figure 6.3: Number of decision variables and constraints of the optimization problems associated with Algorithm 1, Polya's algorithm and SOS algorithm for different degrees of the Lyapunov function and the vector field $f(x)$

Numerical Example 1:

Consider the following nonlinear system (G. Chesi and Vicino (2005)).

$$\dot{x}_1 = x_2,$$

$$\dot{x}_2 = -2x_1 - x_2 + x_1 x_2^2 - x_1^5 + x_1 x_2^4 + x_2^5.$$

Using the polytope

$$\Gamma = \{x_1, x_2 \in \mathbb{R}^2 : 1.428x_1 + x_2 - 0.625 \geq 0, -1.428x_1 + x_2 + 0.625 \geq 0,$$
$$1.428x_1 + x_2 + 0.625 \geq 0, -1.428x_1 + x_2 - 0.625 \geq 0\}, \quad (6.20)$$

and $D-$decomposition

$$D_1 := \{x_1, x_2 \in \mathbb{R}^2 : -x_1 \geq 0, x_2 \geq 0, -1.428x_1 + x_2 - 0.625 \geq 0\}$$

$$D_2 := \{x_1, x_2 \in \mathbb{R}^2 : x_1 \geq 0, x_2 \geq 0, 1.428x_1 + x_2 + 0.625 \geq 0\}$$

$$D_3 := \{x_1, x_2 \in \mathbb{R}^2 : x_1 \geq 0, -x_2 \geq 0, -1.428x_1 + x_2 + 0.625 \geq 0\}$$

$$D_4 := \{x_1, x_2 \in \mathbb{R}^2 : -x_1 \geq 0, -x_2 \geq 0, 1.428x_1 + x_2 + 0.625 \geq 0\},$$

we set-up the LP in (6.13) with $d = 4$. The solution to the LP certified asymptotic stability of the origin and yielded the following piecewise polynomial Lyapunov function. Figure 6.4 shows the largest level set of $V(x)$ inscribed in the polytope Γ.

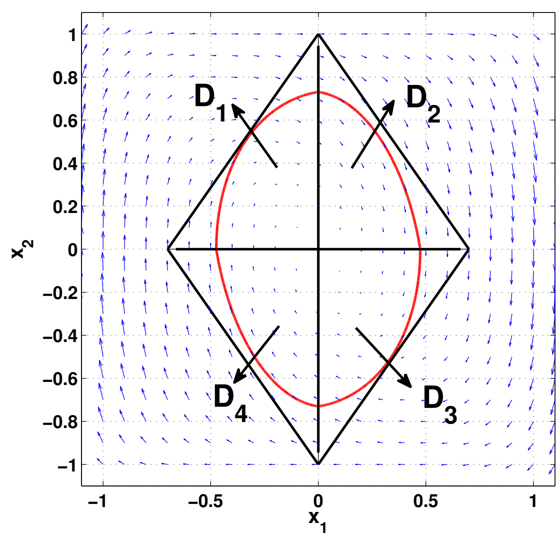


Figure 6.4: The largest level-set of Lyapunov function (6.21) inscribed in Polytope (6.20)

$$V(x) = \begin{cases} 0.543x_1^2 + 0.233x_2^2 + 0.018x_2^3 - 0.074x_1x_2^2 - 0.31x_1^3 & \\ +0.004x_2^4 - 0.013x_1x_2^3 + 0.015x_1^2x_2^2 + 0.315x_1^4 & \text{if } x \in D_1 \\ 0.543x_1^2 + 0.329x_1x_2 + 0.233x_2^2 + 0.018x_2^3 + 0.031x_1x_2^2 & \\ +0.086x_1^2x_2 + 0.3x_1^3 + 0.004x_2^4 + 0.009x_1x_2^3 + 0.015x_1^2x_2^2 & \\ +0.008x_1^3x_2 + 0.315x_1^4 & \text{if } x \in D_2 \\ 0.0543x_1^2 + 0.0233x_2^2 - 0.0018x_2^3 + 0.0074x_1x_2^2 + 0.03x_1^3 & \\ +0.004x_2^4 - 0.013x_1x_2^3 + 0.015x_1^2x_2^2 + 0.315x_1^4 & \text{if } x \in D_3 \\ 0.543x_1^2 + 0.329x_1x_2 + 0.233x_2^2 - 0.018x_2^3 - 0.031x_1x_2^2 & \\ -0.086x_1^2x_2 - 0.3x_1^3 + 0.004x_2^4 + 0.009x_1x_2^3 + 0.015x_1^2x_2^2 & \\ +0.008x_1^3x_2 + 0.315x_1^4 & \text{if } x \in D_4 \end{cases} \tag{6.21}$$

$$\tag{6.22}$$

Numerical Example 2:

In this example, we test the accuracy of our algorithm in approximating the region of attraction of a locally-stable nonlinear system known as the reverse-time Van Der Pol oscillator. The system is defined as

$$\dot{x}_1 = -x_2, \ \dot{x}_2 = x_1 + x_2(x_1^2 - 1). \tag{6.23}$$

We considered the following convex polytopes:

1. Parallelogram Γ_{P_s}, $P_s := \{sp_i\}_{i=1,\cdots,4}$, where

$$p_1 = [-1.31, 0.18], \quad p_2 = [0.56, 1.92], \quad p_3 = [-0.56, -1.92], \quad p_4 = [1.31, -0.18]$$

2. Square Γ_{Q_s}, $Q_s := \{sq_i\}_{i=1,\cdots,4}$, where

$$q_1 = [-1, 1], \quad q_2 = [1, 1], \quad q_3 = [1, -1], \quad q_4 = [-1, -1]$$

3. Diamond Γ_{R_s}, $R_s := \{sr_i\}_{i=1,\cdots,4}$, where

$$r_1 = [-1.41, 0], \quad r_2 = [0, 1.41], \quad r_3 = [1.41, 0], \quad r_4 = [0, -1.41]$$

where $s \in \mathbb{R}_+$ is a scaling factor. We decompose the parallelogram and the diamond into 4 triangles and decompose the square into 4 squares. We solved the following optimization problem for Lyapunov functions of degree $d = 2, 4, 6, 8$:

$$\begin{aligned} &\max_{s\in\mathbb{R}^+} \quad s \\ &\text{subject to} \quad \max \gamma \text{ in LP (6.13) is positive, where} \\ &\Gamma = \Gamma_{P_s} := \{x \in \mathbb{R}^2 : x = \sum_{i=1}^{4} \mu_i s p_i : \mu_i \geq 0 \text{ and } \sum_{i=1}^{K} \mu_i = 1\}. \end{aligned}$$

To solve this problem, we use a bisection search on s in an outer-loop and an LP solver in the inner loop. Figure 6.5 illustrates the largest Γ_{P_s}, i.e.

$$\Gamma_{P_{s^*}} := \{x \in \mathbb{R}^n : x = \sum_{i=1}^{4} \mu_i s^* p_i : \mu_i \geq 0 \text{ and } \sum_{i=1}^{4} \mu_i = 1\}$$

and the largest level-set of $V_i(x)$ inscribed in $\Gamma_{P_{s^*}}$, for different degrees of $V_i(x)$. Similarly, we solved the same optimization problem replacing Γ_{P_s} with the square Γ_{Q_s} and diamond Γ_{R_s}. In all cases, increasing d resulted in a larger maximum inscribed sub-level set of $V(x)$ (see Figure 6.6). We obtained the best results using the parallelogram Γ_{P_s} which achieved the scaling factor $s^* = 1.639$. The maximum scaling factor for Γ_{Q_s} was $s^* = 1.800$ and the maximum scaling factor for Γ_{R_s} was $s^* = 1.666$.

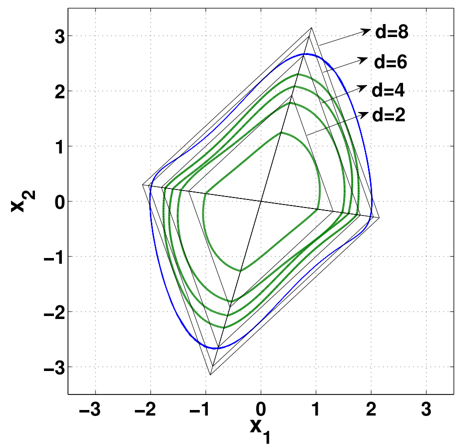
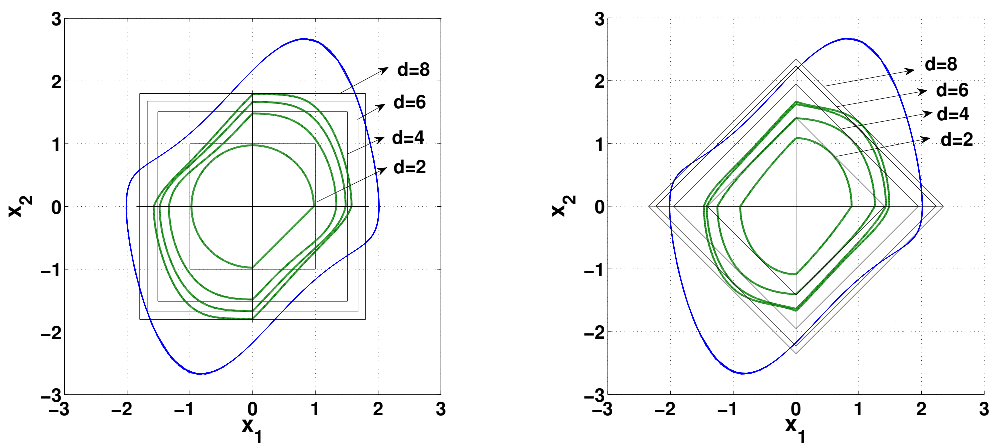


Figure 6.5: Largest level sets of Lyapunov functions of different degrees and their associated parallelograms



(a) Square polytopes (b) Diamond polytopes

Figure 6.6: Largest level sets of Lyapunov functions of different degrees and their associated polytopes

Chapter 7

OPTIMIZATION OF SMART GRID OPERATION: OPTIMAL UTILITY PRICING AND DEMAND RESPONSE

7.1 Background and motivation

Reliable and efficient production and transmission of electricity are essential to the progress of modern industrial societies. Engineers have strived for years to operate power generating systems in a way to achieve the following objectives: 1) Reliability: maintaining an uninterrupted balance between the generated power and demand; 2) Minimizing the cost of generation and transmission of electricity; 3) Reducing the adverse effects of the system on the environment by increasingly the use of renewable sources such as solar energy. Unfortunately, the first two objectives are in conflict: increasing reliability (often by increasing the maximum capacity of generation) results in higher costs. Moreover, the dependence of reliability of power networks and costs on integration of renewables is not yet well-understood.

One concern of electric utilities is that rapid increase in distributed solar generation may change customers' consumption pattern in ways that current generating units cannot accommodate for these changes. One example of such a change is shown in Figure 7.1 (Arizona Public Service (2014)). In this figure, we have compared the daily net demand profile of Arizona's customers in 2014 with its projection in 2029. Because of the misalignment between the solar generation peak (at noon) and the demand peak (at 6 PM), as the solar penetration increases, the resulting demand profile will reshape to a *double-peak* curve (see Figure 7.1). To respond to such variability in the demand profile, utilities will be required to re-structure their generating ca-

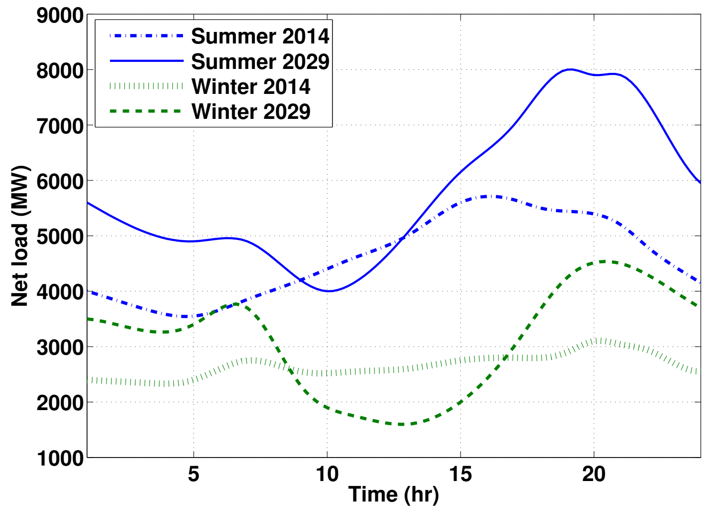


Figure 7.1: Effect of solar power on demand: Net loads for typical summer and winter days in Arizona in 2014 and for 2029 (projected), from Arizona Public Service (2014)

pacity by installing generating units which possess a shorter start-up time and higher generation ramp rates. Moreover, as solar generation by users increases, the total energy provided by the utility will decrease - implying a reduction in revenue for utility companies which charge users based on their total energy consumption. This type of change in the demand can indeed already be seen in a report by the US Energy Information Administration (EIA) as a significant increase in the ratio of the annual demand peak to annual average demand (see Figuew 7.2). Because utilities must pay to build and maintain generating capacity as determined by peak demand, the increasing use of solar will thus result in a decrease in revenue, yet no decrease in this form of cost. Ultimately, utilities might have a significant fraction of solar users with negative energy consumption (kWh) during the day and positive consumption during the evening and morning. Due to net metering, such users might pay nothing for electricity while contributing substantially to the costs of building and maintaining generating capacity.

Recently, there has been extensive research on how to exploit *smart grid* fea-

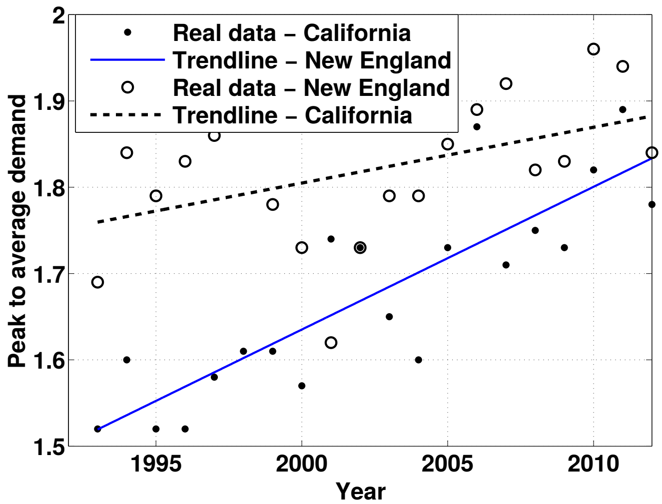


Figure 7.2: Peak to average demand of electricity and its trend-line in California and New England from 1993 to 2012, data adopted from Shear (2014)

tures such as smart metering, energy storage, thermostat programming and variable/dynamic pricing in order to reduce peak demands and cost of generation, enhance monitoring and security of networks, and prevent unintended events such as cascade failures and blackouts. Smart metering enables two-way communications between consumers and utilities. It provides utilities with real-time data of consumption - hence enables them to directly control the load and/or apply prices as a function of consumption. Naturally, utilities have been studying this problem for some time and with the widespread adoption of smart-metering (95% in Arizona), have begun to implement various pricing strategies at scale. Examples of this include on-peak, off-peak and super-peak pricing - rate plans wherein the energy price ($/kWh) depends on the time of day. By charging more during peak hours, utilities encourage conservation or deferred consumption during hours of peak demand. Quite recently, some utilities have introduced demand charges for residential customers (SRP (2015),Rumolo (2013)). These charges are not based on energy consumption, but rather the maximum *rate of consumption* ($/kW) over a billing period. While such charges more accurately reflect the cost of generation for the utilities, in practice the effects of such charges on consumption are not well-understood - meaning that the magnitude of the demand charge must be set in an ad-hoc manner (typically proportional to marginal cost of adding generating capacity).

An alternative approach to reducing peaks in demand is to use energy storage. In this scenario, batteries, pumping and retained heat are used during periods of low demand to create reservoirs of energy which can then be tapped during periods of high demand - thus reducing the need to increase maximum generating capacity. Indeed, the optimal usage of energy storage in a smart-grid environment with dynamic pricing has been recently studied in, for example, Li *et al.* (2011). See Ma *et al.* (2014) for optimal distributed load scheduling in the presence of network capacity constraints. However, to date the high marginal costs of storage infrastructure relative to incentives/marginal cost of additional generating capacity have limited the widespread use of energy storage by consumers/utilities (EPRI-DOE (2003)). As a cost-free alternative to direct energy storage, it has been demonstrated experimentally by Braun *et al.* (2002), Braun (2003), and in-silico by Braun *et al.* (2001) and Keeney and Braun (1997) that the interior structure of buildings and appliances can be exploited as a *passive* thermal energy storage system to reduce the peak-load of HVAC. A typical strategy - known as *pre-cooling* - is to artificially cool the interior thermal mass (e.g., walls and floor) during periods of low demand. Then, during periods of high demand, heat absorption by these cool interior structures supplements or replaces electricity which would otherwise be consumed by the HVAC. Quantitative assessments of the effect of pre-cooling on demand peak and electricity bills can be found in Braun and Lee (2006) and Sun *et al.* (2013). Furthermore, there is an extensive literature on thermostat programming for HVAC systems for on-peak/off-peak pricing (Lu *et al.* (2005); Arguello-Serrano and Velez-Reyes (1999)) as well as *real-time* pricing (Oldewurtel and Morari (2010); Henze *et al.* (2004); Chen (2001)) using Model Predictive

Control (MPC). Kintner-Meyer and Emery (1995) consider optimal thermostat programming with passive thermal energy storage and on-peak/off-peak rates. Braun and Lee (2006) use the concept of *deep* and *shallow* mass to create a simplified analogue circuit model of the thermal dynamics of the structure. By using this model and certain assumptions on the gains of the circuit elements, Braun and Lee (2006) derive an analytical optimal temperature set-point for the demand limiting period which minimizes the demand peak. This scenario would be equivalent to minimizing the demand charge while ignoring on-peak or off-peak rates. Finally, Henze *et al.* (2004) use the heat equation to model the thermal energy storage in the walls and apply MPC to minimize monthly electricity bill in the presence of on-peak and off-peak charges.

7.1.1 Our contributions

In this chapter, we design a computational framework to achieve the three objectives of a modern power network: reliability, cost minimization and integration of renewables to promote sustainability. This framework relies on smart metering, thermal-mass energy storage, distributed solar generation and on-peak, off-peak and demand pricing. This framework consists of two nested optimization problems: 1) Optimal thermostat programming (*user-level problem*); 2) Optimal utility pricing (*utility-level problem*). In the first problem, we consider optimal HVAC usage for a consumer with fixed on-peak, off-peak and demand charges and model passive thermal energy storage using the heat equation. We address both solar and non-solar consumers. For a given range of acceptable temperatures and using typical data for exterior temperature, we pose the optimal thermostat programming problem as a constrained optimization problem and present a Dynamic Programming (DP) algorithm which is guaranteed to converge to the solution. This yields the temperature set-points which minimize the monthly electricity bill for the consumer. For the benefit of the consumers who do not have access to continuously adjustable thermostats, we also develop thermostat programming solutions which include only four programming periods, where each period has a constant interior temperature.

After solving the thermostat programming problem, we use this solution as a model of user behaviour in order to quantify the consumer response to changes in on-peak rates, off-peak rates, and demand charges. Then in the second optimization problem, we apply a descent algorithm to this model in order to determine the prices which minimize the cost-of-generation for the utility. Through several case studies, we show that the optimal prices are NOT necessarily proportional to the marginal costs of generation - meaning that current pricing strategies may be inefficient. Furthermore, we show that in a network of solar and non-solar customers who use our optimal thermostat, the influence of solar generated power on the electricity bills of non-solar customers is NOT significant. Finally, we conclude that although the policy of calculating the demand charge based on the peak consumption over a full-day (rather than the on-peak hours) can substantially reduce the demand peak, it may not reduce optimal cost of production. Our study differs from existing literature (in particular Braun and Lee (2006), Braun (1990), Henze *et al.* (2004) and Kintner-Meyer and Emery (1995)) in that it: 1) Considers demand charges (demand charges are far more effective at reducing demand peaks than dynamic pricing) 2) Uses a PDE model for thermal storage (yields a more accurate model of thermal storage) 3) Uses a regulated model for the utility (although unregulated utility models are popular, the fact is that most US utilities remain regulated) 4) Considers the effect of solar generation on the electricity prices and cost of production.

7.2 Problem statement: user-level and utility level problems

In this section, we first define a model of the thermodynamics which govern heating and cooling of the interior structures of a building. We then use this model to pose the user-level (optimal thermostat programming) problem in Sections 7.2.3 and 7.2.4 as minimization of a monthly electricity bill (with on/peak, off-peak and demand charges) subject to constraints on the interior temperature of the building. Finally, we use this map of on-peak, off-peak and demand prices to electricity consumption to define the utility-level problem in Section 7.2.5 as minimizing the cost of generation, transmission and distribution of electricity.

7.2.1 A model for the building thermodynamics

In 1822, J. Fourier proposed a PDE to model the dynamics of temperature and energy in a solid mass. Now known as the classical one-dimensional unsteady heat conduction equation, this PDE can be applied to an interior wall as

$$\frac{\partial T(t,x)}{\partial t} = \alpha \frac{\partial^2 T(t,x)}{\partial x^2}, \tag{7.1}$$

where $T : \mathbb{R}^+ \times [0, L_{in}] \to \mathbb{R}$ represents the temperature distribution in the interior walls/floor with nominal width L_{in}, and where $\alpha = \frac{k_{in}}{\rho C_p}$ is the coefficient of thermal diffusivity. Here k_{in} is the coefficient of thermal conductivity, ρ is the density and C_p is the specific heat capacity. The wall is coupled to the interior air temperature using Dirichlet boundary conditions, i.e., $T(t,0) = T(t, L_{in}) = u(t)$ for all $t \in \mathbb{R}^+$, where $u(t)$ represents the interior temperature which we assume can be controlled instantaneously by the thermostat. In the Fourier model, the heat/energy flux through the surface of the interior walls is modelled as

$$q_{in}(T(t,x)) := 2C_{in} \frac{\partial T}{\partial x}(t,0), \tag{7.2}$$

where $C_{in} = k_{in} A_{in}$ is the thermal capacitance of the interior walls and A_{in} is the nominal area of the interior walls. We assume that all energy storage occurs in the interior walls and surfaces and that energy transport through exterior walls can be modelled using a steady-state version of the heat equation. This implies that the heat flux q_{loss} through the exterior walls is the linear sink

$$q_{loss}(t, u(t)) := \frac{T_e(t) - u(t)}{R_e}, \tag{7.3}$$

where $T_e(t)$ is the outside temperature and $R_e = L_e/(k_e A_e)$ is the thermal resistance of the exterior walls, where L_e is the nominal width of exterior walls, k_e is the coefficient of thermal conductivity and A_e is the nominal area of the exterior walls. By conservation of energy, the power required from the HVAC to maintain the interior air temperature is

$$q(t, u(t), T(t, x)) = q_{loss}(u(t), T_e(t)) + q_{in}(T(x, t)). \tag{7.4}$$

See Figure 7.3 for an illustration of the model.

Eqn. (7.1) is a PDE. For optimization purposes, we discretize (7.1) in space, using $T(t) \in \mathbb{R}^M$ to replace $T(t, x) \in \mathbb{R}$, where $T_i(t)$ denotes $T(t, i\,\Delta x)$, where $\Delta x := \frac{L_{in}}{M+1}$. Then

$$\dot{T}(t) = A\,T(t) + B\,u(t), \tag{7.5}$$

$$\text{where } A = \frac{\alpha}{\Delta x^2} \begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix},\ B = \frac{\alpha}{\Delta x^2} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^M.$$

We then discretize in time, using $\dot{T}(t) \approx (T(t + \Delta t) - T(t))/\Delta t$ to rewrite Equa-

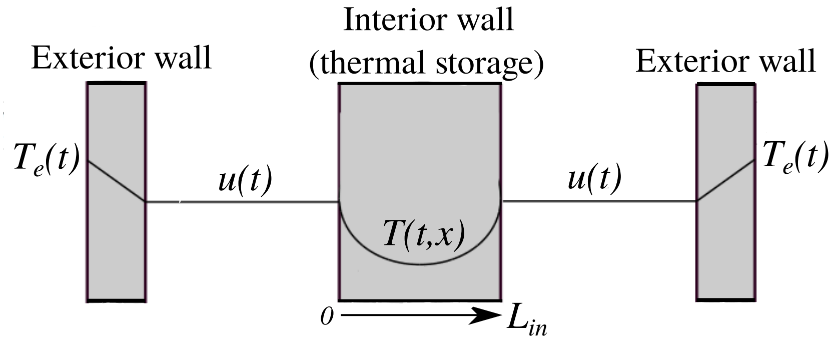


Figure 7.3: A schematic view of our thermal mass model

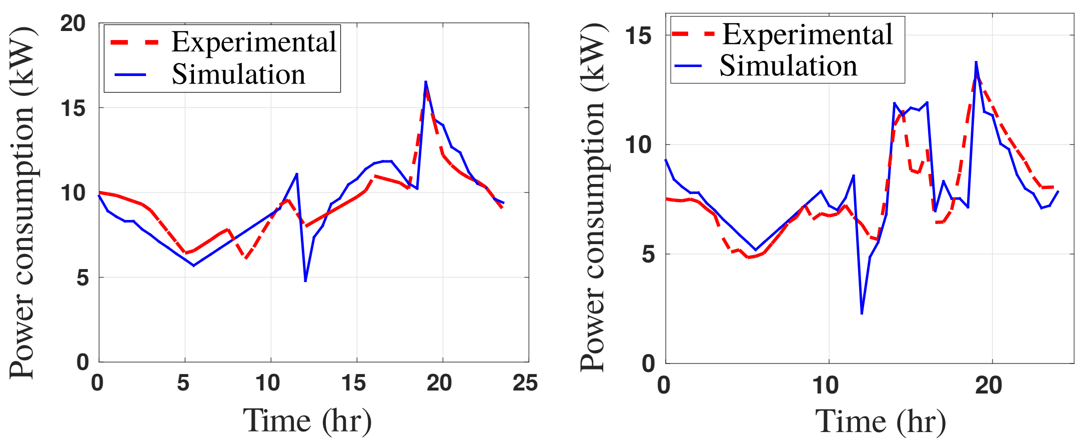
tion (7.5) as a difference equation.

$$T^{k+1} = \begin{bmatrix} T_1^{k+1} \\ \vdots \\ T_M^{k+1} \end{bmatrix} = f(T^k, u_k) = \begin{bmatrix} f_1(T^k, u_k) \\ \vdots \\ f_M(T^k, u_k) \end{bmatrix} = (I + A\,\Delta\, t)T^k + B\,\Delta t\, u_k \qquad (7.6)$$

for $k = 0, \cdots, N_f - 1$, where $T^k = T(k\,\Delta t)$ and $u_k = u(k\,\Delta t)$.

7.2.2 Calibrating the thermodynamics model

To find empirical values for the parameters α, C_{in}, R_e and L_{in} in the thermodynamic model in Section 7.2.1, we collected data from a 4600 sq ft residential building in Scottsdale, Arizona. The building was equipped with a 5 ton two-stage and three 2.5 ton single-stage RHEEM/RUUD heat pumps, 4-set point thermostats, and 5-min data metering for energy consumption and interior and exterior temperature. In this experiment, we applied two different thermostat programming sequences for two non-consecutive summer days. On the first day, we applied a pre-cooling strategy which lowers the interior temperature to 23.9°C during the off-peak hours and allows the temperature to increase to 27.8°C during the on-peak hours, i.e., 12:00 PM to 7:00 PM. In the second day, we applied the same pre-cooling strategy except that the temperature is again lowered to 23.9°C between 2:00 PM and 4:00 PM. We then used Matlab's least squares optimization algorithm to optimize the parameters such



(a) Power consumption corresponding to a pre-cooling strategy for the interior temperature setting

(b) Power consumption corresp. to a pre-cooling strategy with additional cooling from 14:00-16:00

Figure 7.4: Simulated and measured power consumptions

that the root-mean-squared error between the measured power consumption and the simulated power consumption during the entire two days is minimized. The result was the following values for the parameters: $L_{in} = 0.4(m)$, $\alpha = 8.3 \times 10^{-7}(m^2/s)$, $R_e = 0.0015(K/W)$ and $C_{in} = 45(Wm/K)$. In Figure 7.4, we have compared the resulting simulated and measured power consumption for the entire two days.

7.2.3 User-level problem I: optimal thermostat programming

In this section, we define the problem of optimal thermostat programming. We first divide each day into three periods: off-peak hours from 12 AM to t_{on} with electricity price $p_{\text{off}}\,(\$/kWh)$; on-peak hours beginning at t_{on} and ending at $t_{\text{off}} > t_{\text{on}}$ with electricity price $p_{\text{on}}\,(\$/kWh)$; and off-peak hours from t_{off} to 12 AM with electricity price $p_{\text{off}}\,(\$/kWh)$. In addition to the on-peak and off-peak charges, we consider a monthly charge which is proportional to the maximum rate of consumption during the peak hours. The proportionality constant is called the *demand price* $p_d\,(\$/kW)$. Given $p := [p_{\text{on}}, p_{\text{off}}, p_d]$, the total cost of consumption (daily electricity bill) is divided as

$$J_t(\mathbf{u}, T_1, p) = J_e(\mathbf{u}, T_1, p_{\text{on}}, p_{\text{off}}) + J_d(\mathbf{u}, T_1, p_d), \tag{7.7}$$

where J_e is the energy cost, J_d is the demand cost and

$$\mathbf{u} := [u_0, \cdots, u_{N_f-1}] \in \mathbb{R}^{N_f}$$

is the vector temperature settings. The energy cost is

$$J_e(\mathbf{u}, T_1, p_{\text{on}}, p_{\text{off}}) = \Big(p_{\text{off}} \sum_{k \in S_{\text{off}}} g(k, u_k, T_1^k) + p_{\text{on}} \sum_{k \in S_{\text{on}}} g(k, u_k, T_1^k) \Big) \Delta t, \tag{7.8}$$

where $k \in S_{\text{on}}$ if $k\,\Delta t \in [t_{\text{on}}, t_{\text{off}}]$ and $k \in S_{\text{off}}$ otherwise. That is, S_{on} and S_{off} correspond to the set of on-peak and off-peak sampling times, respectively. The function g is a discretized version of q (Eqn. (7.4)):

$$g(k, u_k, T_1^k) := \frac{T_e^k - u_k}{R_e} + 2\, C_{in} \frac{T_1^k - u_k}{\Delta x}. \tag{7.9}$$

i.e., g the power consumed by the HVAC at time step k, where T_e^k denotes the external temperature at time-step k. If demand charges are calculated monthly, the demand cost, J_d, for a single day can be considered as

$$J_d(\mathbf{u}, T_1, p_d) := \frac{p_d}{30} \max_{k \in S_{\text{on}}} g(k, u_k, T_1^k). \tag{7.10}$$

We now define the optimal thermostat programming (or user-level) problem as minimization of the total cost of consumption, J_t, as defined in (7.7), subject to the building thermodynamics in (7.6) and interior temperature constraints:

$$\begin{aligned} J^\star(p) = & \min_{u_k, \gamma \in \mathbb{R}, T^k \in \mathbb{R}^M} J_e(\mathbf{u}, T_1, p_{\text{on}}, p_{\text{off}}) + \frac{p_d}{30} \gamma \\ & \text{subject to } \; g(k, u_k, T_1^k) \leq \gamma \qquad \text{for } k \in S_{\text{on}} \\ & \qquad T^{k+1} = f(T^k, u_k) \qquad \text{for } k \in S_{\text{on}} \cup S_{\text{off}} \\ & \qquad T_{\min} \leq u_k \leq T_{\max} \qquad \text{for } k \in S_{\text{on}} \cup S_{\text{off}} \\ & \qquad T^0 = [T_{\text{init}}(\Delta x), \cdots, T_{\text{init}}(M\,\Delta x)]^T, \end{aligned} \tag{7.11}$$

where $T_{\min}, T_{\max}$ are the acceptable bounds on interior temperature. Note that this optimization problem depends implicitly on the external temperature through the time-varying function g.

7.2.4 User-level problem II: 4-setpoint thermostat program

Most of the commercially available programmable thermostats only include four programming periods per-day, each period possessing a constant temperature. In this section, we account for this contraint. First, we partition the day into programming periods: $P_i := [t_{i-1}, t_i]$, $i = 1, \cdots, 4$ such that

$$\bigcup_{i=1}^{4} P_i = [0, 24], \quad t_{i-1} \le t_i, \quad t_0 = 0 \text{ and } t_4 = 24.$$

We call $t_0, \cdots, t_4$ *switching* times. Similar to the previous model, $u_i \in [T_{\min}, T_{\max}]$ denotes the temperature setting corresponding to the programming period P_i.

To simplify the mathematical formulation of our problem, we introduce some additional notation. Define the set S_i by $k \in S_i$ if $k\Delta t \in P_i$. Denote $L_i := \max_{k \in S_i} k$. For clarity, we have depicted L_i in Figure 7.5. Moreover, for each P_i, we define $\overline{\Delta t}_i$ as the period between the last time-step of P_i and the end of P_i, i.e., $\overline{\Delta t}_i := t_i - L_i \, \Delta t$. See Figure 7.5 for an illustration of $\overline{\Delta t}_i$. In this framework, the daily consumption charge is

$$I_t(\mathbf{u}, T_1, p) = I_e(\mathbf{u}, T_1, p_{\text{on}}, p_{\text{off}}) + I_d(\mathbf{u}, T_1, p_d)$$

where I_e is the energy cost

$$I_e(\mathbf{u}, T_1, p_{\text{on}}, p_{\text{off}}) = \sum_{i=1}^{4} \left(\sum_{k \in S_i} \big(r(k) g(k, u_i, T_1^k) \Delta t \big) + r(L_i) g(k, u_i, T_1^{L_i}) \overline{\Delta t}_i \right), \quad (7.12)$$

and I_d is the demand cost $I_d(\mathbf{u}, T_1, p_d) = \max_{k \in S_{\text{on}}} g(k, u, T_1^k)$, where $L_0 = 0$ and r is defined as

$$r(k) := \begin{cases} p_{\text{on}} & t_{\text{off}} \le k \, \Delta t < t_{\text{on}} \\ p_{\text{off}} & \text{otherwise.} \end{cases}$$

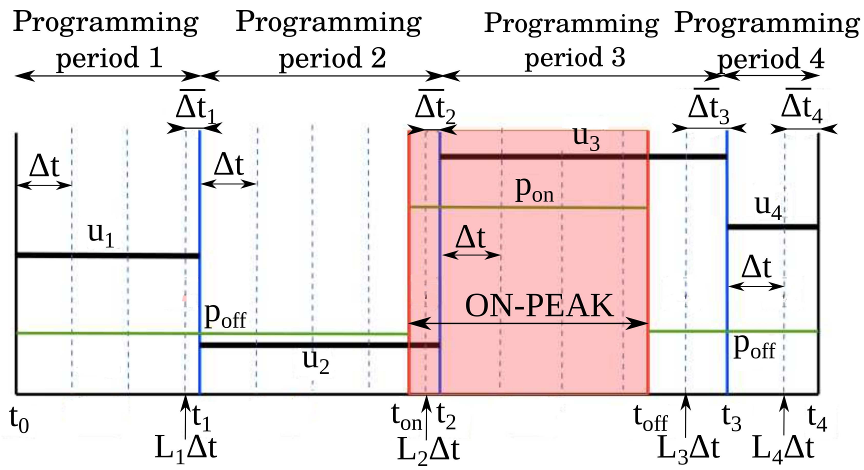


Figure 7.5: An illustration for the programming periods of the 4-Setpoint thermostat problem, switching times t_i, pricing function r, L_i and $\overline{\Delta t}_i$.

Assuming that the demand cost for a single day is $\frac{p_d}{30} \max_{k \in S_{\text{on}}} g(k, u, T_1^k)$, we define the 4-setpoint thermostat programming problem as

$$\begin{aligned}
&\min_{\substack{u_1, \cdots, u_4 \in \mathbb{R} \\ t_1, t_2, t_3, \gamma \in \mathbb{R}, T^k \in \mathbb{R}^M}} \quad I_e(\mathbf{u}, T_1, p_{\text{on}}, p_{\text{off}}) + \frac{p_d}{30}\gamma \qquad \text{subject to} \\
&g(k, u_i, T_1^k) \leq \gamma \qquad \text{for } k \in S_{\text{on}},\ i \in \{1, 2, 3, 4\} \\
&T^{k+1} = f(T^k, u_i) \qquad \text{for } k \in S_i \text{ and } i \in \{1, 2, 3, 4\} \\
&T_{\min} \leq u_i \leq T_{\max} \qquad \text{for } i \in \{1, 2, 3, 4\} \\
&0 \leq t_{i-1} \leq t_i \leq 24 \qquad \text{for } i \in \{1, 2, 3, 4\} \\
&T^0 = [T_{\text{init}}(\Delta x), \cdots, T_{\text{init}}(M\,\Delta x)]^T,
\end{aligned} \tag{7.13}$$

where $t_0 = 0$ and $t_4 = 24$.

7.2.5 *Utility-level optimization problem*

Regulated utilities must meet expected load while maintaining a balance between revenue and costs. Therefore, we define the utility-level optimization problem as minimization of the total cost of generation, transmission and distribution of electricity such that generation is equal to consumption, and the total cost is a fixed percentage of the revenue of the utility company. Note that in this dissertation, we focus on vertically integrated utility companies - meaning that the company provides all aspects of electric services including generation, transmission, distribution, metering and billing services as a single firm. Let $s(t)$ be the amount of electricity produced as a function of time and let $\mathbf{s} := [s_0, \cdots, s_{N_f-1}]$, where $s_k = s(k\,\Delta t)$. The vector $\mathbf{s}$ is determined by the electricity consumed by the users, which we model as a small number of user groups which are lumped according to different building models, temperature limits, and solar generating capacity so that aggregate user group i has N_i members. Next, we define $u_k^{\star,i}(p)$ to be the minimizing user temperature setting for user i at time k with prices p and $T_j^{i,\star,k}(p)$ to be the minimizing interior wall temperatures for aggregate user i at time k and discretization point j for prices p, where minimization is with respect to the user-level problem defined in (7.11). Then the model of electricity consumption by the rational user i at time step k for prices p is given by $g(k, u_k^{\star,i}(p), T_1^{\star,k,i}(p))$. Thus the constraint that production equals consumption at all time implies

$$s_k = \sum_i N_i g(k, u_k^{\star,i}(p), T_1^{i,\star,k}(p)) \text{ for all } k = 0, \cdots, N_f - 1. \tag{7.14}$$

Now, since utility's revenue equals the amount paid by the users, the model for revenue from rational user i becomes $J_t(u^{\star,i}(p), T_1^{i,\star}(p), p)$, where J_t is defined in (7.7). We may now define the utility-level optimization problem as minimization of the total cost subject to equality of generation and consumption and proportionality of revenue and total costs.

$$\begin{aligned}
\min_{p_{\text{on}}, p_{\text{off}}, p_d \in \mathbb{R}} \quad & C(\mathbf{s}) \\
\text{subject to} \quad & s_k = \sum_i N_i\, g(k, u_k^{\star,i}(p), T_1^{i,\star,k}(p)) \quad k = 0, \cdots, N_f - 1 \\
& C(\mathbf{s}) = \lambda \sum_i N_i\, J_t(u^{\star,i}(p), T_1^{i,\star}(p), p),
\end{aligned} \tag{7.15}$$

where $\lambda \leq 1$ is usually determined by the company's assets, accumulated depreciation and allowed rate of return. We refer to the minimizers $p^\star_{\text{on}}, p^\star_{\text{off}}, p^\star_d$ which solve Problem (7.15) as *optimal electricity prices.*

Model of total cost, $C(s)$, to utility company The algorithm defined in the following section was chosen so that only a black-box model of utility costs is required. However, for the case studies included in Section 7.4, we use two models of utility costs based on ongoing discussions and collaboration with Arizona's utility company SRP. In the first model, we consider a linear representation of both fuel and capacity costs.

$$C(\mathbf{s}) := a \sum_{k \in S_{\text{on}} \cup S_{\text{off}}} s_k \, \Delta t + b \max_{k \in S_{\text{on}}} s_k, \tag{7.16}$$

where $a\,(\$/kWh)$ is the marginal cost of producing the next kWh of energy and $b\,(\$/kW)$ is the marginal cost of installing and maintaining the next kW of capacity. Estimated values of the coefficients a and b for SRP can be found in SRP (2014) as $a = 0.0814\$/kWh$ and $b = 59.76\$/kW$. According to SRP (2014), these marginal costs include fuel, building, operation and maintenance of facilities, transmission and distribution costs. The advantage of this model is that the solution to the utility optimization problem does not depend on the number of users, but rather the fraction of users in each group.

Our second model for utility costs includes a quadratic term to represent fuel costs. The quadratic term reflects the increasing fuel costs associated with the required use of older, less-efficient generators when demand increases.

$$C(\mathbf{s}) := \tau \Big(\sum_{k \in S_{\text{on}} \cup S_{\text{off}}} s_k \, \Delta t \Big)^2 + \nu \sum_{k \in S_{\text{on}} \cup S_{\text{off}}} s_k \, \Delta t + b \max_{k \in S_{\text{on}}} s_k \tag{7.17}$$

This model was calibrated using artificially modified fuel, operation and maintenance data provided by SRP, yielding estimated coefficients τ =0.00401 \$/(MWh)^2 and

ν =4.54351 \$/(MWh).

7.3 Solving user- and utility-level problems by dynamic programming

First, we solve the optimal thermostat programming problem using a variant of dynamic programming. This yields consumption as a function of prices $p_{\text{on}}, p_{\text{off}}, p_d$. Next, we embed this implicit function in the Nelder-Mead simplex algorithm in order to find prices which minimize the production cost in the utility-level optimization problem as formulated in (7.15). We start the user-level problem by fixing the variable $\gamma \in \mathbb{R}^+$ and defining a *cost-to-go* function, V_k. At the final time $N_f\,\Delta t = 24$, we have

$$V_{N_f}(x) := \frac{p_d}{30}\gamma. \tag{7.18}$$

Here for simplicity, we use $x = T \in \mathbb{R}^M$ to represent the discretized temperature distribution in the wall. We define the dilated vector of prices by $p_j = p_{\text{off}}$ if $j \in S_{\text{off}}$ and $p_j = p_{\text{on}}$ otherwise. Then, we construct the cost-to-go function inductively as

$$V_{j-1}(x) := \min_{u \in W_{\gamma,j-1}(x)} (p_{j-1}\, g(j-1, u, x_1)\Delta t + V_j(f(x,u))) \tag{7.19}$$

for $j = 1, \cdots, N_f$, where $W_{\gamma,j}(x)$ is the set of allowable inputs (interior air temperatures) at time j and state x:

$$W_{\gamma,j}(x) := \begin{cases} \{u \in \mathbb{R} : T_{\min} \le u \le T_{\max}, g(j,u,x_1) \le \gamma\}, & j \in S_{\text{on}} \\ \{u \in \mathbb{R} : T_{\min} \le u \le T_{\max}\}, & j \in S_{\text{off}}. \end{cases}$$

Now we present the main result.

Theorem 21. *Given $\gamma \in \mathbb{R}^+$, suppose that V_i satisfies* (7.18) *and* (7.19). *Then*

$V_0(T^0) = J^*$, *where*

$$J^*(p) = \min_{u_k, T^k \in \mathbb{R}^M} J_e(\mathbf{u}, T_1, p_{on}, p_{off}) + \frac{p_d}{30}\,\gamma$$

$$\begin{aligned} \textit{subject to} \quad & g(k, u_k, T_1^k) \le \gamma && \textit{for } k \in S_{on} \\ & T^{k+1} = f(T^k, u_k) && \textit{for } k \in S_{on} \cup S_{off} \\ & T_{\min} \le u_k \le T_{\max} && \textit{for } k \in S_{on} \cup S_{off} \\ & T^0 = [T_{init}(\Delta x), \cdots, T_{init}(M\,\Delta x)]^T. \end{aligned} \tag{7.20}$$

To prove Theorem 21, we require the following definitions.

Definition 8. *Given $p_{off}, p_{on}, p_d, \gamma \in \mathbb{R}^+$, $N_f \in \mathbb{N}^+$, and $t_{off}, t_{on}, \Delta t \in \mathbb{R}^+$ such that $\frac{t_{on}}{\Delta t}, \frac{t_{off}}{\Delta t} \in \mathbb{N}$, define the cost-to-go functions*

$$Q_j : \mathbb{R}^{N_f - j} \times \mathbb{R}^{N_f - j + 1} \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R} \quad \textit{for } j = 0, \cdots, N_f \textit{ as}$$

$$Q_j(x, y, p_{on}, p_{off}) :=$$
$$\begin{cases} \left(p_{off} \sum\limits_{\substack{k \in S_{off} \\ k \notin \{0, \cdots, j-1\}}} g(k, x_k, y_k) + p_{on} \sum\limits_{k \in S_{on}} g(k, x_k, y_k) \right) \Delta t + \sum\limits_{i=1}^{3} \Gamma_i & \textit{if } 0 \le j < N_{on} \\ \Gamma_1 = \left(p_{on} \sum\limits_{\substack{k \in S_{on} \\ k \notin \{N_{on}, \cdots, j-1\}}} g(k, x_k, y_k) + p_{off} \sum\limits_{\substack{k \in S_{off} \\ k \notin \{0, \cdots, N_{on}-1\}}} g(k, x_k, y_k) \right) \Delta t + \Gamma_2 + \Gamma_3 & \textit{if } N_{on} \le j < N_{off} \\ \Gamma_2 = p_{off} \sum\limits_{k \in \{j, \cdots, N_f - 1\}} g(k, x_k, y_k) \Delta t + \Gamma_3 & \textit{if } N_{off} \le j < N_f \\ \Gamma_3 = \dfrac{p_d}{30} \gamma & \textit{if } j = N_f, \end{cases} \tag{7.21}$$

where g is defined as in (7.9), and $N_{on} := \frac{t_{on}}{\Delta t}$ and $N_{off} := \frac{t_{off}}{\Delta t}$ are the time-steps corresponding to start and end of the on-peak hours.

Note that from (7.8), it is clear that $Q_0 = J_e + \frac{p_d}{30}\gamma$.

Definition 9. *Given $\gamma, T_{\min}, T_{\max} \in \mathbb{R}$ and $N_f, M \in \mathbb{N}^+$, define the set*

$$\begin{aligned} U_j(x) := \{(u_j, \cdots, u_{N_f-1}) \in \mathbb{R}^{N_f-j} : \\ & g(k, u_k, T_1^k) \leq \gamma && \text{for all } k \in S_{on}, \\ & T^j = x \text{ and } T^{k+1} = f(T^k, u_k) && \text{for all } k \in \{j, \cdots, N_f - 1\}, \\ & T_{\min} \leq u_k \leq T_{\max} && \text{for all } k \in S_{on} \cup S_{off}\} \end{aligned} \tag{7.22}$$

for any $x \in \mathbb{R}^M$ and for every $j \in \{0, \cdots, N_f - 1\}$, where f and g are defined as in (7.6) *and* (7.9)*.*

Definition 10. *Given $N_f, M \in \mathbb{N}^+$, $j \in \{0, \cdots, N_f - 1\}$, let*

$$\overline{\mu}_j := [\mu_j, \cdots, \mu_{N_f-1}]$$

where $\mu_k : \mathbb{R}^M \to \mathbb{R}$ for $k = j, \cdots, N_f - 1$. Consider U_j as defined in (7.22) *and f as defined in* (7.6)*. If*

$$\overline{\mu}_j(w) := [\mu_j(w), \mu_{j+1}(T^{j+1}) \cdots, \mu_{N_f-1}(T^{N_f-1})] \in U_j(T^j)$$

for any $w \in \mathbb{R}^M$, where

$$T^{k+1} = f(T^k, \mu_k(T^k)), T^j = w \text{ for } k = j, \cdots, N_f - 2,$$

then we call $\overline{\mu}_j$ an admissible control law for the system

$$T^{k+1} = f(T^k, \mu_k(T^k)),\ k = j, \cdots, N_f - 1$$

for any $w \in \mathbb{R}^M$.

We now present a proof for Theorem 21.

Proof. Since the cost-to-go function $Q_0 = J_e + \dfrac{p_d}{30}\gamma$, if we show that

$$\min_{\overline{\mu}_j(T^j) \in U_j(T^j)} \; Q_j(\overline{\mu}_j(T^j), T_1, p_{\text{on}}, p_{\text{off}}) = V_j(T^j) \tag{7.23}$$

for $j = 0, \cdots, N_f$ and for any $T^j \in \mathbb{R}^M$, where

$$T_1 := [T^j, f(T^j, \mu_j(T^j)), \cdots, f(T^{N_f-1}, \mu_{N_f-1}(T^{N_f-1}))],$$

then it will follow that $J^* = V_0(T^0)$. For brevity, we denote $\overline{\mu}_j(T^j)$ by $\overline{\mu}_j$, $U_j(T^j)$ by U_j and we drop the last two arguments of Q_j. To show (7.23), we use induction as follows.

<u>*Basis step*</u>: If $j = N_f$, then from (7.18) and (7.21) we have $V_{N_f}(T^{N_f}) = \frac{p_d}{30}\gamma$.

<u>*Induction hypothesis*</u>: Suppose

$$\min_{\overline{\mu}_k \in U_k} \; Q_k(\overline{\mu}_k, T_1) = V_k(T^k)$$

for some $k \in \{0, \cdots, N_f\}$ and for any $T^k \in \mathbb{R}^M$. Then, we need to prove that

$$\min_{\overline{\mu}_{k-1} \in U_{k-1}} \; Q_{k-1}(\overline{\mu}_{k-1}, T_1) = V_{k-1}(T^{k-1}) \tag{7.24}$$

for any $T^k \in \mathbb{R}^M$. Here, we only prove (7.24) for the case which $N_{\text{off}} < k \leq N_f - 1$. The proofs for the cases $0 \leq k \leq N_{\text{on}}$ and $N_{\text{on}} < k \leq N_{\text{off}}$ follow the same exact logic.

Assume that $N_{\text{off}} < k \leq N_f - 1$. Then, from Definition 8

$$\begin{aligned}
&\min_{\overline{\mu}_{k-1} \in U_{k-1}} \; Q_{k-1}(\overline{\mu}_{k-1}, T_1) \\
&= \min_{\mu_{k-1}, \cdots, \mu_{N_f-1} \in R} \; p_{\text{off}} \left(\sum_{j=k-1}^{N_f-2} g\left(j, \mu_j, T_1^j\right) \right) \Delta t \\
&= \min_{\mu_{k-1}, \cdots, \mu_{N_f-1} \in R} p_{\text{off}} \left(g\left(k-1, \mu_{k-1}, T_1^{k-1}\right) + \sum_{j=k}^{N_f-2} g\left(j, \mu_j, T_1^j\right) \right) \Delta t,
\end{aligned} \tag{7.25}$$

where $R := \{x \in \mathbb{R} : T_{\min} \leq x \leq T_{\max}\}$. From the principle of optimality (Bellman and

Dreyfus (1962)) it follows that

$$\begin{aligned}
&\min_{\mu_{k-1},\cdots,\mu_{N_f-1}\in R} p_{\text{off}}\left(g\left(k-1,\mu_{k-1},T_1^{k-1}\right)+\sum_{j=k}^{N_f-1} g\left(j,\mu_j,T_1^j\right)\right)\Delta t \\
&= \min_{\mu_{k-1}\in R}\left(p_{\text{off}}\, g\left(k-1,\mu_{k-1},T_1^{k-1}\right)\Delta t+\min_{\mu_k,\cdots,\mu_{N_f-1}\in R} p_{\text{off}}\sum_{j=k}^{N_f-1} g\left(j,\mu_j,T_1^j\right)\right)\Delta t,
\end{aligned} \tag{7.26}$$

By combining (7.25) and (7.26) we have

$$\begin{aligned}
\min_{\overline{\mu}_{k-1}\in U_{k-1}} Q_{k-1}(\overline{\mu}_{k-1},T_1) = \min_{\mu_{k-1}\in R}&\left(p_{\text{off}}\, g\left(k-1,\mu_{k-1}),T_1^{k-1}\right)\Delta t\right. \\
&\left.+\min_{\mu_k,\cdots,\mu_{N_f-1}\in R} p_{\text{off}}\sum_{j=k}^{N_f-1} g\left(j,\mu_j,T_1^j\right)\right)\Delta t.
\end{aligned} \tag{7.27}$$

From Definition 8, we can write

$$\min_{\mu_k,\cdots,\mu_{N_f-1}} p_{\text{off}}\,\Delta t\sum_{j=k}^{N_f-1} g\left(j,\mu_j,T_1^j\right) = \min_{\overline{\mu}_k\in U_k} Q_k(\overline{\mu}_k,T_1). \tag{7.28}$$

Then, by combining (7.27) and (7.28) and using the induction hypothesis it follows that

$$\begin{aligned}
\min_{\overline{\mu}_{k-1}\in U_{k-1}} Q_{k-1}(\overline{\mu}_{k-1},T_1) &= \min_{\mu_{k-1}\in R}\left(p_{\text{off}}\, g\left(k-1,\mu_{k-1},T_1^{k-1}\right)\Delta t+\min_{\overline{\mu}_k\in U_k} Q_k(\overline{\mu}_k,T_1)\right) \\
&= \min_{\mu_{k-1}\in R}\left(p_{\text{off}}\, g\left(k-1,\mu_{k-1},T_1^{k-1}\right)\Delta t+V_k(T^k)\right)
\end{aligned}$$

for any $T^k\in\mathbb{R}^M$. By substituting for T^k from (7.6) and using the definition of V in (7.19) we have

$$\begin{aligned}
\min_{\overline{\mu}_{k-1}\in U_{k-1}} Q_{k-1}(\overline{\mu}_{k-1},T_1) = \min_{\mu_{k-1}\in R}&\left(p_{\text{off}}\, g\left(k-1,\mu_{k-1},T_1^{k-1}\right)\Delta t\right. \\
&\left.+V_k\left(f\left(T^{k-1},\mu_{k-1}\left(T^{k-1}\right)\right)\right)\right) = V_{k-1}\left(T^{k-1}\right)
\end{aligned}$$

for any $T^{k-1}\in\mathbb{R}^M$. By using the same logic it can be shown that

$$\min_{\overline{\mu}_{k-1}\in U_{k-1}} Q_{k-1}(\overline{\mu}_{k-1},T_1) = V_{k-1}(T^{k-1})$$

for any $k \in \{0, \cdots, N_{\text{off}} - 1\}$ and for any $T^{k-1} \in \mathbb{R}^M$. Therefore, by induction, (7.23) is true. Thus, $J^* = V_0(T^0)$. $\square$

The optimal temperature set-points for Problem (7.20) can be found as the sequence of minimizing arguments in the value function (7.19). However, this is not a solution to the original user-level optimization problem in (7.11), as the solution only applies for a fixed consumption bound, γ. However, as this consumption bound is scalar, we may apply a bisection on γ to solve the original optimization problem as formulated in (7.11). Details are presented in Algorithm 9. The computational complexity of this algorithm is proportional to $N_f \cdot n_s^M \cdot n_u$, where N_f is the number of discretization points in time, M is the state-space dimension of the discretized system in (7.6), n_s is the number of possible discrete values for each state, T and n_u is the number of possible discrete values for the control input (interior air temperature). In all of the case studies in Section 7.4, we use $N_f = 73, M = 3, n_s = n_u = 13$. The execution time of our Matlab implementation of Algorithm 9 for solving the three-day user-level problem on a Core i7 processor with 8 GB of RAM was less than 4.5 minutes.

Finding a solution to the 4-Setpoint thermostat programming problem (7.13) is significantly more difficult due to the presence of the switching times t_1, t_2, t_3 as decision variables. However, for this specific problem, a simple approach is to use Algorithm 9 as an inner loop for fixed t_i and then use a Monte Carlo search over t_i. For fixed t_i, our Matlab implementation for Algorithm 9 solves the 4-Setpoint thermostat programming problem in less than 17 seconds on a Core i7 processor with 8 GB of RAM. Our experiments on the same machine show that the total execution time for a Monte Carlo search over 300 valid (i.e., $t_i \leq t_{i+1}$) random combinations of t_1, t_2, t_3 is less than 1.41 hours.

To solve the utility-level problem in (7.15), we used Algorithm 9 as an inner loop for the Nelder-Mead simplex algorithm (Olsson and Nelson (1975)). The Nelder-Mead simplex algorithm is a heuristic optimization algorithm which is typically applied to problems where the derivatives of the objective function and/or constraint functions are unknown. Each iteration is defined by a reflection step and possibly a contraction or expansion step. The reflection begins by evaluation of the inner loop (Algorithm 9) at each of 4 vertices of a polytope. The polytope is then reflected about the hyperplane defined by the vertices with the best three objective values. The polytope is then either dilated or contracted depending on the objective value of the new vertex. In all of our case studies in Section 7.4, this hybrid algorithm achieved an error convergence of $< 10^{-4}$ in less than 15 iterations. Using a Core i7 machine with 8 GB of RAM, the execution time of the hybrid algorithm for solving the utility-level problem was less than 2.25 hours.

7.4 Policy implications and analysis

In this section, we use Algorithms 9 and 10 in three case studies to assess the effects of passive thermal storage, solar power and various cooling strategies on utility prices, peak demand and cost to the utility company.

In Case I, we compare our optimal thermostat program with other HVAC programming strategies and analyze the resulting peak demands and electricity bills for a set of electricity prices.

In Case II, we apply the Nelder-Mead simplex and Algorithm 9 to the user-level problem in (7.11) and the utility-level problem in (7.15) to compute optimal electricity prices and optimal cost of production.

In Case III, we first define an optimal thermostat program for solar users. Then, we examine the effect of solar power generation on the electricity prices of non-solar

Inputs: p_{on}, p_{off}, p_{d}, T_e, $t_{\text{on}}, t_{\text{off}}$, R_e, C_{in}, T_{init}, Δt, Δx, $T_{\min}, T_{\max}$, maximum number of bisection iterations $b_{\max}$, lower bound γ_l and upper bound γ_u for bisection search.

Main loop:

Set $k = 0$.

while $k < b_{\max}$ **do**

  Set $\gamma = \frac{\gamma_u + \gamma_l}{2}$.

  if V_0 *in* (7.19) *exists* **then**

    Calculate $u_0, \cdots, u_{N_f-1}$ as the minimizers of the RHS of (7.19) using a policy iteration technique.

    Set $\gamma_u = \gamma$. Set $u_i^\star = u_i$ for $i = 0, \cdots, N_{f-1}$.

  else

    Set $\gamma_l = \gamma$.

  end

  Set $k = k + 1$.

end

Outputs: Optimal interior temperature setting: $u_0^\star, \cdots, u_{N_f-1}^\star$.

Algorithm 9: A bisection/dynamic programming algorithm for optimal thermostat programming

users by solving a two-user single-utility optimization problem. We ran all cases for three consecutive days prorated from a one month billing cycle with the time-step $\Delta t = 1\ hr$, spatial-step $\Delta x = 0.1\ m$ and with the building parameters in Table 7.1. These parameters were determined using the model calibration procedure described in Section 7.2.2. We used an external temperature profile measured for three typical summer days in Phoenix, Arizona (see Figure 7.6). For each day, the on-peak period starts at 12 PM and ends at 7 PM. We used min and max interior temperatures as $T_{\min} = 22°C$ and $T_{\max} = 28°C$.

Inputs: a, b, reflection parameter θ, expansion parameter κ, contraction parameter ζ, reduction parameter τ, initial prices $p^0_{d_i}, p^0_{\text{on}_i}$ such that $p^0_{d_i} + p^0_{\text{on}_i} < 1$ for $i = 1, \cdots, 4$, stopping threshold ϵ and inputs to Algorithm 9.

Initialization:

Set $p_{d_i} = p^0_{d_i}$, $p_{\text{on}_i} = p^0_{\text{on}_i}$, $p_{\text{off}_i} = 1 - p_{\text{on}_i} - p_{d_i}$, $p_{\text{old}_i} = [10^{12}, 10^{12}, 10^{12}]$ for $i = 1, \cdots, 4$.

Main loop:

while $\sum_{i=1}^4 \|p_{old_i} - p_i\|_2^2 > \epsilon$ **do**

  for $i = 1, \cdots, 4$ **do**

    Calculate opt. temp. setting $u^\star_{0,i}, \cdots, u^\star_{N_f-1,i}$ associated with prices p_i using Alg. 9.

    Calculate the cost C_i associated with $u^\star_{0,i}, \cdots, u^\star_{N_f-1,i}$ using (7.16) and (7.14).

  end

  Re-index the costs and their associated prices and temperature settings such that $C_1 \leq C_2 \leq C_3 \leq C_4$. Set $p^\star = p_1$.

  Calculate the centroid of all the prices as $\bar{p} = \left[\sum_{i=1}^4 p_{\text{off}_i}, \sum_{i=1}^4 p_{\text{on}_i}, \sum_{i=1}^4 p_{d_i}\right]$.

  Calculate reflected prices as $p_r = \bar{p} + \theta(\bar{p} + p_4)$.

  Calculate optimal temperature setting $u^\star_{0,r}, \cdots, u^\star_{N_f-1,r}$ and the cost C_r associated with the reflected prices using Algorithm 9.

  if $C_1 \leq C_r < C_3$ **then**

    Set $p_{\text{old}_4} = p_4$ and $p_4 = p_r$. Go to the beginning of the loop.

  else if $C_r < C_1$ **then**

    Calculate expanded prices as $p_e = \bar{p} + \kappa(\bar{p} + p_4)$.

    Calculate optimal temperature setting $u^\star_{0,e}, \cdots, u^\star_{N_f-1,e}$ and the cost C_e associated with expanded prices using Algorithm 9.

    if $C_e < C_r$ **then** Set $p_{\text{old}_4} = p_4$ and $p_4 = p_e$. Set $p^\star = p_e$.

    else Set $p_{\text{old}_4} = p_4$ and $p_4 = p_r$. Set $p^\star = p_r$.

    Go back to While loop.

  else

    Calculate contraction prices as $p_c = \bar{p} + \xi(\bar{p} + p_4)$.

    Calculate optimal temperature setting $u^\star_{0,c}, \cdots, u^\star_{N_f-1,c}$ and the cost C_c associated with expanded prices using Algorithm 9.

    if $C_c < C_4$ **then** Set $p_{\text{old}_4} = p_4$ and $p_4 = p_c$. Go back to While loop.

    else Set $p_{\text{old}_i} = p_i$ for $i = 2, 3, 4$. Update prices as $p_i = p_1 + \tau(p_i - p_1)$ for $i = 2, 3, 4$.

  end

end

Outputs: Optimal electricity prices $p^\star$.

Algorithm 10: An algorithm for computing optimal electricity prices

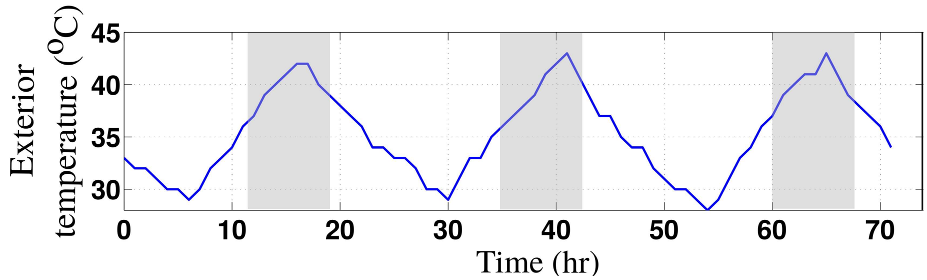


Figure 7.6: External temperature of three typical summer days in Phoenix, Arizona. Shaded areas correspond to on-peak hours.

Table 7.1: Building's parameters as determined in Section 7.2.1

$L_{in}(m)$	$\alpha(m^2/s)$	$R_e(K/W)$	$C_{in}(Wm/K)$	$\Delta x(m)$
0.4	8.3×10^{-7}	0.0015	45	0.1

7.4.1 Effect of electricity prices on peak demand and production costs

In this case, we first applied Algorithm 9 to the optimal and 4-Setpoint thermostat programming problems (See (7.11) and (7.13)) for a non-solar customer using the electricity prices determined by APS in Table 7.2 (Rumolo (2013)). The resulting electricity bills are given in Table 7.3 as the total cost paid for three days prorated from a one month billing cycle with the external temperature profile shown in Figure 7.6. Prorated in this case means that for a 30-day month, the bill is one-tenth of the monthly bill based on repetition of the three-day cycle ten times. Practically, what this means is that the period in Problems (7.11) and (7.13) is tripled while the demand charge in Problems (7.11) and (7.13) uses a demand price $\frac{1}{10}\, p_d = 1.35\frac{\$}{kW}$. For comparison, we have solved Problem (7.11) using the general-purpose optimization solver GPOPS (Patterson and Rao (2013)). We have also compared our results with a naive strategy of setting the temperature to $T_{\max}$ (constant) and a pre-cooling strategy with the temperature setting: $u = 25°C$ from 12 AM to 8 AM; $u = T_{\min} = 22°C$ from 8 AM to 12 PM; $u = T_{\max} = 28°C$ from 12 PM to 8 PM; $u = 25°C$ from 8 PM to 12 AM. As can be seen from Table 7.3, our algorithm offers significant improvement

Table 7.2: On-peak, off-peak & demand prices of Arizona utility APS

	On-peak ($\frac{\$}{kWh}$)	Off-peak ($\frac{\$}{kWh}$)	Demand ($\frac{\$}{kWh}$)
APS	0.089	0.044	13.50

over heuristic approaches. The power consumption and the temperature setting as a function of time for each strategy can be found in Figure 7.7. For convenience, the on-peak and off-peak intervals are indicated on the figure.

To examine the impact of changes in electricity prices on peak demand, we next chose several different prices corresponding to high, medium and low penalties for peak demand. Again, in each case, our algorithms (optimal and 4-setpoint) are compared to GPOPS and the same pre-cooling strategy. The results are summarized in Table 7.4. Note that for brevity, in this section, we refer to the total cost of generation, transmission and distribution as simply production cost. For each price, the smallest computed production cost and associated demand peak are listed in bold. The power consumption and the temperature settings as a function of time for the optimal and 4-Setpoint strategies can be found in Figures 7.8 and 7.9. For the optimal strategy, notice that by increasing the demand penalty, relative to the low-penalty case, the peak consumption is reduced by 14% and 23% in the medium and high penalty cases respectively. Furthermore, notice that by using the optimal strategy and the high demand-limiting prices, we have reduced the demand peak by 29% with respect to the constant strategy in Table 7.3. Of course, a moderate reduction in peak demand at the expense of large additional production costs may not be desirable. Indeed, the question of optimal distribution of electricity prices for minimizing the production cost is discussed in Case II.

Table 7.3: CASE I: Electricity bills (for three days) and demand peaks for different strategies. Electricity prices are from APS.

Temperature setting	Electricity bill ($)	Demand peak (kW)
Optimal (Theorem 21)	**36.58**	9.222
GPOPS (Patterson and Rao (2013))	37.03	9.155
4-Setpoint (Theorem 21)	37.71	9.401
Pre-cooling	39.23	**8.803**
Constant	39.42	10.462

Table 7.4: CASE I: Costs of production (for three days) and Demand peaks for various prices and strategies. Prices are non-regulated and SRP's coefficients of utility cost are: τ =0.00401 \$/(MWh)^2, ν =4.54351 \$/(MWh)

	Prices $[p_{\text{off}}, p_{\text{on}}, p_d]$	Demand-limiting	Production cost	Demand peak
Optimal	$[0.007, 0.010, 13.616]$	high	**46.78**\$ ($0.086\frac{\$}{kWh}$)	**7.4132** kW
Optimal	$[0.015, 0.045, 13.573]$	medium	**51.56**\$ ($0.116\frac{\$}{kWh}$)	**8.2898** kW
Optimal	$[0.065, 0.095, 13.473]$	low	**59.42**\$ ($0.168\frac{\$}{kWh}$)	9.6749 kW
4-Setpoint	$[0.007, 0.010, 13.616]$	high	53.47\$ ($0.114\frac{\$}{kWh}$)	8.5914 kW
4-Setpoint	$[0.015, 0.045, 13.573]$	medium	55.19\$ ($0.130\frac{\$}{kWh}$)	8.910 kW
4-Setpoint	$[0.065, 0.095, 13.473]$	low	61.24\$ ($0.169\frac{\$}{kWh}$)	9.974 kW
GPOPS	$[0.007, 0.010, 13.616]$	high	49.53\$ ($0.109\frac{\$}{kWh}$)	7.9440 kW
GPOPS	$[0.015, 0.045, 13.573]$	medium	56.48\$ ($0.142\frac{\$}{kWh}$)	9.1486 kW
GPOPS	$[0.065, 0.095, 13.473]$	low	59.19\$ ($0.159\frac{\$}{kWh}$)	9.6221 kW
Precooling	$[0.007, 0.010, 13.616]$	high	54.75\$ ($0.116\frac{\$}{kWh}$)	8.8031 kW
Precooling	$[0.015, 0.045, 13.573]$	medium	54.75\$ ($0.116\frac{\$}{kWh}$)	8.8031 kW
Precooling	$[0.065, 0.095, 13.473]$	low	**54.75**\$ ($0.116\frac{\$}{kWh}$)	**8.8031** kW

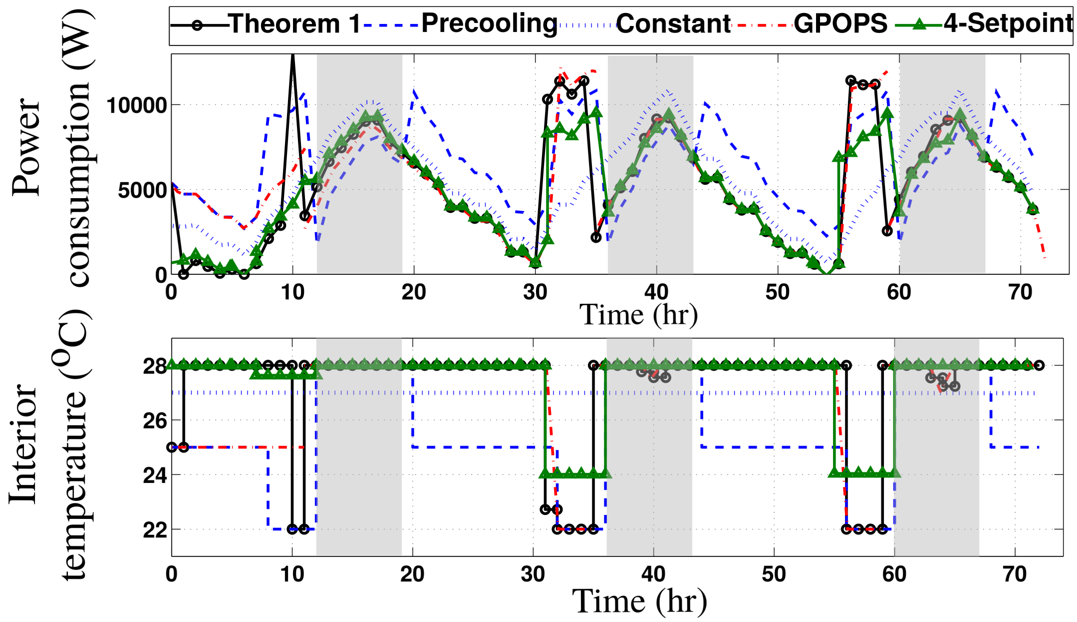


Figure 7.7: CASE I: Power consumption and temperature settings for various programming strategies using APS's rates.

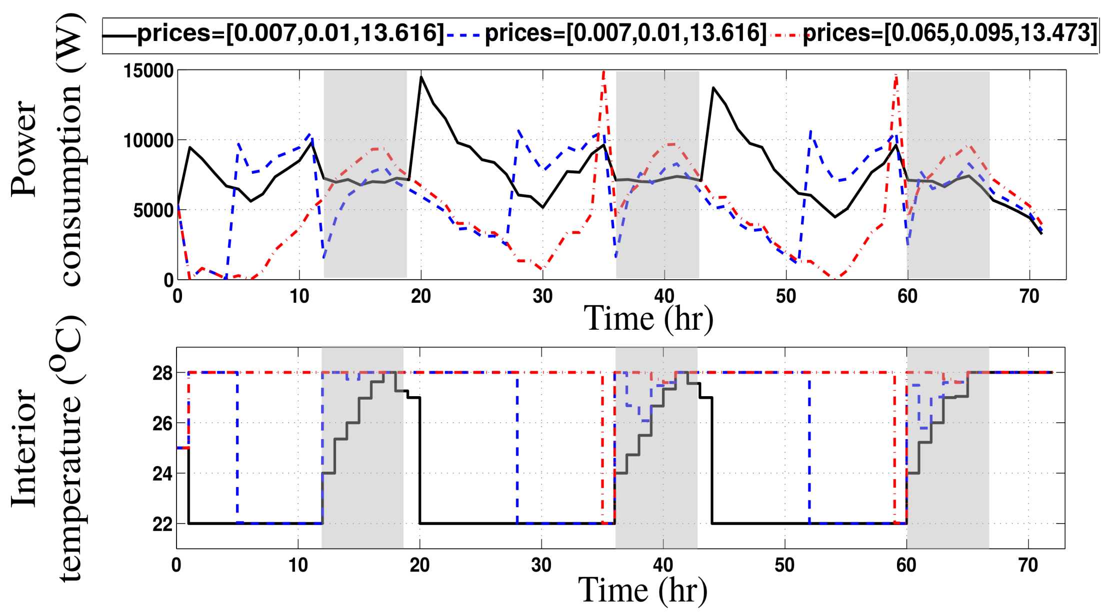


Figure 7.8: CASE I: Power consumption and optimal temperature settings for high, medium and low demand penalties. Shaded areas correspond to on-peak hours.

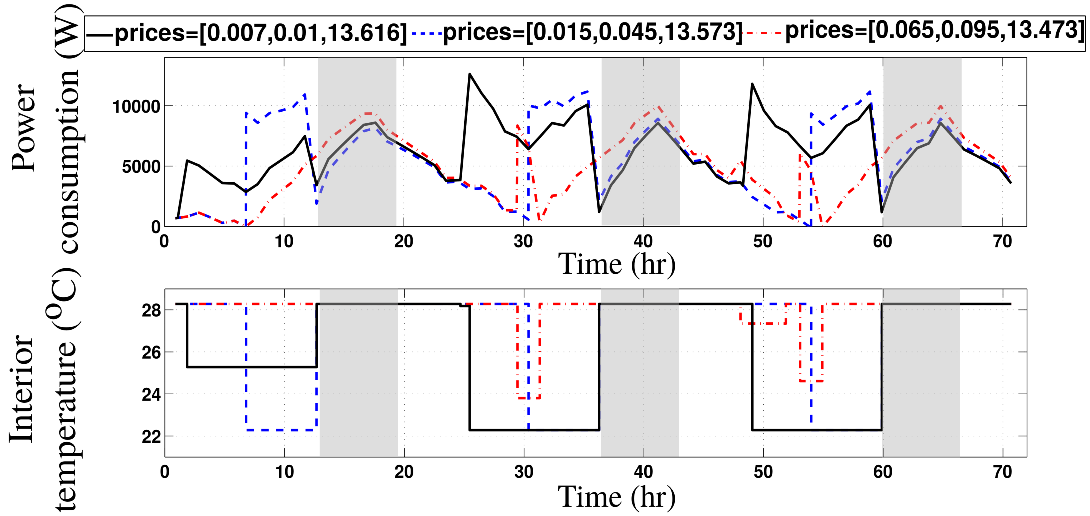


Figure 7.9: CASE I: Power consumption and temperature settings for high, medium and low demand penalties using 4-Setpoint thermostat programming.

7.4.2 Optimal thermostat programming with optimal electricity prices

In this case, we consider the quadratic model of fuel cost defined in Section (7.17). A typical pricing strategy for SRP and other utilities is to set prices proportional to marginal production costs. SRP estimates the mean marginal fuel cost at $a = 0.0814\$/kWh$ (See (7.16)). Linearizing our quadratic model of fuel cost and equating to this estimate of the marginal cost yields an estimate of the mean load. Dividing this mean load by the aggregate user defined in Case I yields an estimate of the mean number of users of this class at $N = 24,405$.

To compare the marginal pricing strategy with the optimal pricing strategy, we use this mean number of users in the utility optimization problem under the assumption that the building parameters in Section 7.2.2 represent a single aggregate rational user. The resulting optimal prices, associated production cost, and associated peak demand are listed in Table 7.5. For comparison, we also included in Table 7.5 the production cost and demand peak for the same rational user subject to prices based solely on the marginal costs, where these prices are scaled so that revenue equals costs. In other words, we solved (7.15) for $\lambda = 1$, meaning that the regulated utility does not make any profit from generation, transmission and distribution.

From Table 7.5, optimal pricing results in a slight reduction ($82,000) in production costs. The discrepancy between optimal prices and marginal costs may be surprising given that both the user and utility are trying to minimize the cost of electricity. However, there are several reasons for this difference. The first and most obvious reason is that the price structure for the user and the cost structure for the utility are not perfectly aligned. In the first place, the utility has a quadratic in consumption model for costs, where the user has a linear model. The second misalignment is that the capacity cost for the utility is calculated as a maximum over 24 hours and the demand charge for the user is calculated only during peak hours. An additional reason that marginal costs will not always be optimal prices is nonlinearity of the cost function and heterogeneity of the users. To see this, suppose that cost function exactly equaled the price function for each user. The problem in this case is that the sum of the individual bills is NOT equal to the total production cost. This can be seen in the demand charge, where $\sup_x f(x) + \sup_x g(x) \neq \sup_x(f(x) + g(x))$.

Table 7.5: CASE II: Production costs (for three days) and demand peaks associated with regulated optimal electricity prices (calculated by Algorithm 10) and SRP's electricity prices. SRP's marginal costs: $a = 0.0814\frac{\$}{kWh}, b = 59.76\frac{\$}{kW}$

Strategy	$[p_{\text{off}}(\frac{\$}{kWh}), p_{\text{on}}(\frac{\$}{kWh}), p_d(\frac{\$}{kW})]$	Production cost	Demand peak
Optimal	$[0.0564, 0.0667, 51.1859]$	**1,595,309** $	195.607 MW
SRP	$[0.0668, 0.0668, 49.0018]$	1,677,516 $	211.79 MW

7.4.3 *Optimal thermostat programming for solar customers - impact of distributed solar generation on non-solar customers*

We now evaluate the impact of solar power on the bills of non-solar users in a regulated electricity market. We consider a network consisting of a utility company and two aggregate users - one solar and one non-solar. For the non-solar user, we define optimal thermostat programming as in (7.11). For the solar user, the optimal thermostat programming problem is as defined in (7.11), where we have now redefined the consumption function as

$$g(k, u_k, T_1^k) := \frac{T_e^k - u_k}{R_e} + 2\, C_{in} \frac{T_1^k - u_k}{\Delta x} - Q_k, \tag{7.29}$$

where Q_k is the power supplied locally by solar panels. We assume that solar penetration is 50%, so that both aggregate users contribute equally to revenue and costs to the utility. For Q_k, we used data generated on June 4-7 from a typical 13kW south-facing rooftop PV array in Scottsdale, AZ. We applied Algorithm 10 separately to each user, while considering (7.16) as the utility cost model. The results are presented in Table 7.6. For comparison, we have also included optimal prices, prorated electricity bills over three days and demand peaks of both users. From Table 7.6 we observe that the difference between the electricity bill of a non-solar user in a single-user network and the bill of a non-solar user in a two-user network (solar and non-solar) is $< 2\%$. This increase in bill for the solar user is $< 8\%$. The utility-generated power, solar-generated power and optimal temperature settings are shown in Figure 7.10.

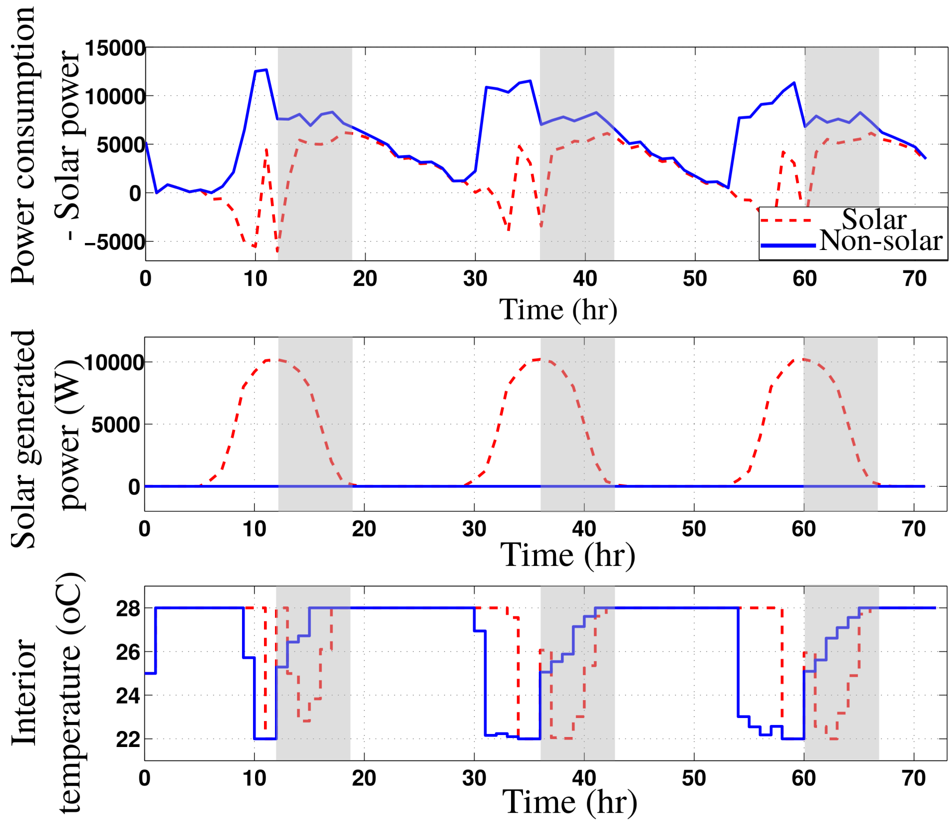


Figure 7.10: CASE III: Power consumption, solar generated power and optimal temperature settings for the non-solar and solar users.

Table 7.6: CASE III: Optimal electricity prices, bills (for three days) and demand peaks for various customers. Marginal costs from SRP: $a = 0.0814\frac{\$}{kWh}, b = 59.76\frac{\$}{kW}$

Customers	$[p^\star_{\text{off}}(\frac{\$}{kWh}), p^\star_{\text{on}}(\frac{\$}{kWh}), p^\star_d(\frac{\$}{kW})]$	Elect. Bill	Demand peak
Solar &	$[0.089, 0.115, 51.988]$	\$ 50.052	6.1947 kW
Non-solar		\$ 84.717	8.6787 kW
Single Non-solar	$[0.081, 0.108, 54.004]$	\$ 83.333	8.3008 kW
Single Solar	$[0.088, 0.118, 58.556]$	\$ 54.311	6.1916 kW

Chapter 8

SUMMARY, CONCLUSIONS AND FUTURE DIRECTIONS OF OUR RESEARCH

8.1 Summary and conclusions

Thanks to the development of converse Lyapunov theory, a broad class of problems in stability analysis and control can be formulated as optimization of polynomials. In this dissertation, we focus on design and implementation of parallel algorithms for optimization of polynomials. In Chapter 1, we provide a brief overview of the existing state-of-the-art algorithms for solving polynomial optimization and optimization of polynomials problems. As our contribution, we chose to design our optimization algorithms based on two well-known results in algebraic geometry, known as Polya's theorem and Handelman's theorem. Our motivation behind this choice is that these theorems define structured parameterizations[1] for positive polynomials - a property that our parallel algorithms exploit to achieve near-linear speed-up and scalability.

In Chapter 2, we discuss how variants of Polya's theorem, Handelman's theorem and the Positivstellensatz results can be applied to optimization of polynomials defined over various compact sets, e.g., simplex, hypercube, convex polytopes and semi-algebraic sets. We show that applying these theorems to an optimization of polynomials problem yields convex optimization problems in the form of LPs and/or SDPs. By solving these LPs and SDPs, one can find asymptotic solutions to the optimization of polynomials problems (as we define in (2.12)). Subsequently, by combining this method with a branch-and-bound algorithm, one can find solutions to

[1] A detailed discussion on the structure of these parameterizations can be found in Section 4.3.3

polynomial optimization problems (as we define in (2.10)).

In Chapter 3, we briefly review Newton-based descent algorithms for constrained optimization of convex functions. In particular, we discuss a state-of-the-art primal-dual interior-point path-following algorithm (Helmberg *et al.* (2005)) for solving semi-definite programs. We later decentralize the computation of the search directions of this algorithm to design a parallel SDP solver in Chapter 4.

In Chapter 4, we propose a parallel-computing approach to stability analysis of large-scale linear systems of the form $\dot{x}(t) = A(\alpha)x(t)$, where A is a real-valued polynomial, $\alpha \in \Delta^l \subset \mathbb{R}^l$ and $x \in \mathbb{R}^n$. This approach is based on mapping the structure of the SDPs associated with Polya's theorem to a parallel computing environment. We first design a parallel set-up algorithm with no centralized computation to construct the SDPs associated with Polya's theorem. We then show that by choosing a block-diagonal starting point for the SDP algorithm in Helmberg *et al.* (2005), the primal and dual search directions will preserve their block-diagonal structure at every iteration. By exploiting this property, we decentralize the computation of the search directions - the most computationally expensive step of an SDP algorithm. The result is a parallel algorithm which under certain conditions, can solve the NP-hard problem of robust stability of linear systems at the same per-core computational cost as solving the Lyapunov inequality for a linear system with no parametric uncertainty. Theoretical and experimental results verify near-linear speed-up and scalability of our algorithm for up to 200 processors. In particular, our numerical tests on cluster computers show that our MPI/C++ implementation of the SDP algorithm outperforms the existing state-of-the-art SDP solvers such as SDPARA (Yamashita *et al.* (2003)) in terms of speed-up. Moreover, our experimental tests on a mid-size (9-node) Linux-based cluster computer demonstrate the ability of our algorithm in performing robust stability analysis of systems with 100+ states and several uncertain parameters. A comprehensive complexity analysis of both set-up and solver algorithms can be found in Sections 4.4 and 4.6.1.

In Chapter 5, we further extend our analysis to consider linear systems with uncertain parameters inside hypercubes. We propose an extended version of Polya's theorem for positivity over a multi-simplex (Cartesian product of standard simplicies). We claim that every polynomial defined over a hypercube has an equivalent homogeneous representation over the multi-simplex. Therefore, our the multi-simplex version of Polya's theorem can be used to verify positivity over hypercubes. In the next step, we generalize our parallel set-up algorithm from Chapter 4 to construct the SDPs associated with our multi-simplex version of Polya's theorem. Our complexity analysis shows that for sufficiently large number of available processors, at each Polya's iteration, the per processor computation and communication cost of the algorithm scales polynomially with the number of states and uncertain parameters. Through numerical experiments on a large cluster computer, we show that the algorithm can achieve a near-perfect speed-up.

In Chapter 6, we extend our approach to consider optimization of polynomials defined over more complicated geometries such as convex polytopes. Specifically, we apply Handelman's theorem to construct piecewise polynomial Lyapunov functions for nonlinear dynamical systems defined by polynomial vector fields. Unfortunately, neither Polya's theorem nor Handelman's theorem can readily certify non-negativity of polynomials which have zeros in the interior of a simplex/polytope. Our proposed solution to this problem is to decompose the domain of analysis (in this case a polytope) into several convex sub-polytopes with a common vertex at the equilibrium. Then, by using Handelman's theorem, we derive a new set of affine feasibility conditions - solvable by linear programming - on each sub-polytope. Any solution to this feasibility problem yields a piecewise polynomial Lyapunov function on the entire polytope. In a computational complexity analysis, we show that for large number of states and large degrees of the Lyapunov function, the complexity of the proposed feasibility problem is less than the complexity of certain semi-definite programs associated with alternative methods based on Sum-of-Squares and Polya's theorem.

Finally, in chapter 7, we address a real-world optimization problem in energy planning and smart grid control. We consider the coupled problems of optimal control of HVAC systems for residential customers and optimal pricing of electricity by utility companies. Our framework consists of multiple users (solar and non-solar customers) and a single regulated utility company. The utility company sets prices for the users, who pay for both total energy consumed (\$/kWh, including peak and off-peak rates) and the peak rate of consumption in a month (a demand charge) (\$/kW). The cost of electricity for the utility company is based on a combination of capacity costs (\$/kW) and fuel costs (\$/kWh). On the demand side, the users minimize the amount paid for electricity while staying within a pre-defined temperature range. The users have access to energy storage in the form of thermal capacitance of interior structures. Meanwhile, the utility sets prices designed to minimize the total cost of generation, transmission and distribution of electricity. To solve the user-level problem, we use a variant of dynamic programming. To solve the utility-level problem, we use the Nelder-Mead simplex algorithm coupled with our dynamic programming code - yielding optimal on-peak, off-peak and demand prices. We then apply our algorithms to a variety of scenarios in which show that: 1) Thermal storage and optimal thermostat programming can reduce electricity bills using current rates from utilities Arizona Public Service (APS) and Salt River Project (SRP). 2) Our optimal pricing can reduce the total cost to the utility companies. 3) In the presence of demand charges, the impact of distributed solar generation on the electricity bills of the non-solar users is not significant ($< \mathbf{2}\%$).

8.2 Future directions of our research

In the following sections, we discuss how the proposed algorithms in this dissertation can be extended to solve three well-known problems in controls: 1) Robust stability analysis of nonlinear systems; 2) Synthesis of parameter-varying $\mathcal{H}_\infty$-optimal controller; 3) Computing value functions in approximate dynamic programming problems.

8.2.1 *A Parallel algorithm for nonlinear stability analysis using Polya's theorem*

Consider the problem of local stability analysis of a nonlinear system of the form

$$\dot{x}(t) = A(x, \alpha)x(t), \tag{8.1}$$

where $A : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n \times n}$ is a matrix-valued polynomial and $A(0, 0) \neq 0$. From converse Lyapunov theory, this problem can be expressed as a search for a polynomial $V : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ which satisfies the Lyapunov inequalities

$$W_1(x, \alpha) \leq V(x, \alpha) \leq W_2(x, \alpha)$$

$$\langle \nabla_x V, f \rangle \leq -W_3(x, \alpha)$$

for all $x, \alpha \in \Omega \subset \mathbb{R}$, where $0 \in \Omega$. However, as we discussed in Section 6.1, Polya's theorem (simplex and multi-simplex versions) cannot certify positivity of polynomials which have zeros in the interior of the unit- and/or multi-simplex. Moreover, if $F(x)$ in (5.2) has a zero in the interior of Φ^n, then any multi-homogeneous polynomial $P(x, y)$ that satisfies (5.2) has a zero in the interior of the multi-simplex $\Delta^2 \times \cdots \times \Delta^2$ - hence cannot be parameterized by Polya's theorem. One way to enforce the condition $V(\mathbf{0}, \mathbf{0}) = 0$ is to search for coefficients of a matrix-valued polynomial P which defines a Lyapunov function of the form $V(x, \alpha) = x^T P(x, \alpha)x$. It can be shown that $V(x,\alpha) = x^T P(x,\alpha)x$ is a Lyapunov function for System (8.1) if and only if γ^* in the following optimization of polynomials problem is positive.

$$\begin{aligned} \gamma^* = &\max_{\gamma\in\mathbb{R},\alpha,P\in\mathbb{R}[x,\alpha]} \quad \gamma \\ &\text{subject to} \begin{bmatrix} P(x,\alpha) & 0 \\ 0 & -Q(x,\alpha) \end{bmatrix} - \gamma I \geq 0 \quad \text{for all } x \in \Phi^n,\ \alpha \in \Phi^m, \end{aligned} \tag{8.2}$$

where

$$\begin{aligned} Q(x,\alpha) = A^T(x,\alpha)P(x,\alpha) &+ P(x,\alpha)A(x,\alpha) \\ &+ \frac{1}{2}\left(A^T(x,\alpha) \begin{bmatrix} x^T \frac{\partial P(x,\alpha)}{\partial x_1} \\ \vdots \\ x^T \frac{\partial P(x,\alpha)}{\partial x_n} \end{bmatrix} + \begin{bmatrix} x^T \frac{\partial P(x,\alpha)}{\partial x_1} \\ \vdots \\ x^T \frac{\partial P(x,\alpha)}{\partial x_n} \end{bmatrix}^T A(x,\alpha) \right). \end{aligned}$$

As we discussed in Section 2.3.2, by applying bisection search on γ and using the multi-simplex version of Polya's theorem (Theorem 16) as a test for feasibility of Constraint (8.2), we can compute lower bounds on γ^*. Suppose there exists a matrix-valued multi-homogeneous polynomial S of degree vector $d_s \in \mathbb{N}^n$ ($d_s = [d_{s_1}, \cdots, d_{s_n}, d_{s_{n+1}}, \cdots, d_{s_{n+m}}]$, where for $i \in \{1, \cdots, n\}$, d_{s_i} is the degree of y_i and for $i \in \{n+1, \cdots, m\}$, d_{s_i} is the degree of β_i) such that

$$\begin{aligned} &\{P(x,\alpha) \in \mathbb{S}^n : x \in \Phi^n,\ \alpha \in \Phi^m\} = \\ &\{S(y,z,\beta,\eta) \in \mathbb{S}^n : (y_i,z_i),(\beta_j,\eta_j) \in \Delta^2, i = 1,\cdots,n, \text{and } j = 1,\cdots,m\}. \end{aligned}$$

Likewise, suppose there exist matrix-valued multi-homogeneous polynomials B and C of degree vectors $d_b \in \mathbb{N}^n$ and $d_c = d_s \in \mathbb{N}^n$ such that

$$\begin{aligned} &\{A(x,\alpha) \in \mathbb{R}^{n\times n} : x \in \Phi^n\} = \\ &\{B(y,z,\beta,\eta) \in \mathbb{R}^{n\times n} : (y_i,z_i),(\beta_j,\eta_j) \in \Delta^2, i = 1,\cdots,n, \text{ and } j = 1,\cdots,m\} \end{aligned}$$

and

$$\left\{\left[\frac{\partial P(x,\alpha)}{\partial x_1}x, \cdots, \frac{\partial P(x,\alpha)}{\partial x_n}x\right] \in \mathbb{R}^{n\times n} : x \in \Phi^n, \text{ and } \alpha \in \Phi^m\right\} =$$
$$\left\{C(y,z,\beta,\eta) \in \mathbb{R}^{n\times n} : (y_i, z_i), (\beta_j, \eta_j) \in \Delta^2, i = 1, \cdots, n, \text{ and } j = 1, \cdots, m\right\}.$$

Given $\gamma \in \mathbb{R}$, it follows from Theorem 16 that the inequality condition in (8.2) holds for all $\alpha \in \Phi^l$ if there exist $e \geq 0$ such that

$$\left(\prod_{i=1}^{n}(y_i + z_i)^e \cdot \prod_{j=1}^{m}(\beta_j + \eta_j)^e\right)\left(S(y,z,\beta,\eta) - \gamma I\left(\prod_{i=1}^{n}(y_i + z_i)^{d_{p_i}} \cdot \prod_{j=1}^{m}(\beta_i + \eta_i)^{\hat{d}_{p_i}}\right)\right) \tag{8.3}$$

and

$$\begin{aligned}\left(\prod_{i=1}^{n}(y_i + z_i)^e \cdot \prod_{j=1}^{m}(\beta_j + \eta_j)^e\right)\Big(B^T(y,z,\beta,\eta)S(y,z,\beta,\eta) + S(y,z,\beta,\eta)B(y,z,\beta,\eta)& \\ +\frac{1}{2}\left(B^T(y,z,\beta,\eta)C^T(y,z,\beta,\eta) + C(y,z,\beta,\eta)B(y,z,\beta,\eta)\right)& \\ -\gamma I\left(\prod_{i=1}^{n}(y_i + z_i)^{d_{q_i}} \cdot \prod_{j=1}^{m}(\beta_i + \eta_i)^{\hat{d}_{q_i}}\right)\Big)&\end{aligned} \tag{8.4}$$

have all positive coefficients, where d_{p_i} and $\hat{d}_{p_i}$ are the degrees of x_i and α_i in $P(x,\alpha)$, and d_{q_i} and $\hat{d}_{q_i}$ are the degrees of x_i and α_i in $Q(x,\alpha)$. Now, let S, B and C be of the following forms.

$$S(y,z,\beta,\eta) = \sum_{\substack{h,g\in\mathbb{N}^{n+m}\\ h+g=d_s}} S_{h,g} y_1^{h_1} z_1^{g_1} \cdots y_n^{h_n} z_n^{g_n} \beta_1^{h_{n+1}} \eta_1^{g_{n+1}} \cdots \beta_m^{h_{n+m}} \eta_m^{g_{n+m}} \tag{8.5}$$

$$B(y,z,\beta,\eta) = \sum_{\substack{h,g\in\mathbb{N}^{n+m}\\ h+g=d_b}} B_{h,g} y_1^{h_1} z_1^{g_1} \cdots y_n^{h_n} z_n^{g_n} \beta_1^{h_{n+1}} \eta_1^{g_{n+1}} \cdots \beta_m^{h_{n+m}} \eta_m^{g_{n+m}} \tag{8.6}$$

$$C(y,z,\beta,\eta) = \sum_{\substack{h,g\in\mathbb{N}^{n+m}\\ h+g=d_c}} C_{h,g} y_1^{h_1} z_1^{g_1} \cdots y_n^{h_n} z_n^{g_n} \beta_1^{h_{n+1}} \eta_1^{g_{n+1}} \cdots \beta_m^{h_{n+m}} \eta_m^{g_{n+m}} \tag{8.7}$$

Note that the coefficients $C_{h,g}$ can be written as linear combinations of $S_{h,g}$. For brevity we have denoted $C_{h,g}(S_{h,g})$ as $C_{h,g}$. By combining (8.5), (8.6) and (8.7) with (8.3) and (8.4) it follows that for a given $\gamma \in \mathbb{R}$, the inequality condition in (8.2) holds for all $\alpha \in \Phi^n$ if there exist some $e \geq 0$ such that

$$\sum_{\substack{h,g \in \mathbb{N}^{n+m} \\ h+g=d_s}} f_{\{q,r\},\{h,g\}} S_{h,g} > 0 \quad \text{for all} \;\; q,r \in \mathbb{N}^{n+m} : q+r = d_s + 2\, e \cdot \mathbf{1}_{n+m} \tag{8.8}$$

and

$$\sum_{\substack{h,g \in \mathbb{N}^{n+m} \\ h+g=d_s}} M^T_{\{u,v\},\{h,g\}} S_{h,g} + S_{h,g} M_{\{u,v\},\{h,g\}} + N^T_{\{u,v\},\{h,g\}} C^T_{h,g} + C_{h,g} N_{\{u,v\},\{h,g\}} < 0 \tag{8.9}$$

for all $u,v \in \mathbb{N}^{n+m} : u+v = d_s + d_b + 2\, e \cdot \mathbf{1}_{n+m}$, where we define $f_{\{q,r\},\{h,g\}}$ to be the coefficient of

$$S_{h,g}\, y_1^{q_1} z_1^{r_1} \cdots y_n^{q_n} z_n^{r_n} \beta_1^{q_{n+1}} \eta_1^{r_{n+1}} \cdots \beta_m^{q_{n+m}} \eta_m^{r_{n+m}}$$

after substituting (8.5) into (8.3). Likewise, we define $M_{\{u,v\},\{h,g\}}$ to be the coefficient of

$$S_{h,g}\, y_1^{u_1} z_1^{v_1} \cdots y_n^{u_n} z_n^{v_n} \beta_1^{u_{n+1}} \eta_1^{v_{n+1}} \cdots \beta_m^{u_{n+m}} \eta_m^{v_{n+m}}$$

and $N_{\{u,v\},\{h,g\}}$ to be the coefficient of

$$C_{h,g}\, y_1^{u_1} z_1^{v_1} \cdots y_n^{u_n} z_n^{v_n} \beta_1^{u_{n+1}} \eta_1^{v_{n+1}} \cdots \beta_m^{u_{n+m}} \eta_m^{v_{n+m}}$$

after substituting (8.6) and (8.7) into (8.4). For any $\gamma \in \mathbb{R}$, if there exist $e \geq 0$ and $\{S_{h,g}\}$ such that Conditions (8.8) and (8.9) hold, then γ is a lower bound for γ^* as defined in (8.2). Furthermore, if γ is positive, then origin is an asymptotically stable equilibrium for System (8.1). Fortunately, Conditions (8.8) and (8.9) form an SDP with a block-diagonal structure - hence an algorithm similar to Algorithm 7 can be developed to set-up the SDP in parallel. Furthermore, our parallel SDP solver in Section 4.5 can be used to efficiently solve the SDP.

8.2.2 *Parallel computation for parameter-varying $\mathcal{H}_\infty$-optimal control synthesis*

Algorithm 5 can be generalized to consider a more general class of feasibility problems, i.e.,

$$\sum_{i=1}^{N} \big(A_i(\alpha)X(\alpha)B_i(\alpha) + B_i^T(\alpha)X(\alpha)A_i^T(\alpha) + R_i(\alpha)\big) < -\gamma I \quad \text{for all} \;\; \alpha \in \Delta^l,$$

where A_i, B_i and R_i are polynomials. Formulations such as this can be used to solve a wide variety of problem in systems analysis and control such as $\mathcal{H}_\infty$-optimal control synthesis for systems with parametric uncertainty. To see this, consider a plant G with the state-space formulation

$$\dot{x}(t) = A(\alpha)x(t) + \begin{bmatrix} B_1(\alpha) \; B_2(\alpha) \end{bmatrix} \begin{bmatrix} \omega(t) \\ u(t) \end{bmatrix},$$

$$\begin{bmatrix} z(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} C_1(\alpha) \\ C_2(\alpha) \end{bmatrix} x(t) + \begin{bmatrix} D_{11}(\alpha) & D_{12}(\alpha) \\ D_{21}(\alpha) & 0 \end{bmatrix} \begin{bmatrix} \omega(t) \\ u(t) \end{bmatrix}, \tag{8.10}$$

where $\alpha \in Q \subset \mathbb{R}^l$, $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$ is the control input, $\omega(t) \in \mathbb{R}^p$ is the external input and $z(t) \in \mathbb{R}^q$ is the external output. Suppose $(A(\alpha), B_2(\alpha))$ is stabilizable and $(C_2(\alpha), A(\alpha))$ is detectable for all $\alpha \in Q$. According to P. Gahinet (1994) there exists a state feedback gain $K(\alpha) \in \mathbb{R}^{m\times n}$ such that

$$\|S(G, K(\alpha))\|_{\mathcal{H}_\infty} \leq \gamma, \text{ for all } \alpha \in Q,$$

if and only if there exist $P(\alpha) > 0$ and $R(\alpha) \in \mathbb{R}^{m\times n}$ such that $K(\alpha) = R(\alpha)P^{-1}(\alpha)$ and

$$\begin{bmatrix} \begin{bmatrix} A(\alpha) B_2(\alpha) \end{bmatrix} \begin{bmatrix} P(\alpha) \\ R(\alpha) \end{bmatrix} + \begin{bmatrix} P(\alpha) R^T(\alpha) \end{bmatrix} \begin{bmatrix} A^T(\alpha) \\ B_2^T(\alpha) \end{bmatrix} & \star & \star \\ B_1^T(\alpha) & -\gamma I & \star \\ \begin{bmatrix} C_1(\alpha) \; D_{12}(\alpha) \end{bmatrix} \begin{bmatrix} P(\alpha) \\ R(\alpha) \end{bmatrix} & D_{11}(\alpha) & -\gamma I \end{bmatrix} < 0, \tag{8.11}$$

for all $\alpha \in Q$, where $\gamma > 0$ and $S(G, K(\alpha))$ is the map from the external input ω to the external output z of the closed loop system with a static full state feedback controller. The symbol $\star$ denotes the symmetric blocks in the matrix inequality. To find a solution to the robust H_∞-optimal static state-feedback controller problem with optimal feedback gain $K(\alpha) = P(\alpha)R^{-1}(\alpha)$, one can solve the following optimization of polynomials problem.

$$
\begin{aligned}
&\gamma^* = \min_{P,R\in\mathbb{R}[\alpha],\gamma\in\mathbb{R}} \quad \gamma \\
&\text{subject to} \\
&\begin{bmatrix} -P(\alpha) & \star & \star & \star \\ 0 & \begin{bmatrix} A(\alpha) B_2(\alpha) \end{bmatrix} \begin{bmatrix} P(\alpha) \\ R(\alpha) \end{bmatrix} + \begin{bmatrix} P(\alpha) R^T(\alpha) \end{bmatrix} \begin{bmatrix} A^T(\alpha) \\ B_2^T(\alpha) \end{bmatrix} & \star & \star \\ 0 & B_1^T(\alpha) & 0 & \star \\ 0 & \begin{bmatrix} C_1(\alpha) \; D_{12}(\alpha) \end{bmatrix} \begin{bmatrix} P(\alpha) \\ R(\alpha) \end{bmatrix} & D_{11}(\alpha) & 0 \end{bmatrix} - \gamma \begin{bmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,I\,0 \\ 0\,0\,0\,I \end{bmatrix} < 0 \\
&\qquad \text{for all } \alpha \in Q.
\end{aligned}
\tag{8.12}
$$

In Problem (8.12), if $Q = \Delta^l$ as defined in (2.15), then we can apply Polya's theorem as described in Section 2.3.1 to find a $\gamma \leq \gamma^*$ and P and R which satisfy the inequality in (8.12). Suppose $P, A, B_1, B_2, C_1, D_{11}$ and D_{12} are homogeneous polynomials (otherwise use the procedure in Section 4.2 to homogenize them). Let

$$
F(P(\alpha), R(\alpha)) := \begin{bmatrix} -P(\alpha) & \star & \star & \star \\ 0 & \begin{bmatrix} A(\alpha) B_2(\alpha) \end{bmatrix} \begin{bmatrix} P(\alpha) \\ R(\alpha) \end{bmatrix} + \begin{bmatrix} P(\alpha) R^T(\alpha) \end{bmatrix} \begin{bmatrix} A^T(\alpha) \\ B_2^T(\alpha) \end{bmatrix} & \star & \star \\ 0 & B_1^T(\alpha) & 0 & \star \\ 0 & \begin{bmatrix} C_1(\alpha) \; D_{12}(\alpha) \end{bmatrix} \begin{bmatrix} P(\alpha) \\ R(\alpha) \end{bmatrix} & D_{11}(\alpha) & 0 \end{bmatrix},
$$

and denote the degree of F by d_f. Given $\gamma \in \mathbb{R}$, the inequality in (8.12) holds if there exist $e \geq 0$ such that all of the coefficients of the polynomial

$$\left(\sum_{i=1}^{l} \alpha_i\right)^e \left(F(P(\alpha), R(\alpha)) - \gamma \begin{bmatrix} 0\,0\,0\,0 \\ 0\,0\,0\,0 \\ 0\,0\,I\,0 \\ 0\,0\,0\,I \end{bmatrix} \left(\sum_{i=1}^{l} \alpha_i\right)^{d_f} \right) \tag{8.13}$$

are negative-definite. Let P and R be of the forms

$$P(\alpha) = \sum_{h \in W_{d_p}} P_h \alpha_1^{h_1} \cdots \alpha_l^{h_l}, P_h \in \mathbb{S}^n \tag{8.14}$$

and

$$R(\alpha) = \sum_{h \in W_{d_r}} R_h \alpha_1^{h_1} \cdots \alpha_l^{h_l}, R_h \in \mathbb{R}^{n \times n}, \tag{8.15}$$

where W_{d_p} and W_{d_r} are the exponent sets defined in (4.2). By combining (8.14) and (8.15) with (8.13) it follows from Polya's theorem that for a given γ, the inequality in (8.12) holds, if there exist $e \geq 0$ such that

$$\sum_{h \in W_{d_p}} \left(M_{h,q}^T P_h + P_h M_{h,q}\right) + \sum_{h \in W_{d_r}} \left(N_{h,q}^T R_h^T + R_h N_{h,q}\right) < 0 \quad \text{for all } q \in W_{d_f+e}, \tag{8.16}$$

where we define $M_{h,q} \in \mathbb{R}^{n \times n}$ as the coefficient of $P_h \alpha_1^{q_1} \cdots \alpha_l^{q_l}$ after substituting (8.14) and (8.15) into (8.13). Likewise, $N_{h,q} \in \mathbb{R}^{n \times n}$ is the coefficient of $R_h \alpha_1^{q_1} \cdots \alpha_l^{q_l}$ after substituting (8.14) and (8.15) into (8.13). For given $\gamma > 0$, if there exist $e \geq 0$ such that LMI (8.16) has a solution, say $P_h, h \in W_{d_p}$ and $R_g, g \in W_{d_r}$, then

$$K(\alpha) = \left(\sum_{h \in W_{d_p}} P_h \alpha_1^{h_1} \cdots \alpha_l^{h_l}\right) \left(\sum_{g \in W_{d_r}} R_g \alpha_1^{g_1} \cdots \alpha_l^{g_l}\right)^{-1}$$

is a feedback law of an H_∞-suboptimal static state-feedback controller for System (8.10). By performing bisection search on γ and solving (8.16) for each γ of the bisection, one may find an H_∞-optimal controller for System (8.10).

8.2.3 Parallel computation of value functions for approximate dynamic programming

Consider the discrete-time optimal control problem

$$
\begin{aligned}
J^* := \min_{u_k \in U} \quad & \sum_{k=0}^{\infty} \beta^k g(x_k, u_k) \\
\text{subject to} \quad & x_{k+1} = f(x_k, u_k) && \text{for } k = 1, 2, 3, \cdots \\
& x_0 = z, \; x_k \in X && \text{for } k = 1, 2, 3, \cdots,
\end{aligned} \tag{8.17}
$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ are given polynomials, $\beta \in (0, 1]$ is a discount factor, $U \subset \mathbb{R}^m, X \subset \mathbb{R}^n$, and $z \in \mathbb{R}^n$ is a given initial condition for the dynamical system. It is well-known that dynamic programming approach (Bertsekas *et al.* (1995)) provides sufficient conditions for existence of a solution to the optimal control problem in (8.17). The key idea underlying dynamic programming is that optimization over-time can often be considered as optimization in stages. In such framework, optimal control is any decision which minimizes the sum of: 1. cost of transition from current stage k to the next stage $k + 1$; and 2. cost of all stages subsequent to $k + 1$, incurred by the decision made at stage k. This is referred to as the principle of optimality and was first formulated by Bellman (Bellman and Kalaba (1965)) as

$$J^* = V^*(z) = (\mathcal{P}V^*)(z) := \inf_{v \in U} \{g(z, v) + \beta\, V^*(f(z, v))\} \quad \text{for all } z \in X. \tag{8.18}$$

The unique solution to Bellman's equation is called the value function - can be thought of as the minimum cost-to-go from the current state. Existence of the value functions is a sufficient condition for existence of an optimal control. In fact, an optimal policy $\mu^* : X \to U$ can be expressed in terms of the value function V^*:

$$\mu^*(z) = \arg\min_{u \in U} \{g(z, u) + \beta\, V^*(f(z, u))\}$$

for any $x_0 \in X$. Thus, Bellman's equation solves the optimal control problem by providing a closed-loop feedback law for *every* initial condition.

It is shown that the Bellman's operator $\mathcal{P}$ defined in (8.18) possesses the following two properties:

1. Iteratively applying of Bellman's operator $\mathcal{P}$ on any function $h : X \to \mathbb{R}$ results in a pointwise convergence to a value function, i.e.,

$$V^*(x) = \lim_{k\to\infty} (\mathcal{P}^k h)(x) \quad \text{for all } \ x \in X. \tag{8.19}$$

2. Bellman's operator is *monotonic*: If V satisfies the Bellman's inequality $V(x) \leq (\mathcal{P}V)(x)$ for all $x \in X$, then $V(x) \leq (\mathcal{P}^k V)(x)$ for all $x \in X$ and for any $k \geq 1$.

From these two properties one can conclude that

$$V \leq \mathcal{P}^k V \ \text{ for some } k \geq 1 \ \Rightarrow \ V \leq V^*.$$

Unfortunately, for $k > 1$, the constraint $V \leq \mathcal{P}^k V$ is non-convex in the coefficients of polynomial V. A sufficient condition for $V \leq \mathcal{P}^k V$ is to search for polynomials V and W_i, $i = 1, \cdots, k$ such that

$$V \leq \mathcal{P} W_1, \ W_1 \leq \mathcal{P} W_2, \ \cdots, W_{k-1} \leq \mathcal{P} V.$$

Note that all of these constraints are convex in the coefficients of V and W_i. Let V and W_i be polynomials of forms

$$V(x) = \sum_{\alpha \in I(d_V)} V_\alpha x^\alpha \text{ and } W_i(x) = \sum_{\alpha \in I(d_{W_i})} W_{i,\alpha} x^\alpha,$$

where $I(d) := \{\alpha \in \mathbb{N}^n : \|\alpha\|_1 \leq d\}$. Then, any polynomial V which solves the convex optimization problem

$$
\begin{aligned}
J_k := \max_{V_\alpha, W_{i,\alpha}} \quad & \sum_{\alpha \in I(d_V)} V_\alpha z^\alpha \\
\text{subject to} \quad & \sum_{\alpha \in I(d_V)} V_\alpha x^\alpha \leq \mathcal{P} \sum_{\alpha \in I(d_{W_1})} W_{1,\alpha} x^\alpha && \text{for all } x \in X \\
& \sum_{\alpha \in I(d_{W_i})} W_{i,\alpha} x^\alpha \leq \mathcal{P} \sum_{\alpha \in I(d_{W_{i+1}})} W_{i+1,\alpha} x^\alpha && \text{for all } x \in X \text{ and } i = 1, \cdots, k-2 \\
& \sum_{\alpha \in I(d_{W_{k-1}})} W_{k-1,\alpha} x^\alpha \leq \mathcal{P} \sum_{\alpha \in I(d_V)} V_\alpha x^\alpha && \text{for all } x \in X
\end{aligned} \tag{8.20}
$$

for any initial condition $z \in X$ and some $k \geq 1$, is an under-estimator for the value function V^*. Moreover, from monotonicity of $\mathcal{P}$ it follows that

$$J_1 \leq J_2 \leq \cdots \leq J_k \leq \cdots \leq V^*(z).$$

In other words, by increasing k, the lower bound J_k defined in (8.20) can only improve or remain constant. By substituting for $\mathcal{P}$ in (8.20) from (8.18), removing the infimum and enforcing the constraints of Problem 8.20 for all control inputs $u \in U$, we get the following optimization of polynomials problem.

$$
\begin{aligned}
\max_{V_\alpha, W_{i,\alpha}} \quad & \sum_{\alpha \in I(d_V)} V_\alpha z^\alpha \\
\text{subject to} \quad & \sum_{\alpha \in I(d_V)} V_\alpha x^\alpha \leq g(x,u) + \beta \sum_{\alpha \in I(d_{W_1})} (W_{1,\alpha} f(x,u)^\alpha) && \text{for } x \in X, u \in U \\
& \sum_{\alpha \in I(d_{W_i})} W_{i,\alpha} x^\alpha \leq g(x,u) + \beta \sum_{\alpha \in I(d_{W_{i+1}})} (W_{i+1,\alpha} f(x,u)^\alpha) && \text{for } x \in X, u \in U, i = 1, \cdots, k-2 \\
& \sum_{\alpha \in I(d_{W_{k-1}})} W_{k-1,\alpha} x^\alpha \leq g(x,u) + \beta \sum_{\alpha \in I(d_V)} (V_\alpha f(x,u)^\alpha) && \text{for } x \in X, u \in U,
\end{aligned} \tag{8.21}
$$

where for brevity, we have denoted $f_1(x,u)^{\alpha_1} \cdots f_n(x,u)^{\alpha_n}$ by $f(x,u)^\alpha$.

Problem (8.21) has some interesting computational properties. Since all of the constraints in this problem have the same structure, if we choose the same degree for

V and W_i, it is then sufficient to set-up only one of the constraints in order to set-up the entire Problem (8.21). If X and U are simplicies, Algorithm 5 can be used to perform Polya's iterations on the constraints of Problem (8.21). The result is a linear program whose solution yields an under-estimator for the value function V^* defined in (8.18). Likewise, if X and U are hypercubes (or polytopes), then Algorithm 7 (or Algorithm 2) can be used to perform Polya's iterations (or Handelman's iterations) on the constraints of Problem (8.21). Another interesting property of Problem (8.21) is that increasing the accuracy of the under-estimations (by increasing k) amounts to a *linear* growth in the number of decision variables and number of constraints.

REFERENCES

"Appendix schedule B - summary of marginal costs", available at: `http://www.srpnet.com/prices/priceprocess/pdfx/Unbundled.pdf` (2014).

"Standard electric price plans", Salt River Project Agricultural Improvement and Power District Corporate Pricing (November 2015).

Ackermann, J., A. Bartlett, D. Kaesbauer, W. Sienel and R. Steinhauser, *Robust Control: Systems with Uncertain Physical Parameters* (Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001).

Adams, W. and P. Loustaunau, *An Introduction to Groebner Bases* (American Mathematical Society, 1994).

Alizadeh, F., J. Haeberly and M. Overton, "Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results", SIAM Journal of Optimization **8**, 3, 746–768 (1998).

Amdahl, G. M., "Validity of the single processor approach to achieving large-scale computing capabilities", No. 30, pp. 483–485 (AFIPS Conference Proceedings, 1967).

Arguello-Serrano, B. and M. Velez-Reyes, "Nonlinear control of a heating, ventilating, and air conditioning system with thermal load estimation", IEEE Transactions on Control Systems Technology **7**, 1, 56–63 (1999).

Arizona Public Service, "2014 integrated resource plan: Executive summary", (2014).

Artin, E., "Uber die zerlegung definiter funktionen in quadra", Quadrate, Abh. Math. Sem. Univ. Hamburg **5**, 85–99 (1927).

Barmish, B. R. and C. L. DeMarco, "A new method for improvement of robustness bounds for linear state equations", (in Proceedings Conf. Inform. Sci. Syst. Princeton University, 1986).

Bellman, R. and R. E. Kalaba, *Dynamic programming and modern control theory* (Academic Press New York, 1965).

Bellman, R. E. and S. E. Dreyfus, *Applied Dynamic Programming* (Princeton University Press, 1962).

Ben-Tal, A. and A. Nemirovski, "Robust convex optimization", Math. Operat. Res. **23**, 4, 769–805 (1998).

Benson, S., Y. Ye and X. Zhang, "Solving large-scale sparse semidefinite programs for combinatorial optimization", SIAM Journal on Optimization **10**, 2, 443–461 (2000).

Benson, S. J., “DSDP3: Dual scaling algorithm for general positive semidefinite programs”, Preprint ANL/MCS-P851-1000, Argonne National Labs (2001).

Benson, S. J., Y. Ye and X. Zhang, “Solving large-scale sparse semidefinite programs for combinatorial optimization”, SIAM Journal on Optimization **10**, 443–461 (1998).

Bernstein, S., “Sur la repr sentation des polynomes positif”, Soobshch. Har'k. Mat. Obshch. **2**, 14, 227–228 (1915).

Bertsekas, D. P., D. P. Bertsekas, D. P. Bertsekas and D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1 (Athena Scientific Belmont, MA, 1995).

Bhattacharyya, S. P., H. Chapellat and L. H. Keel, *Robust Control: The Parametric Approach* (Prentice Hall, 1995).

Bliman, P.-A., “A convex approach to robust stability for linear systems with uncertain scalar parameters”, SIAM Journal on Control and Optimization **42**, 6, 2016–2042 (2004a).

Bliman, P. A., “An existence result for polynomial solutions of parameter dependent LMIs”, Systems & Control Letters , 3-4, 165–169 (2004b).

Bliman, P.-A., R. Oliveira, V. Montagner and P. Peres, “Existence of homogeneous polynomial solutions for parameter-dependent linear matrix inequalities with parameters in the simplex”, in “IEEE Conference on Decision and Control”, pp. 1486–1491 (2006).

Blondel, V. and J. Tsitsiklis, “A survey of computational complexity results in systems and control”, Automatica **36**, 9, 1249–1274 (2000).

Blondel, V. D. and J. N. Tsitsiklis, “Complexity of stability and controllability of elementary hybrid systems”, Automatica **35**, 3, 479–489 (1999).

Borchers, B. and J. G. Young, “Implementation of a primal dual method for SDP on a shared memory parallel architecture”, Computational Optimization and Applications **37**, 3, 355–369 (2007).

Boudaoud, F., F. Caruso and M. Roy, “Certificates of positivity in the bernstein basis”, Discrete and Computational Geometry **39**, 4, 639–655 (2008).

Boukas, E. K., “Static output feedback control for stochastic hybrid systems: LMI approach”, Automatica **42**, 1, 183–188 (2006).

Boyd, S. and L. Vandenberghe, *Convex optimization* (Cambridge university press, 2004).

Braun, J. E., “Reducing energy costs and peak electrical demand through optimal control of building thermal storage”, ASHRAE transactions **96**, 2, 876–888 (1990).

Braun, J. E., "Load control using building thermal mass", Journal of solar energy engineering **125**, 3, 292–301 (2003).

Braun, J. E., T. Lawrence, C. Klaassen and J. House, "Demonstration of load shifting and peak load reduction with control of building thermal mass", Teaming for Efficiency: Commercial buildings: technologies, design, performance analysis, and building industry trends **3**, 55 (2002).

Braun, J. E. and K. H. Lee, "Assessment of demand limiting using building thermal mass in small commercial buildings", Transactions on American Society of Heating, Refrigerating and Air-Conditioning Engineers **112**, 1, 547–558 (2006).

Braun, J. E., K. W. Montgomery and N. Chaturvedi, "Evaluating the performance of building thermal mass control strategies", HVAC&R Research **7**, 4, 403–428 (2001).

Briat, C., "Robust stability and stabilization of uncertain linear positive systems via integral linear constraints: L1-gain and L2-gain characterization", International Journal of Robust and Nonlinear Control **23**, 17, 1932–1954 (2013).

Brown, C., "QEPCAD B: a program for computing with semi-algebraic sets using CADs", ACM SIGSAM Bulletin **37**, 4, 97–108 (2003).

Castle, M., V. Powers and B. Reznick, "Polya's theorem with zeros", Journal of Symbolic Computation **46**, 9, 1039–1048 (2011).

Chang, Y. and B. Wah, "Polynomial programming using Groebner bases", IEEE Computer Software and Applications Conference pp. 236–241 (1994).

Chen, T. Y., "Real-time predictive supervisory operation of building thermal systems with thermal mass", Journal of Energy and Buildings **33**, 2, 141–150 (2001).

Chesi, G., "Establishing stability and instability of matrix hypercubes", System and Control Letters **54**, 381–388 (2005).

Chesi, G., A. Garulli, A. Tesi and A. Vicino, "Polynomially parameter-dependent lyapunov functions for robust stability of polytopic systems: an lmi approach", IEEE Transactions on Automatic Control **50**, 3, 365–370 (2005).

Collins, G. and H. Hoon, "Partial cylindrical algebraic decomposition for quantifier elimination", Journal of Symbolic Computation **12**, 3, 299–328 (1991).

de Loera, J. and F. Santos, "An effective version of polya's theorem on positive definite forms", Journal of Pure and Applied Algebra **108**, 3, 231–240 (1996).

Deitz, S., *High-level programming language abstractions for advanced and dynamic parallel computations*, Ph.D. thesis, Computer Science and Engineering Department, University of Washington (2005).

Delzell, C., "Impossibility of extending polya's theorem to forms with arbitrary real exponents", Journal of Pure and Applied Algebra **212**, 12, 2612–2622 (2008).

Dennis Jr, J. E. and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, vol. 16 (SIAM, 1996).

Dickinson, P. and J. Pohv, "On an extension of polya's positivstellensatz", Journal of Global Optimization pp. 1–11 (2014).

Dolzmann, A. and T. Sturm, "Redlog: Computer algebra meets computer logic", ACM SIGSAM Bulletin **31**, 2, 2–9 (1997).

Dullerud, G. and F. Paganini, *A Course in Robust Control Theory, A Convex Approach* (Springer-Verlag New York, 2000).

EPRI-DOE, "Handbook of energy storage for transmission and distribution applications", 1001834, EPRI, Palo Alto, CA, and the U.S. Department of Energy, Washington, DC (2003).

F. Alizadeh, M. L. O., J. P. A. Haeberly, "Primal-dual interior-point methods for semidefinite programming", Math Programming Symposium (Ann Arbor 1994).

Farin, G. E., *Curves and surfaces for CAGD: a practical guide* (Morgan Kaufmann, 2002).

G. Blekherman, P. P. and R. Thomas, *Semidefinite optimization and convex algebraic geometry* (MOS-SIAM Series on Optimization, Philadelphia, 2013).

G. Chesi, A. T., A. Garulli and A. Vicino, "Lmi-based computation of optimal quadratic lyapunov functions for odd polynomial systems", International Journal of Robust and Nonlinear Control **15**, 1, 35–49 (2005).

G. Hardy, J. L. and G. Polya, *Inequalities* (Cambridge University Press, 1934).

Gahinet, P., P. Apkarian and M. Chilali, "Affine parameter-dependent lyapunov functions and real parametric uncertainty", IEEE Transactions on Automatic Control **41**, 3, 436–442 (1996).

Gatermann, K. and P. Parrilo, "Symmetry groups, semidefinite programs, and sums of squares", Journal of Pure and Applied Algebra **192**, 1, 95–128 (2004).

Geromel, J. and M. de Oliveira, "H_2 and H_∞; robust filtering for convex bounded uncertain systems", IEEE Transactions on Automatic Control **46**, 1, 100–107 (2001).

Green, M. and D. J. N. Limebeer, *Linear robust control* (Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995).

Greenlaw, R., H. Hoover and W. Ruzzo, *Limits to parallel computation: P-completeness theory* (Oxford University Press, USA, 1995).

Gripenberg, G., "Computing the joint spectral radius", Linear Algebra and its Applications **234**, 43–60 (1996).

Gugercin, S. and A. Antoulas, "A survey of model reduction by balanced truncation and some new results", International Journal of Control **77**, 8, 748–766 (2004).

Habicht, W., “Uber die zerlegung strikte definiter formen in quadrate”, Commentarii Mathematici Helvetici **12**, 1, 317–322 (1939).

Handelman, D., “Representing polynomials by positive linear functions on compact convex polyhedra”, Pacific Journal of Mathematics **132**, 1, 35–62 (1988a).

Handelman, D., “Representing polynomials by positive linear functions on compact convex polyhedra”, Pac. J. Math **132**, 1, 35–62 (1988b).

Hardy, G., J. E. Littlewood and G. Pólya, *Inequalities* (Cambridge University Press, 1934).

Helmberg, C. and F. Rendl, “A spectral bundle method for semidefinite programming”, SIAM Journal on Optimization **10**, 3, 673–696 (2000).

Helmberg, C., F. Rendl, R. J. Vanderbei and H. Wolkowicz, “An interior-point method for semidefinite programming”, SIAM Journal of Optimization **6**, 2, 342–361 (1996).

Helmberg, C., F. Rendl, R. J. Vanderbei and H. Wolkowicz, “An interior-point method for semidefinite programming”, Princeton University. Princeton, NJ, USA (2005).

Henze, G., C. Felsmann and G. Knabe, “Evaluation of optimal control for active and passive building thermal storage”, International Journal of Thermal Sciences **43**, 2, 173–183 (2004).

Hilbert, D., “Uber die darstellung definiter formen als summe von formen quadratens”, Math. Ann. **32**, 342–350 (1888).

Hilbert, D., “Uber ternare definite formen”, Acta Math. **17**, 169–197 (1893).

Jeyakumar, V., J.-B. Lasserre and G. Li, “On polynomial optimization over non-compact semi-algebraic sets”, Journal of Optimization Theory and Applications **163**, 3, 707–718 (2014).

Kalé, L., B. Ramkumar, A. Sinha and A. Gursoy, “The CHARM parallel programming language and system: Part I–description of language features”, Parallel Programming Laboratory Technical Report No. 95-02 (1994).

Kamyar, R. and M. Peet, “Decentralized computation for robust stability analysis of large state-space systems using polya’s theorem”, American Control Conference (2012a).

Kamyar, R. and M. Peet, “Decentralized computation for robust stability of large-scale systems with parameters on the hypercube”, IEEE 51st Conference on Decision and Control pp. 6259–6264 (2012b).

Kamyar, R. and M. Peet, “Decentralized polya’s algorithm for stability analysis of large-scale nonlinear systems”, IEEE Conference on Decision and Control pp. 5858–5863 (2013).

Kamyar, R., M. Peet and Y. Peet, “Solving large-scale robust stability problems by exploiting the parallel structure of polya’s theorem”, IEEE Transactions on Automatic Control **58**, 8, 1931–1947 (2013).

Keeney, K. and J. E. Braun, “Application of building precooling to reduce peak cooling requirements”, ASHRAE transactions **103**, 1, 463–469 (1997).

Kim, S., M. Kojima and H. Waki, “Generalized lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems”, SIAM Journal on Optimization **15**, 3, 697–719 (2005).

Kintner-Meyer, M. and A. F. Emery, “Optimal control of an HVAC system using cold storage and building thermal capacitance”, Energy and Buildings **23**, 1, 19–31 (1995).

Kuhn, H. W., A. W. Tucker and J. Neyman, “Nonlinear programming”, Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability pp. 481–492 (1951).

L. Blum, M. S., F. Cucker and S. Smale, *Complexity and real computation* (Springer-Verlag, New York, 1998).

Lasserre, J. B., “Global optimization with polynomials and the problem of moments”, SIAM Journal on Optimization **11**, 3, 796–817 (2001).

Laurent, M., “Sums of squares, moment matrices and optimization over polynomials”, Springer: The IMA Volumes in Mathematics and its Applications **149**, 157–270 (2009).

Lavaei, J. and A. G. Aghdam, “Robust stability of LTI systems over semialgebraic sets using sum-of-squares matrix polynomials”, IEEE Transactions on Automatic Control **53**, 1, 417–423 (2008).

Leroy, R., “Convergence under subdivision and complexity of polynomial minimization in the simplicial bernstein basis”, Reliable Computing **17**, 11–21 (2012).

Li, H. and M. Fu, “A linear matrix inequality approach to robust H_∞ filtering”, IEEE Transactions on Signal Processing, **45**, 9, 2338–2350 (1997).

Li, N., L. Chen and S. Low, “Optimal demand response based on utility maximization in power networks”, Proceedings of IEEE Power and Energy Society General Meeting pp. 1–8 (2011).

Lombardi, H., “Effective real nullstellensatz and variants”, in “Effective Methods in Algebraic Geometry”, pp. 263–288 (Springer, 1991).

Lu, L., W. Cai, Y. S. Chai and L. Xie, “Global optimization for overall HVAC systems - Part II problem solution and simulations”, Energy Conversion and Management **46**, 7, 1015–1028 (2005).

Ma, W.-J., V. Gupta and U. Topcu, "On distributed charging control of electric vehicle with power network capacity constraints", in "American Control Conference, Portland", (2014).

Monteiro, R., "Primal-dual path-following algorithms for semidefinite programming", SIAM Journal of Optimization **7**, 3, 663–678 (1997).

Motzkin, T., "The arithmetic-geometric inequality", Symposium on Inequalities, Academic Press pp. 205–224 (1967).

Nayakkankuppam, M., "Solving large-scale semidefinite programs in parallel", Mathematical programming **109**, 2, 477–504 (2007).

Nemirovskii, A., "Several NP-hard problems arising in robust stability analysis", Mathematics of Control, Signals and Systems **6**, 2, 99–105 (1993).

Oldewurtel, F. and M. Morari, "Reducing peak electricity demand in building climate control using real-time pricing and model predictive control", IEEE Conf. on Decision and Control pp. 1927–1932 (2010).

Oliveira, R., P. Bliman and P. Peres, "Robust LMIs with parameters in multi-simplex: Existence of solutions and applications", IEEE 47th Conference on Decision and Control pp. 2226–2231 (2008).

Oliveira, R. C. and P. L. Peres, "Parameter-dependent LMIs in robust analysis: characterization of homogeneous polynomially parameter-dependent solutions via LMI relaxations", IEEE Transactions on Automatic Control **52**, 7, 1334–1340 (2007).

Oliveira, R. C. L. F. and P. L. D. Peres, "A less conservative LMI condition for the robust stability of discrete-time uncertain systems", Syst. Control Lett. **43**, 4, 371–378 (2001).

Oliveira, R. C. L. F. and P. L. D. Peres, "Stability of polytopes of matrices via affine parameter-dependent Lyapunov functions: Asymptotically exact LMI conditions", Linear Algebra Appl. **405**, 3, 209–228 (2005).

Olsson, D. M. and L. S. Nelson, "The Nelder-Mead simplex procedure for function minimization", Technometrics **17**, 45–51 (1975).

P. Gahinet, P. A., "A linear matrix inequality approach to h infinity control", International Journal of Robust and Nonlinear Control **4**, 4, 421–448 (1994).

Packard, A. and J. Doyle, "Quadratic stability with real and complex perturbations", IEEE Transactions on Automatic Control **35**, 2, 198–201 (1990).

Papachristodoulou, A., J. Anderson, G. Valmorbida, S. Prajna, P. Seiler and P. A. Parrilo, "SOSTOOLS: Sum of squares optimization toolbox for MATLAB", preprint, arXiv:1310.4716 (2013).

Papachristodoulou, A., M. M. Peet and S. Lall, "Analysis of polynomial systems with time delays via the sum of squares decomposition", IEEE Transactions on Automatic Control **54**, 5, 1058–1064 (2009).

Papchristodoulou, A. and S. Prajna, “Robust stability analysis of nonlinear hybrid systems”, IEEE Transactions on Automatic Control **54**, 5, 1034–1041 (2009).

Parrilo, P., “Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization”, Ph.D thesis, California Institute of Technology (2000).

Parrilo, P., “Exploiting algebraic structure in sum of squares programs”, Positive polynomials in control pp. 580–580 (2005).

Parrilo, P. A., “Polynomial games and sum of squares optimization”, in “Decision and Control, 2006 45th IEEE Conference on”, pp. 2855–2860 (IEEE, 2006).

Patterson, M. A. and A. V. Rao, “GPOPS- II: A MATLAB software for solving multiple-phase optimal control problems”, ACM Transactions on Mathematical Software **39**, 3, 1–41 (2013).

Peaucelle, D. and D. Arzelier, “Robust performance analysis with lmi-based methods for real parametric uncertainty via parameter-dependent lyapunov functions”, IEEE Transactions on Automatic Control **46**, 4, 624–630 (2001).

Peet, M., “Exponentially stable nonlinear systems have polynomial lyapunov functions on bounded regions”, Automatic Control, IEEE Transactions on **54**, 5, 979–987 (2009).

Peet, M. M. and Y. V. Peet, “A parallel-computing solution for optimization of polynomials”, (Proceedings of the American Control Conference, 2010).

Permenter, F. and P. A. Parrilo, “Selecting a monomial basis for sums of squares programming over a quotient ring.”, in “CDC”, pp. 1871–1876 (2012).

Powers, V. and B. Reznick, “Polynomials that are positive on an interval”, Transactions of the American Mathematical Society **352**, 10, 4677–4692 (2000).

Powers, V. and B. Reznick, “A new bound for polya’s theorem with applications to polynomials positive on polyhedra”, Journal of Pure and Applied Algebra **164**, 221–229 (2001).

Powers, V. and B. Reznick, “A quantitative polya’s theorem with corner zeros”, ACM International Symposium on Symbolic and Algebraic Computation (2006).

Prajna, S. and A. Papachristodoulou, “Analysis of switched and hybrid systems-beyond piecewise quadratic methods”, in “Proceedings of the 2003 American Control Conference”, vol. 4, pp. 2779–2784 (2003).

Prestel, A. and C. Delzell, *Positive polynomials: from Hilberts 17th problem to real algebra* (Springer New York, 2004).

Putinar, M., “Positive polynomials on compact semi-algebraic sets”, Indiana University Mathematics Journal **42**, 969–984 (1993).

Ramos, D. and P. Peres, “An LMI approach to compute robust stability domains for uncertain linear systems”, (Proceedings of the American Control Conference, 2001).

Ramshaw, L., *A connect-the-dots approach to splines* (Digital Systems Research Center, 1987).

Repin, I. M., “Quadratic liapunov functionals for systems with delay”, Journal of Applied Mathematics and Mechanics **29**, 3, 669–672 (1965).

Reznick, B., “Some concrete aspects of hilbert’s 17th problem”, Contemporary Mathematics **253**, 251–272 (2000).

Reznick, B., “On the absence of uniform denominators in hilberts 17th problem”, Proceedings of the American Mathematical Society **133**, 10, 2829–2834 (2005).

Rumolo, D., “APS rate schedule ECT-2 residential service”, available at: `http://votesolar.org/wp-content/uploads/2013/07/ECT-2.pdf` (2013).

S. Sankaranarayanan, X. C. and E. Abrahm, “Lyapunov function synthesis using handelman representations”, The 9th IFAC Symposium on Nonlinear Control Systems (2013).

Sankaranarayanan, S., X. Chen and E. Abrahám, “Lyapunov function synthesis using handelman representations”, in “The 9th IFAC Symposium on Nonlinear Control Systems”, pp. 576–581 (2013).

Sassi, M. A. B., R. Testylier, T. Dang and A. Girard, “Reachability analysis of polynomial systems using linear programming relaxations”, pp. 137–151 (2012).

Sassi, M. B. and A. Girard, “Computation of polytopic invariants for polynomial dynamical systems using linear programming”, Automatica **48**, 12, 3114–3121 (2012).

Scheiderer, C., “Sums of squares on real algebraic surfaces”, Springer: manuscripta mathematica **119**, 4, 395–410 (2006).

Scheiderer, C., “Positivity and sums of squares: a guide to recent results”, Emerging applications of algebraic geometry, Springer New York pp. 271–324 (2009).

Scherer, C. W., “LMI relaxations in robust control”, European Journal of Control **12**, 1, 3–29 (2006).

Scherer, C. W. and C. W. J. Hol, “Matrix sum-of squares relaxations for robust semi-definite programs”, Math. programming Ser. B **107**, 1-2, 189–211 (2006).

Schmudgen, M., “The k-moment problem for compact semi-algebraic sets”, Mathematische Annalen **289**, 203–206 (1991).

Schweighofer, M., “Certificates for nonnegativity of polynomials with zeros on compact semialgebraic sets”, manuscripta mathematica **117**, 4, 407–428 (2005).

Shear, T., “Today in energy: Peak-to-average electricity demand ratio rising in New England and many other U.S. regions”, US Energy Information Administration (EIA), Independent statistics and Analysis (2014).

Sherali, H. and W. Adams, “A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems”, SIAM Journal on Discrete Mathematics **3**, 3, 411–430 (1990).

Sherali, H. and L. Liberti, *Reformulation-linearization technique for global optimization* (Encyclopedia of Optimization, Springer, USA, 2009).

Sherali, H. and C. Tuncbilek, “A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique”, Journal of Global Optimization **2**, 101–112 (1992).

Sherali, H. and C. Tuncbilek, “New reformulation- linearization technique based relaxations for univariate and multivariate polynomial programming problems”, Operations Research Letters **21**, 1, 1–10 (1997).

Sivaramakrishnan, K. K., “A parallel interior point decomposition algorithm for block angular semidefinite programs”, Comput. Optim. Appl. **46**, 1, 1–29 (2010).

Slater, M., “Lagrange multipliers revisited”, Springer: Traces and Emergence of Nonlinear Programming pp. 293–306 (2014).

Stengle, G., “A Nullstellensatz and a Positivstellensatz in semialgebraic geometry”, Mathematische Annalen **207**, 2, 87–97 (1974).

Sturm, J., “Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones”, Optimization methods and software **11**, 1-4, 625–653 (1999).

Sun, Y., S. Wang, F. Xiao and D. Gao, “Peak load shifting control using different cold thermal energy storage facilities in commercial buildings: a review”, Energy Conversion and Management **71**, 101–114 (2013).

Tan, W. and A. Packard, “Stability region analysis using polynomial and composite polynomial lyapunov functions and sum-of-squares programming”, IEEE Transactions on Automatic Control **53**, 2, 565–571 (2008).

Tarski, A., *A Decision Method for Elementary Algebra and Geometry* (Random Corporation monograph, Berekley and Los Angeles, 1951).

Topcu, U., A. K. Packard, P. Seiler and G. J. Balas, “Robust region-of-attraction estimation”, IEEE Transactions on Automatic Control **55**, 1, 137–142 (2010).

Tutuncu, R., K. Toh and M. Todd, “Solving semidefinite-quadratic-linear programs using SDPT3, mathematical programming”, Mathematical Programming Series B **95**, 189–217 (2003).

Waki, H., S. Kim, M. Kojima, M. Muramatsu and H. Sugimoto, “Algorithm 883: SparsePOP—a sparse semidefinite programming relaxation of polynomial optimization problems”, ACM Trans. Math. Softw. **35**, 2 (2008).

Walker, D. and J. Dongarra, “MPI: a standard message passing interface”, Supercomputer **12**, 56–68 (1996).

Wang, T.-C., S. Lall and M. West, “Polynomial level-set methods for nonlinear dynamical systems analysis”, in “Allerton conference on communication, control and computing”, pp. 640–649 (2005).

Wang, T.-C., S. Lall and M. West, “Polynomial level-set method for polynomial system reachable set estimation”, Automatic Control, IEEE Transactions on **58**, 10, 2508–2521 (2013).

Witrant, E., E. Joffrin, S. Brémont, G. Giruzzi, D. Mazon, O. Barana and P. Moreau, “A control-oriented model of the current profile in tokamak plasma”, Plasma Physics and Controlled Fusion **49**, 1075–1105 (2007).

Yamashita, M., K. Fujisawa and M. Kojima, “SDPARA: Semidefinite programming algorithm parallel version”, Parallel Computing **29**, 1053–1067 (2003).

Yamashita, M., K. Fujisawa and K. Nakata, “A high-performance software package for semidefinite programs: SDPA 7”, Technical report B-460, Dep. of Mathematical and Computing Sciences, Tokyo Institute of Technology (2010).

Zhang, X. and P. Tsiotras, “Parameter-dependent lyapunov functions for stability analysis of LTI parameter dependent systems”, pp. 5168–5173 (in Proceedings of the IEEE 42nd Conference on Decision and Control, 2003).

Zhang, X., P. Tsiotras and P. A. Bliman, “Multi-parameter dependent lyapunov functions for the stability analysis of parameter-dependent LTI systems”, pp. 1263–1268 (in Proceedings of IEEE International Symposium on, Mediterrean Conference on Control and Automation, 2005).

Zhou, K. and J. Doyle, *Essentials of Robust Control* (Prentice Hall, 1998).

Zhou, K., J. C. Doyle, K. Glover *et al.*, *Robust and optimal control*, vol. 40 (Prentice hall New Jersey, 1996).