A Chordal Sparsity Approach to Scalable Linear and Nonlinear Systems Analysis

Richard Mason
Lincoln College
University of Oxford

Thesis submitted for the degree of
Doctor of Philosophy
Trinity 2015

Acknowledgements

Firstly, thank you to Prof. Elspeth Garman and Prof. David Gavaghan for giving me the opportunity to be a student at the Life Sciences Interface DTC. I would also like to express my gratitude towards the Engineering Science department and the EPSRC for providing the environment and funding that made my PhD research possible.

Next I would like to thank Prof. Basil Kouvaritakis and his Hungarian algorithm for assigning me to be Antonis' student half a decade ago! I simply could not have found a better supervisor than Antonis. Over the years he has been endlessly generous and supportive, always making time to sit down and talk with me amongst his busy schedule. I will really miss those discussions, and I want to thank him for all of the help and advice that he has given me over the years.

I also feel very lucky to have met some great friends and colleagues during my PhD. In particular: James, Lorenz, Federico, Joe, Danny and Keary, I hope we will be life-long friends. I would also like to thank Tom, Xuan, James, Ed, Giorgio, Dhruva, Andreas and Reza for making Antonis' group such a fun and interesting place to work.

Most of all, I wish to thank my family for their love and support, especially my parents Lesley and Stephen and my sister Joanna, I dedicate this thesis to you.

Abstract

In this thesis we investigate how the properties of chordal graphs can be used to exploit sparsity in several optimisation problems that arise in control theory. In particular, we focus on analysis and synthesis problems that involve semidefinite constraints and can be formulated as semidefinite programming (SDP) problems. Using a relationship between chordal graphs and sparse semidefinite matrices, we decompose the semidefinite constraints in the associated SDP problems into multiple, smaller semidefinite constraints along with some additional equality constraints. The benefit of this approach is that for sparse dynamical systems we can solve significantly larger analysis and synthesis problems than is possible using traditional dense methods.

We begin by considering the properties of chordal graphs and their connection to sparse positive semidefinite matrices. We then turn our attention to the problem of constructing Lyapunov functions for linear time-invariant (LTI) systems. From this starting point, we derive methods of exploiting chordal sparsity in other analysis problems found in control theory. In particular, this approach is applied to the problem of bounding the input-output properties of systems via the KYP lemma for both continuous and discrete-time systems.

We then consider how the properties of chordal graphs can be exploited in the SDPs that arise in static state feedback controller synthesis problems for LTI systems. We show that the sparse inverse property of the maximum determinant completion of a partial positive matrix can be used to design controllers with a pre-specified sparsity pattern. We then consider how to exploit chordal sparsity when designing a static state feedback controller to minimise the H_∞ norm of an LTI system.

Next we shift from linear systems to nonlinear systems and develop a chordal sparsity approach to scalable stability analysis of systems with polynomial dynamics using the Sums of Squares (SOS) technique. We develop a method of exploiting chordal sparsity that avoids the computationally costly step of forming the coefficient matrix in the SOS problem. We then apply this method to the problem of constructing Lyapunov functions for systems with correlatively sparse polynomial vector fields. Finally, we conclude by discussing some directions for future research.

Contents

Chapter 1

Introduction

Many analysis and synthesis problems in control theory can be formulated as convex optimisation problems and solved in polynomial time. This means that in principle we can use iterative methods to solve these analysis and synthesis problems efficiently. However, in practice we find that in many cases constraints on computational resources remain a limiting factor on the scale of systems that can be analysed and designed using these convex optimisation techniques.

To address with this computational challenge we may consider methods of exploiting sparsity within optimisation problems to enable them to be solved more efficiently. In particular, this thesis is about how to use the relationship between chordal graphs and positive semidefinite matrices to more efficiently solve some of the Semidefinite Programs (SDPs) that arise in control theory. We apply this approach, which we call the chordal sparsity approach, to three problems in control theory: constructing Lyapunov functions for LTI systems, controller synthesis for LTI systems and nonlinear stability analysis for systems with polynomial vector

fields using SOS. To help introduce these concepts we next briefly discuss notion of sparsity via graphs.

Graphs can be used to provide an abstract representation of the connections between objects. For example, the network of computers that makes up the internet and the interconnection of neurons in our brain can both be represented using a graph. One of the benefits of graphical models is that they allow us to model the notion of sparsity; a feature that is common to many important systems that are encountered in the world.

In order to define what we mean by a sparse graph, let us first consider a dense graph. A dense graph is a graph consisting of n nodes where every node is connected to itself and every other node in the graph to give the maximum possible n^2 (directed) edges. By contrast to a dense graph, we say that a graph consisting of n nodes is sparse if it has significantly less edges than the maximum possible n^2.

One of the reasons why it is important to consider sparsity is the consequences it has on our ability to solve problems via algorithms. In the simplest case, a sparse problem will likely require less memory to be stored in a computer. On a deeper level, sparsity also leads to the classification of graphs according to their sparsity patterns, e.g., the class of tree graphs, cycle graphs, bipartite graphs. Some problems that are computationally difficult to solve when posed on general graphs become tractable for graphs in these special classes.

For example, consider the inference problem of computing a marginal for a prob-

ability distribution that factorises according to a graph. For general graphs the computation time required to marginalise over all of the other variables in the probability distribution grows exponentially with the size of the graph. However, for tree graphs the same problem can be solved in linear time using a message passing algorithm that performs computations locally, i.e., by exchanging information between nodes that are neighbours in the graph, see Appendix A.1 for further details [1].

In this thesis, we will focus primarily on the class of chordal graphs, which are the class of undirected graphs where every cycle of length greater than or equal to four has a chord, i.e., an edge connecting two nonconsecutive nodes in the cycle [2]. Several important problems that are computationally difficult to solve on general graphs can be solved in polynomial time for chordal graphs. For example, consider the classical problem in graph theory of finding the maximum clique of a given graph. For general graphs this is known to be an NP-hard problem, but for chordal graphs the maximum clique problem can be solved in a time that grows linearly in the number of nodes and edges in the graph [3].

Chordal graphs have been found to have applications in a number of fields. For example, in numerical algebra they are used to facilitate the solution of sparse linear systems via sparse Cholesky factorisations [4]. In inference and machine learning they have been applied to maximum likelihood estimation for sparse graphical models [5], as well as to generalise message passing algorithms to graphs with cycles [6]. In this thesis, our focus will be on the applications of chordal graphs to problems in control theory, and in particular, on exploiting the relationship be-

tween chordal graphs and sparse SDPs.

SDPs are conic optimisation problems, meaning that they involve the optimisation of a convex function over the intersection of an affine subspace and a convex cone [7]. They can be thought of as a generalisation of Linear Programs (LP), where the cone of vectors in the positive orthant has been generalised to the cone of positive semidefinite matrices [8, 9]. SDPs arise in several fundamental analysis and synthesis problems in control theory [10, 11, 12, 13]. For example, we may formulate the problem of constructing a Lyapunov function for a linear system as an SDP [14]. Beyond control theory, SDPs have found an array of important applications to problems in combinatorial optimisation [15], polynomial optimisation [16, 17], sensor network localisation [18, 19], manifold embedding [20], optimal power flow problems [21] and approximating the intersection of ellipsoids [22].

The interest in SDPs was triggered by the development of interior-point methods for solving SDPs [23, 24, 25]. Interior-point methods enabled larger scale SDPs to be solved than had previously been practical using the ellipsoid algorithm [26] and turned SDPs into a more practically interesting modelling framework. These theoretical developments were accompanied by the release of several open-source software packages that implemented interior-point methods, such as SeDuMi [27], SDPA [28], SDPT3 [29], DSDP [30] and CSDP [31]. These open-source software packages enabled researchers to quickly apply SDPs to their individual domains. More recently, the alternating direction method of multipliers (ADMM) [32] has also been applied to solving SDPs, and a number of SDP solvers that implement ADMM have been developed, such as SDPAD [33] and SDPNAL [34].

As interest in SDPs and their applications grew, researchers naturally became interested in solving increasingly large instances of SDPs. This has led to an exploration of the special structures within SDPs that can be exploited to improve the efficiency with which they are solved. Three main types of special structure in SDPs have emerged [35]: chordal sparsity in the data matrices [36, 37], low rank data matrices [38], and algebraic symmetry where the data matrices belong to a low dimensional matrix algebra [39, 40]. In this thesis we will focus exclusively on chordal sparsity.

The theory behind the chordal sparsity approach originates from two papers by Grone *et. al* [41] and Agler *et. al* [42]. In these papers, the authors proved two important results that relate chordal graphs to sparse positive semidefinite matrices. Grone's and Agler's results apply to the dual cones of partial positive and sparse positive semidefinite matrices respectively, and essentially reduce the problem of checking whether a sparse matrix is positive semidefinite (or can be completed to be positive semidefinite) to checking whether special submatrices of the matrix are positive semidefinite.

Based on Grone's and Agler's results, Fukuda *et. al.* showed that Grone's theorem could be applied to decompose the semidefinite constraints found on the primal side of an SDP [36]. By 'decompose' we mean that the semidefinite constraint of the SDP is converted into multiple, smaller semidefinite constraints, at the cost of introducing extra equality constraints/free variables. As the duality between Grone's and Agler's theorems became better known, this led to the application

of Agler's theorem by Kim *et. al.* to decomposing the semidefinite constraints found on the dual side of the SDP [43]. The impact of this line of research is that when the data matrices in an SDP are sparse this decomposition process can be applied to significantly improve the rate at which the semidefinite programs are solved numerically [44].

We now focus on one particular SDP from control theory to demonstrate some of the computational challenges that can arise. Consider the stability analysis problem for an LTI system $\dot{x} = Ax$ with $x \in \mathbb{R}^{n \times n}$ and $A \in \mathbb{R}^{n \times n}$ (we will return to this problem in Chapter 3). A classical result in control theory is that A is Hurwitz if and only if there exists a Lyapunov function for the system of the form $V(x) = x^T P x$ with $P \succeq \epsilon I$ and $\epsilon > 0$, where $A \succeq B$ means that the matrix $A - B$ is positive semidefinite. It is well known that we can compute P by picking a negative definite matrix Q and solving the Lyapunov equation $Q = A^T P + PA$, but for the sake of exposition, let us suppose that we wish to tackle this problem by solving an SDP. We may formulate the problem as finding a $P = P^T$ that satisfies the following semidefinite constraints

$$\begin{bmatrix} P & 0 \\ 0 & -(A^T P + PA) \end{bmatrix} \succeq \epsilon I, \quad \epsilon > 0. \tag{1.1}$$

Then using a set of basis matrices $E_1, E_2, \ldots, E_m \in \mathbb{R}^{n \times n}$, where each matrix E_i corresponds to one of the $m = n(n+1)/2$ free variables $p_{11}, p_{12}, \ldots, p_{nn}$ in P, we can reformulate (1.1) as the following SDP feasibility problem

$$p_{11}F_1 + p_{12}F_2 + \cdots + p_{nn}F_m - \epsilon I \succeq 0,$$

where

$$F_i = \begin{bmatrix} E_i & 0 \\ 0 & -(A^T E_i + E_i A) \end{bmatrix}, \ i = 1, 2, \ldots, m.$$

In principle, we may solve this SDP using a standard interior-point method. However, we note that the number of free variables in this SDP grows quadratically with n and the multiplication of A with a dense P matrix results in a dense matrix variable. These characteristics mean that standard interior-point methods are restricted to solving relatively small instances of this problem in practice.

Rather than solving (1.1) using a dense P, we could instead restrict P to have some sparsity pattern, as was proposed in [45]. For example, we may restrict P to be a diagonal matrix, and under this assumption the SDP would involve n free variables, and $A^T P + PA$ would have the same sparsity pattern as $A^T + A$. The trade-off for this reduction in the number of free variables and increase in sparsity in the SDP is that we lose the guarantee that a Lyapunov function exists except in some special cases [46, 47].

The observation that we can influence the sparsity pattern of the SDP is useful in cases where the dimension of the system makes it impossible to solve the SDP for dense matrix variables - a situation that is becoming increasingly common as larger systems are analysed. By restricting P to be a sparse matrix we open up the possibility of exploiting chordal sparsity. However, the challenge then becomes to structure P so as to make the most of the sparsity pattern of A. We next give an outline of the thesis.

1.1 Outline of the Thesis and Contributions

- In Chapter 2 we first discuss chordal graphs and a number of their properties. We state Grone's and Agler's theorems, which relate chordal graphs to sparse positive semidefinite matrices, and explain how these results can be applied to SDPs to exploit sparsity, with particular emphasis on the duality relationships.

- In Chapter 3 we apply Grone's and Agler's theorems to the analysis of sparse linear systems in continuous-time and discrete-time. We first give examples of systems with special structures that can be exploited, and then generalise this to more general sparse systems via an iterative algorithm that searches for increasingly dense Lyapunov functions. We then apply this approach to computing bounds on the H_∞ norm of a system via the KYP Lemma and compare it to the bounds computed using the standard dense method. The work in this chapter formed the basis of a paper published at the American Control Conference in 2014 [48].

- In Chapter 4 we turn our attention to exploiting the properties of chordal graphs when designing static state feedback controllers. In the first half of the chapter, we combine the maximum determinant completion property of chordal graphs with concepts from robust control theory to design sparse state feedback controllers. In the second half of the chapter, we consider how to exploit chordal sparsity within the problem of constructing state feedback

controllers for large, sparse systems using diagonal Lyapunov functions.

- In Chapter 5 we consider the problem of verifying local stability of systems with polynomial dynamics. We develop a method of decomposing the associated SDP problem via the concept of correlative sparsity and Agler's theorem. We then provide some numerical results for large, sparse systems to demonstrate the effectiveness of the approach.

- In Chapter 6 we provide some concluding remarks and consider potential directions for future research.

Chapter 2

Preliminaries

In this chapter, we cover the mathematical preliminaries on chordal graphs and SDPs that provide the theoretical foundation of the thesis. The material that is specific to control theory will be introduced separately at the start of each subsequent chapter. In the first section of this chapter, we consider some of the characterisations of chordal graphs and the algorithms related to them. In the second section, we discuss the relationship between chordal graphs and positive semidefinite matrices and build towards the theorems by Grone *et. al.* and Agler *et. al.*. This leads to the third section where these theorems are applied to the decomposition of sparse SDPs. We show that the duality between Grone's and Agler's theorems is mirrored by the duality between the primal and dual problems of an SDP.

2.1 Chordal Graphs

Let $G = (V, E)$ be a connected, simple, undirected graph i.e., a connected graph without self-loops or multiple edges, with set of nodes (or vertices) $V = \{1, 2, \ldots, n\}$ and set of edges $E \subseteq V \times V$. Two vertices $u, v \in V$ are said to be neighbours or adjacent if $(u, v) \in E$. The set of vertices that are neighbours of $v \in V$ is denoted by $N(v) = \{u \in V \mid (u, v) \in E\}$. For a subset $S \subset V$, the induced subgraph of $G = (V, E)$ is the graph $G[S] = (S, F)$ with vertex set S and edge set $F = E \cap (S \times S)$. A clique is a subset $S \subset V$ such that for any $i, j \in S$, $(i, j) \in E$. A maximal clique is a clique that is not a subset of another clique. A cycle is a sequence of pairwise distinct vertices $\gamma = (v_1, v_2, \ldots, v_s)$ having the property that

$$(v_1, v_2), (v_2, v_3), \ldots, (v_{s-1}, v_s), (v_s, v_1) \in E,$$

and s is called the length of the cycle. A chord of a cycle γ is an edge $(v_i, v_j) \in E$ where $1 \leq i < j \leq s$, $(i, j) \neq (1, s)$, and $|i - j| \geq 2$ [41].

Definition 2.1.1 A graph G is chordal if every cycle of length ≥ 4 has a chord.

Chordal graphs are sometimes also called triangulated or rigid circuit graphs in the literature. See Figure 2.1 for some examples of chordal graphs. Chordal graphs have a number of appealing properties from a computational point of view. For example, they can be recognised in polynomial time [2], and the vertex colouring problem, which is NP-complete for general graphs, can be solved in polynomial time for chordal graphs [49].

Definition 2.1.2 For a given graph $G = (V, E)$, we say that a graph $G_{ch} = (V, F)$ is a chordal extension of G if G_{ch} is chordal and $E \subseteq F$.

A given graph may have many possible chordal extensions and frequently in applications it is desirable to find chordal extensions that add the minimum number of edges to the graph in order to extend it to be chordal. However, the problem of finding a chordal extension that adds the minimum number of edges is known to be NP-complete [50]. Fortunately, effective heuristics for finding minimal chordal extensions have been developed, such as permuting the adjacency matrix of the graph using a minimum-degree ordering (using for example MATLAB's `symamd` function) and then performing a symbolic Cholesky factorisation [36].

In the next three subsections we give some further definitions related to chordal graphs and state four theorems that are important for characterising them. For the proofs of these theorems we refer the reader to the review by Blair and Peyton [51]. Several of the characterisations that we discuss involve running algorithms on the graph and we include pseudo-code for these algorithms.

2.1.1 Perfect Elimination Orderings

A vertex v is called a simplicial vertex if all its neighbours are adjacent to each other. A bijection $\alpha : V \mapsto \{1, \ldots, n\}$ is called an ordering of G and can be written as $\alpha = \langle v_1, v_2, \ldots, v_n \rangle$ where $v_i = \alpha^{-1}(i)$. An ordering α of G is a perfect elimination ordering if, for each $i \in \{1, \ldots, n\}$, v_i is a simplicial vertex of the subgraph induced by $S = \{v_i, \ldots, v_n\}$ [4].

Theorem 2.1.1 [52] A graph G is chordal if and only if G has a perfect elimination ordering.

In 1976 Rose, Lueker and Tarjan proposed an algorithm to generate an elimination

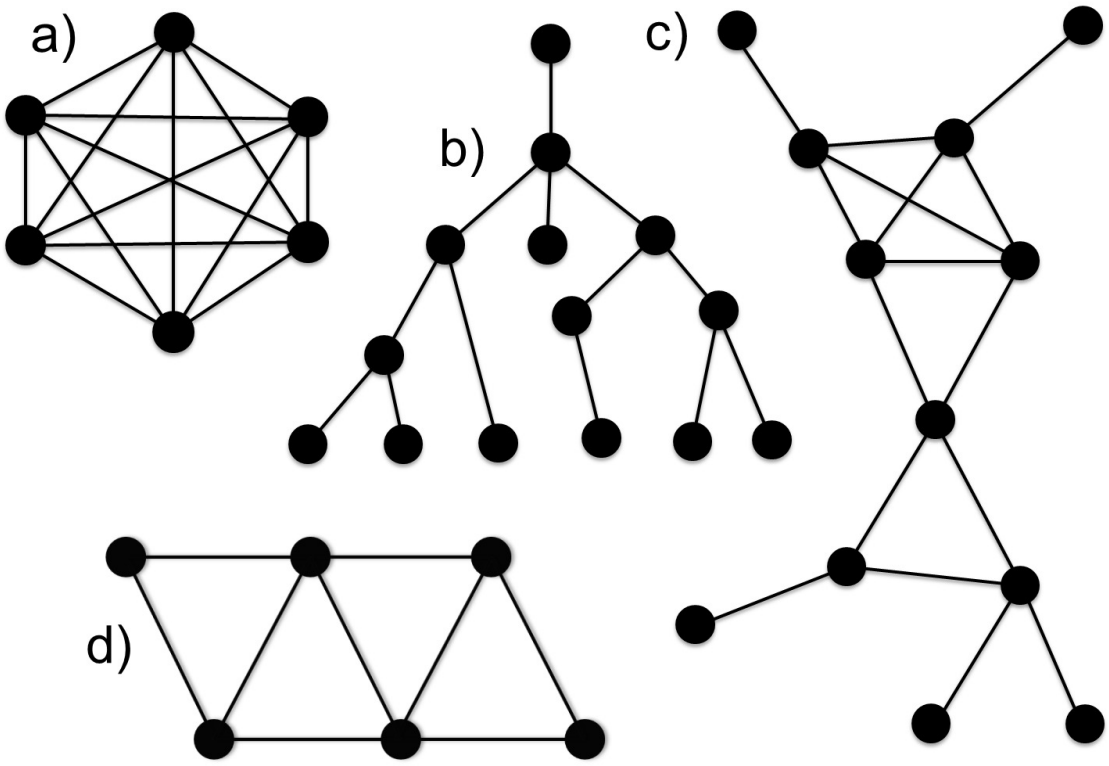


Figure 2.1: Some examples of chordal graphs.

ordering called Lexicographical Breadth-first Search [3]. Subsequently, in 1984, Tarjan and Yannakakis introduced a conceptually simpler algorithm called the Maximum Cardinality Search algorithm that runs in $O(|V| + |E|)$ time [53]. The concept of the algorithm is to label each node in the graph with a number that defines the node's place in the elimination ordering. It works by starting with an arbitrary node in the graph and initialising all the nodes to have a 'weight' of zero. This weight keeps a running total that is updated at each iteration by incrementing the weights of all the nodes that are adjacent to the node that was previously labeled.

Maximum Cardinality Search Algorithm [53]**:** Given a graph $G = (V, E)$, output an ordering α of V.

Let $n = |V|$ be the number of nodes in the graph.

Let $w = \mathbf{0}$ be the n-dimensional vector of node weights.

for $i = 1 : n$ **do**

  pick an unnumbered node v with maximum weight; set $\alpha(v) = i$

  for all unnumbered node u adjacent to v **do**

    $w(u) = w(u) + 1$

  end

end

$\alpha \leftarrow$ reverse the ordering of α

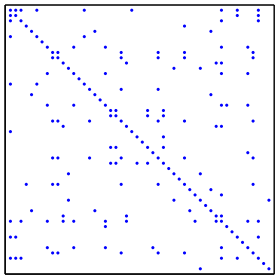
To check whether a given graph is chordal we can simply check whether the ordering returned by the Maximum Cardinality Search algorithm is a perfect elimination

ordering. This also requires $O(|V|+|E|)$ time since it requires us to check for each node $i = 1, 2, \ldots, n-1$ in the ordering whether it is a simplicial node in the subgraph $G(S_i)$ defined by the set of nodes $S_i = \{i, i+1, \ldots, n\}$.

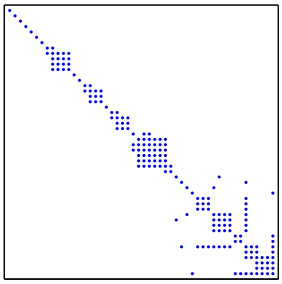
The characterisation of chordal graphs in terms of perfect elimination orderings has an important application in numerical algebra. Consider the problem of solving $Ax = b$ when A is positive definite and sparse. The standard method for solving this numerically is to first compute the Cholesky factorisation $A = LL^T$, where L is a lower triangular matrix, and then solve for x using a forward substitution followed by a backward substitution. In general, the matrix $L + L^T$ will be denser than A, but if the graph describing the sparsity pattern of A is chordal there exists a permutation matrix P such that L has the same sparsity pattern as the lower triangular part of A. More specifically, suppose $G(A)$ is a chordal graph with perfect elimination ordering α and define the matrix P as

$$P_{ij} = \begin{cases} 1 & \text{if } j = \alpha(i) \\ 0 & \text{otherwise.} \end{cases}$$

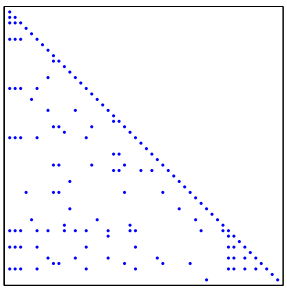
then the Cholesky factorisation of $P^T AP$ has the same sparsity pattern as A. Making use of this property helps to minimise the memory required to solve such systems, which is crucial when the system is of large dimension [54]. See Figure 2.2 for an example.



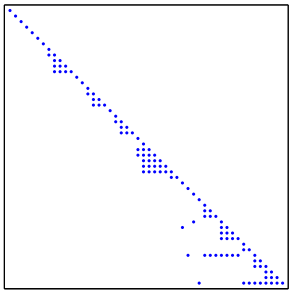
(a) A (nz= 184)



(b) $P^T AP$ (nz=184)



(c) Cholesky factor of A (nz=144)



(d) Cholesky factor of $P^T AP$ (nz=117)

Figure 2.2: Sparsity pattern of a 50×50 chordal matrix A and its Cholesky factor before and after applying a perfect elimination ordering permutation matrix P. Note that the sparsity pattern of $P^T AP$ in (b) is preserved in its Cholesky factorisation (d). The variable nz is the number of nonzero entries in each matrix.

2.1.2 Maximal Cliques

For general graphs, the problem of listing all of the maximal cliques is a computationally challenging problem because the number of maximal cliques can grow exponentially in the number of nodes. Chordal graphs on the other hand can have at most n maximal cliques [49]. Given a perfect elimination ordering, the maximal cliques of a chordal graph can be found in $O(|V| + |E|)$ time using the Maximal Clique Algorithm [52].

Maximal Clique Algorithm [52]**:** Given a chordal graph $G = (V, E)$ consisting of n nodes and a perfect elimination ordering $\alpha = \langle v_1, v_2, \ldots, v_n \rangle$ output the maximal cliques.

Initialise: $C_0 = \emptyset$

for $i = 1 : n$ **do**

$C_i = \{\nu_i\} \cup \{\, u \in N(\nu_i) \mid \alpha(u) > \alpha(\nu_i) \,\}$

if C_i is not a subset of C_0 **do**

C_i is a maximal clique

C_0=C_i

end

end

For example, consider the chordal graph shown in Figure 2.3. A perfect elimination ordering of this graph is given by $\alpha = \langle 2, 3, 1, 5, 4, 6, 7, 8 \rangle$. If we run the Maximal Clique Algorithm on this graph with this perfect elimination ordering the sets C_i

for $i = 1, 2, \ldots, 8$ are given by

$$
\begin{aligned}
C_1 &= \{2, 3, 1, 4\} \quad \leftarrow \text{maximal clique} \\
C_2 &= \{3, 1, 4\} \\
C_3 &= \{1, 4, 8\} \quad \leftarrow \text{maximal clique} \\
C_4 &= \{5, 4, 6\} \quad \leftarrow \text{maximal clique} \\
C_5 &= \{4, 6, 8\} \quad \leftarrow \text{maximal clique} \\
C_6 &= \{6, 7, 8\} \quad \leftarrow \text{maximal clique} \\
C_7 &= \{7, 8\} \\
C_8 &= \{8\}.
\end{aligned}
$$

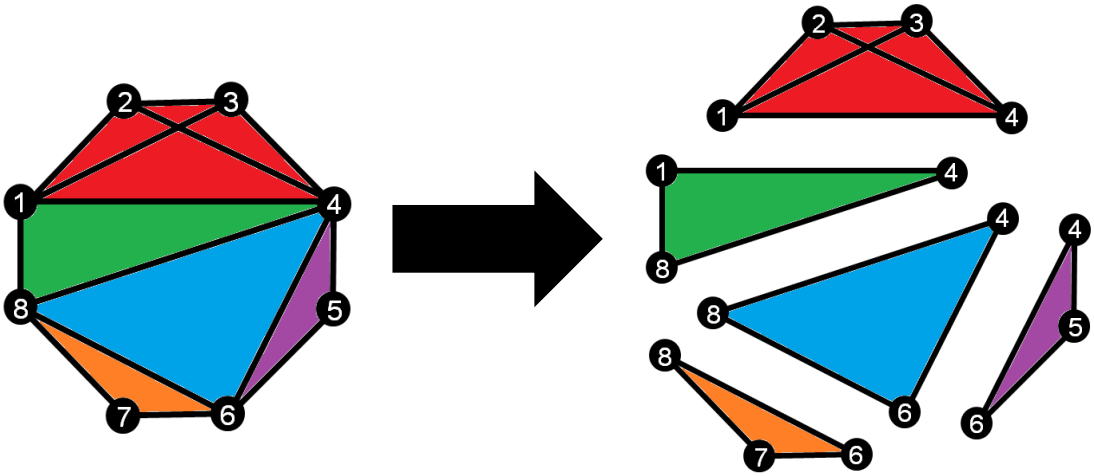


Figure 2.3: Decomposing a chordal graph into its maximal cliques.

2.1.3 Clique Trees

Let G be a connected graph with set of maximal cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$. A compact representation of G can be formed by treating the maximal cliques of

the graph as nodes in a connected, acyclic graph called a clique tree, $T = (\mathcal{C}, \mathcal{E})$ where $\mathcal{E} \subseteq \mathcal{C} \times \mathcal{C}$ is an arbitrary edge set. A clique tree is said to satisfy the clique-intersection property if for every pair of distinct $C_i, C_j \in \mathcal{C}$, the set $C_i \cap C_j$ is contained in every clique on the unique path connecting C_i and C_j in the clique tree, see Figure 2.4 for an example. This brings us to the second characterisation of chordal graphs.

Theorem 2.1.2 [51] A connected graph G is chordal if and only if there exists a clique tree $T = (\mathcal{C}, \mathcal{E})$ for which the clique-intersection property holds.

Clique trees that satisfy the clique-intersection property have important applications to probabilistic inference problems as part of the junction tree algorithm. For this application, the clique-intersection property is vital to the functioning of the algorithm as it ensures consistency between the marginals of the probability distribution [6]. Figure 2.4 shows an example of a chordal graph that satisfies the clique-intersection property, as shown by the accompanying clique tree.

We next consider how to compute a clique tree that satisfies the clique-intersection property for a given chordal graph. For an arbitrary graph G with set of maximal cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$, we define the weighted clique graph $W(G)$ to be the weighted graph with set of nodes $\mathcal{C}$ and edge weights $w_{ij} = |C_i \cap C_j|$. Let $\mathcal{T}$ be the (nonempty) set of maximum-weight spanning trees of $W(G)$ i.e., the set of spanning trees on the nodes of $W(G)$ for which the sum of the edge weights is maximised.

Theorem 2.1.3 [51] A graph G is chordal if and only if every $T \in \mathcal{T}$ is a clique tree that satisfies the clique-intersection property.

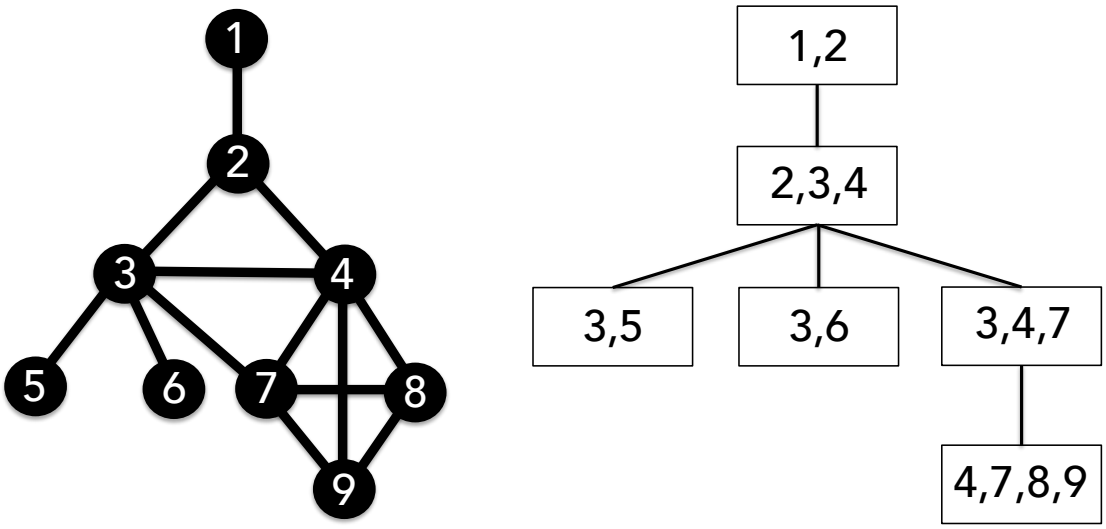


Figure 2.4: A chordal graph and one possible clique tree that satisfies the clique-intersection property. As an example of the clique-intersection property, consider the two maximal cliques $\{2, 3, 4\}$ and $\{4, 7, 8, 9\}$. The intersection of these sets is $\{4\}$, which is a subset of the maximal clique $\{3, 4, 7\}$ that lies on the unique path between them on the clique tree.

Given a chordal graph G, Theorem 2.1.2 guarantees that there exists a clique tree that satisfies the clique-intersection property, and the problem of computing such a clique tree reduces to computing a single maximum-weight spanning tree of its weighted clique graph $W(G)$, since every $T \in \mathcal{T}$ satisfies the clique-intersection property by Theorem 2.1.3. This process can be carried out using Prim's algorithm, which is stated below [55].

Prim's Algorithm [55]**:** Given a connected weighted graph $G = (V, E)$

output $G' = (V', E')$ that is a maximum-weight spanning tree for G.

Initialise: $V' = x$, for some arbitrary $x \in V$, $E' = \emptyset$

repeat until $V' = V$:

find an edge (u, v) with maximum weight such that $u \in V'$ and $v \notin V'$

$V' \leftarrow (V' \cup v)$ and $E' \leftarrow (E' \cup (u, v))$

end

One final characterisation of chordal graphs will be necessary for the work in later chapters. This characterisation is based on what is known as the running intersection property. An ordering of the maximal cliques of a graph, say $C_1, C_2, \ldots, C_p$ satisfies the running intersection property if for every $k = 1, 2, \ldots, p-1$,

$$\left(C_{k+1} \cap \bigcup_{j=1}^{k} C_j \right) \subseteq C_s \text{ for some } s \leq k.$$

Theorem 2.1.4 [51] A connected graph G is chordal if and only if there exists an ordering of the maximal cliques for which the running intersection property holds.

Given a clique tree for a chordal graph that satisfies the clique-intersection property we can compute an ordering that satisfies the running intersection property by finding a topological ordering for the nodes in the clique tree. A topological ordering is simply a numbering of the nodes in any rooted tree such that each parent node has a lower number than its children. For example, for the clique tree in Figure 2.4 we may start with the root of the tree and number the maximal

cliques as $C_1 = \{1, 2\}, C_2 = \{2, 3, 4\}, C_3 = \{3, 5\}, C_4 = \{3, 6\}, C_5 = \{3, 4, 7\}, C_6 = \{4, 7, 8, 9\}$.

2.2 Chordal Graphs and Positive Semidefinite Matrices

In this section we proceed by discussing a connection between chordal graphs and positive semidefinite matrices. We first introduce some notation and then state the aforementioned Grone's and Agler's theorems. These theorems treat the sparsity pattern of a square symmetric matrix as representing a graph where an edge (i, j) is present if the (i, j)th element of the matrix is nonzero. When this graph is chordal these theorems allow the positive (semi)definiteness of the matrix to be concluded from the positive (semi)definiteness of the submatrices of the matrix that correspond to the maximal cliques of the graph.

For any simple graph $G = (V, E)$, let $G' = (V, E')$ where $E' = E \cup_{i=1}^{n} (i, i)$ i.e., an augmented graph where each node has a loop. Throughout this section the theorems involve both graphs and matrices with sparsity patterns given by graphs. To keep the notation simple, our convention for the rest of the thesis will be that when we refer to an edge set E in the context of a graph it is the edge set of the simple graph $G = (V, E)$, but when we refer to the edge set E in the context of a matrix we are technically referring to E', the augmented edge set of the graph $G' = (V, E')$.

Given a graph $G = (V, E)$, a partial symmetric matrix, X, is a symmetric $n \times n$ matrix whose element $X_{ij} = X_{ji}$ is specified if and only if $(i, j) \in E$. A completion of X is an $n \times n$ matrix M which satisfies $M_{ij} = X_{ij}$ for all $(i, j) \in E$. We say

that M is a positive completion of X if and only if M is a completion of X and M is positive semidefinite. We will use the following notation:

$$\begin{aligned}
&\mathbb{S}^n(E,?) = \text{the set of } n \times n \text{ partial symmetric matrices with} \\
&\qquad\qquad \text{elements defined on } E \\
&\mathbb{S}^n_+(E,?) = \{X \in \mathbb{S}^n(E,?) |\ \exists M \succeq 0, M_{ij} = X_{ij}\ \forall (i,j) \in E\} \\
&\mathbb{S}^n_{++}(E,?) = \{X \in \mathbb{S}^n(E,?) |\ \exists M \succ 0, M_{ij} = X_{ij}\ \forall (i,j) \in E\} \\
&\mathbb{S}^n(E,0) = \{X \in \mathbb{S}^n \mid X_{ij} = 0 \text{ if } (i,j) \notin E\} \\
&\mathbb{S}^n_+(E,0) = \{X \in \mathbb{S}^n(E,0) \mid X \succeq 0\} \\
&\mathbb{S}^n_{++}(E,0) = \{X \in \mathbb{S}^n(E,0) \mid X \succ 0\} \\
&U_{ij} = \text{the } n \times n \text{ symmetric matrix with 1 in the } (i,j)\text{th} \\
&\qquad \text{and } (j,i)\text{th elements and 0 elsewhere.}
\end{aligned}$$

In the sequel we will often need to restrict our focus to a particular submatrix of a given matrix and so we introduce some further notation to facilitate this. Let $C \subseteq V$ be an arbitrary subset of the nodes and define the following sets:

$$\begin{aligned}
&\mathbb{S}^n(C) = \{X \in \mathbb{S}^n \mid X_{ij} = 0 \text{ if } (i,j) \notin C \times C\} \\
&\mathbb{S}^n_+(C) = \{X \in \mathbb{S}^n(C) \mid X \succeq 0\} \\
&X(C) = \{Y \in \mathbb{S}^n(C) \mid Y_{ij} = X_{ij} \text{ for all } (i,j) \in C \times C)\} \\
&J(C) = \{(i,j) \in C \times C \mid 1 \leq i \leq j \leq n\}
\end{aligned}$$

Finally, given a clique tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ that satisfies the clique intersection property,

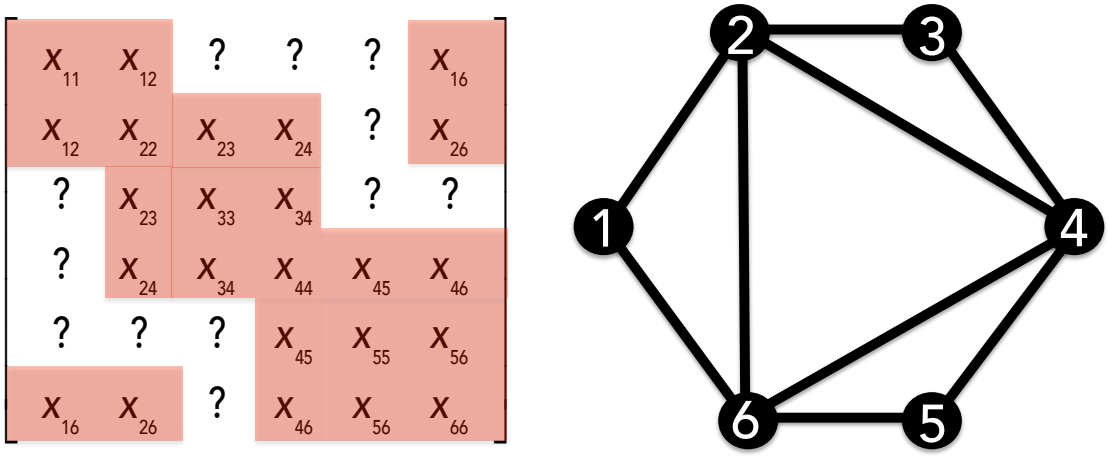


Figure 2.5: An example of a partial symmetric matrix $\mathbb{S}^n(E,?)$ and associated sparsity pattern graph $G = (V, E)$.

we denote the minimal set of overlapping elements by Λ, where

$$\Lambda = \{(i,j,k,l) \mid (i,j) \in J(C_k \cap C_l),\ (C_k, C_l) \in \mathcal{E}\}.$$

2.2.1 Grone's Theorem

We are now ready to present Grone's theorem, which states that when a partial symmetric matrix has a chordal sparsity pattern there exists a positive semidefinite completion of the matrix if and only if all of the submatrices corresponding to the maximal cliques of the underlying graph are positive semidefinite.

Theorem 2.2.1 [41] Let $G = (V, E)$ be a chordal graph with set of maximal cliques $\mathcal{C} = \{C_1, \ldots, C_p\}$. Suppose that $X \in \mathbb{S}^n(E,?)$. Then $X \in \mathbb{S}^n_+(E,?)$ if and only if $X(C_k) \succeq 0$ for $k = 1, 2, \ldots, p$.

This theorem describes a deep connection between chordal graphs and positive

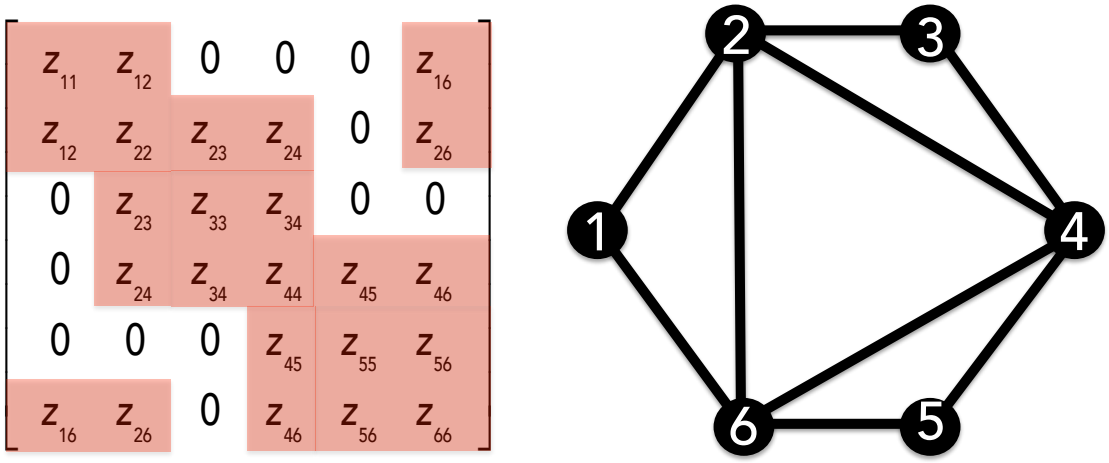


Figure 2.6: An example of a sparse symmetric matrix $\mathbb{S}^n(E, 0)$ and associated sparsity pattern graph $G = (V, E)$ with the same sparsity pattern as in Figure 2.5.

semidefinite matrices, and will be crucial later for decomposing the constraints on the primal side of the SDP when the data matrices are sparse.

Example 2.2.1 As an example of how Grone's theorem can be applied, consider the following partial symmetric matrix

$$X = \begin{bmatrix} 10 & 4 & ? & ? & ? & -4 \\ 4 & 10 & -2 & -2 & ? & -2 \\ ? & -2 & 12 & 8 & ? & ? \\ ? & -2 & 8 & 9 & -1 & 2 \\ ? & ? & ? & -1 & 7 & -4 \\ -4 & -2 & ? & 2 & -4 & 9 \end{bmatrix} \in \mathbb{S}^n(E, ?).$$

This matrix has a chordal sparsity pattern (it has the same sparsity pattern as the partial matrix in Figure 2.5), and the graph $G(X) = (V, E)$ has four maximal cliques, which are $C_1 = \{2, 4, 6\}$, $C_2 = \{1, 2, 6\}$, $C_3 = \{2, 3, 4\}$ and $C_4 = \{4, 5, 6\}$.

The matrices corresponding to these maximal cliques are

$$X(C_1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & -2 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 9 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 2 & 0 & 9 \end{bmatrix} \succeq 0, \quad X(C_2) = \begin{bmatrix} 10 & 4 & 0 & 0 & 0 & -4 \\ 4 & 10 & 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & -2 & 0 & 0 & 0 & 9 \end{bmatrix} \succeq 0,$$

$$X(C_3) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & -2 & -2 & 0 & 0 \\ 0 & -2 & 12 & 8 & 0 & 0 \\ 0 & -2 & 8 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \succeq 0, \quad X(C_4) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9 & -1 & 2 \\ 0 & 0 & 0 & -1 & 7 & -4 \\ 0 & 0 & 0 & 2 & -4 & 9 \end{bmatrix} \succeq 0.$$

Since $X(C_1), X(C_2), X(C_3), X(C_4)$ are positive semidefinite, Grone's theorem states there exists a positive semidefinite completion of X i.e., $X \in \mathbb{S}^n_+(E, ?)$. One positive semidefinite completion of X is the following matrix

$$M = \begin{bmatrix} 10 & 4 & -2 & -2 & 1 & -4 \\ 4 & 10 & -2 & -2 & 0 & -2 \\ -2 & -2 & 12 & 8 & -1 & 1 \\ -2 & -2 & 8 & 9 & -1 & 2 \\ 1 & 0 & -1 & -1 & 7 & -4 \\ -4 & 2 & 1 & 2 & -4 & 9 \end{bmatrix} \succeq 0.$$

2.2.2 Agler's Theorem

We next state Agler's theorem which has a dual relationship to Grone's theorem. Agler's theorem states that a sparse positive semidefinite matrix with a chordal sparsity pattern can be decomposed into a sum of positive semidefinite matrices that have sparsity patterns that correspond to the maximal cliques of the underlying graph.

Theorem 2.2.2 [42] Let $G = (V, E)$ be a chordal graph with set of maximal cliques $\mathcal{C} = \{C_1, \ldots, C_p\}$. Suppose that $A \in \mathbb{S}^n(E, 0)$. Then $A \in \mathbb{S}^n_+(E, 0)$ if and only if there exists a set of matrices $\{A_1, A_2 \ldots, A_p\}$ such that

$$A = \sum_{k=1}^{p} A_k, \;\; A_k \in \mathbb{S}^n_+(C_k), \; k = 1, \ldots, p.$$

Example 2.2.2 As an example of how Theorem 2.2.2 can be applied, consider the following positive semidefinite matrix

$$A = \begin{bmatrix} 6 & -1 & 0 & 0 & 0 & 3 \\ -1 & 8 & -6 & -3 & 0 & 1 \\ 0 & -6 & 15 & 5 & 0 & 0 \\ 0 & -3 & 5 & 8 & -3 & 1 \\ 0 & 0 & 0 & -3 & 6 & -1 \\ 3 & 1 & 0 & 1 & -1 & 4 \end{bmatrix} \succeq 0.$$

Since A has a chordal sparsity pattern (it has the same sparsity pattern as the matrix in Example 2.2.1) we may apply Theorem 2.2.2 to decompose this matrix

into four positive semidefinite matrices, one for each maximal clique:

$$A_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1.5 & 0 & 2 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 1 \end{bmatrix} \succeq 0, \quad A_2 = \begin{bmatrix} 6 & -1 & 0 & 0 & 0 & 3 \\ -1 & 3 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0.5 & 0 & 0 & 0 & 2 \end{bmatrix} \succeq 0,$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & -6 & -1.5 & 0 & 0 \\ 0 & -6 & 15 & 5 & 0 & 0 \\ 0 & -1.5 & 5 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \succeq 0, \quad A_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & -3 & 0.5 \\ 0 & 0 & 0 & -3 & 6 & -1 \\ 0 & 0 & 0 & 0.5 & -1 & 1 \end{bmatrix} \succeq 0,$$

and by adding these together we may verify that $A = A_1 + A_2 + A_3 + A_4$.

For our purposes we next restate Theorem 2.2.2 using clique trees, we will call this restated version of the theorem Agler's theorem.

Theorem 2.2.3 [43] Let $G = (V, E)$ be a chordal graph with set of maximal cliques $\mathcal{C} = \{C_1, \ldots, C_p\}$. Suppose that $A \in \mathbb{S}^n(E, 0)$ and let $\{A_1, A_2, \ldots, A_p\}$ be a set of matrices with $A_k \in \mathbb{S}^n(C_k)$ for $k = 1, 2, \ldots, p$ that satisfies $A = \sum_{k=1}^{p} A_k$. Then $A \in \mathbb{S}^n_+(E, 0)$ if and only if the system of LMIs

$$A_k - L_k(\theta) \in \mathbb{S}^n_+(C_k), \ k = 1, 2, \ldots, p,$$

is feasible, where θ is a vector of variables of the form $\theta = (\theta_{ijkl} \mid (i,j,k,l) \in \Lambda)$ and

$$L_k(\theta) = \sum_{(i,j,l) \mid (i,j,k,l) \in \Lambda} U_{ij}\theta_{ijkl} - \sum_{(i,j,h) \mid (i,j,h,k) \in \Lambda} \theta_{ijhk}U_{ij}$$

for every $\theta = (\theta_{ijkl} \mid (i,j,k,l) \in \Lambda),\ k \in \{1,2,\ldots,p\}$.

It can be seen that Grone's and Agler's theorems apply to the dual cones $\mathbb{S}^n_+(E,?)$ and $\mathbb{S}^n_+(E,0)$, and in fact one can be derived from the other using this dual relationship [56].

2.3 Semidefinite Programs and Chordal Sparsity

In this section, we expand on our discussion of SDPs that began in the introduction and consider how to apply Grone's and Agler's theorems to the constraints in the primal and dual SDP. The primal form of an SDP is

$$\begin{aligned} \text{minimize} \quad & A_0 \bullet X \\ \text{subject to} \quad & A_i \bullet X = b_i, \ \ i = 1,\ldots,m \\ & X \succeq 0 \end{aligned} \tag{2.1}$$

where $b \in \mathbb{R}^m$ and $A_0, A_1, \ldots, A_m \in \mathbb{S}^n$ are given data, $X \in \mathbb{S}^n$ is the free variable and $A \bullet B = \sum_{ij} A_{ij}B_{ij}$. An SDP is a convex optimisation problem and hence Lagrange duality plays an important role, allowing the optimal value of the SDP

to be lower bounded by the optimal value of a dual SDP of the form

$$\begin{aligned} \text{maximise} \quad & b^T y \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i A_i + Z = A_0 \\ & Z \succeq 0 \end{aligned} \tag{2.2}$$

where $y \in \mathbb{R}^m$ and $Z \in \mathbb{S}^n$ are called the dual variables.

The idea of using Grone's theorem to decompose the constraints in sparse SDPs was pioneered by Fukuda *et. al.* in [36]. By applying Theorem 2.2.1 to (2.1) the semidefinite constraint can be decomposed into multiple smaller semidefinite constraints, each of dimension equal to the size of the corresponding maximal cliques in the graph describing the sparsity pattern, plus a number of extra equality constraints to ensure consistency where the maximal cliques overlap.

The papers by Nakata *et. al.* [44], Waki *et. al.* [16], Lassere [57] and Andersen *et. al.* [54], continued in this vein, developing algorithms to exploit Grone's theorem on the primal side of the SDP and finding applications in polynomial optimisation. Later, as the duality between Grone's and Agler's theorem became more widely appreciated, Kim *et. al.* showed that the semidefinite constraints in the dual SDP could also be decomposed by using Agler's theorem [43]. This completed the picture of how chordal sparsity could be exploited in both the primal and dual sides of an SDP.

2.3.1 Exploiting Chordal Sparsity in the Primal SDP

Now following [43], we will describe the steps required for decomposing the constraints in the primal and dual SDP using Grone's and Agler's theorems. To describe the sparsity pattern of the data in the primal SDP problem (2.1), we introduce the aggregate sparsity pattern. Given a set of data matrices $A_1, A_2, \ldots, A_m \in \mathbb{S}^n$, the aggregate sparsity pattern is defined as the following edge set

$$E = \{\, (i,j) \mid [A_k]_{ij} \neq 0 \text{ for some } k \in \{1, 2 \ldots, m\} \,\}.$$

We treat the set E as the edge set of a graph $G = (V, E)$ with set of nodes $V = \{1, 2, \ldots, n\}$.

Suppose that $A \in \mathbb{S}^n(E, 0)$ is some matrix with the same sparsity pattern as the aggregate sparsity pattern defined above, and let $G_{ch} = (V, F)$, be a chordal extension of $G = (V, E)$. Since $(i,j) \notin E \Rightarrow A_{ij} = 0$, we have that for an arbitrary $X \in \mathbb{S}^n$

$$A \bullet X = \sum_{(i,j)\in E} A_{ij}X_{ij} = \sum_{(i,j)\in F} A_{ij}X_{ij}. \tag{2.3}$$

In other words, the entries X_{ij} where $(i,j) \notin E$ do not contribute to the sum because they are multiplied by zeros. We now use this to replace the inner products

in (2.1) and write the primal form of the SDP in the equivalent form

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in F} [A_0]_{ij} X_{ij} \\ \text{subject to} \quad & \sum_{(i,j)\in F} [A_q]_{ij} X_{ij} = b_q,\ q = 1, 2, \ldots, m \\ & X \in \mathbb{S}^n_+(F, ?), \end{aligned} \tag{2.4}$$

where we have used the fact that the values of the objective and constraint functions are not affected by completing X. From Theorem 2.2.1 the semidefinite constraint will be satisfied if and only if the submatrices of X corresponding to the maximal cliques of $G_{ch} = (V, F)$ are positive semidefinite. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$ be the maximal cliques of $G = (V, F)$, then (2.4) can be written as:

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in F} [A_0]_{ij} X_{ij} \\ \text{subject to} \quad & \sum_{(i,j)\in F} [A_q]_{ij} X_{ij} = b_q,\ q = 1, 2, \ldots, m \\ & X(C_k) \in \mathbb{S}^n_+(C_k),\ k = 1, 2, \ldots, p. \end{aligned} \tag{2.5}$$

We may decompose any $A_q \in \mathbb{S}(E, 0)$ into a summation of matrices of the form $A^k_q \in \mathbb{S}^n(C_k)$ since $E \subset F$, i.e.,

$$A_q = \sum_{k=1}^{p} A^k_q, \quad A^k_q \in \mathbb{S}^n(C_k),\ k = 1, 2, \ldots, p. \tag{2.6}$$

Using (2.6) we can write the inner product of A_q with an arbitrary matrix $X \in \mathbb{S}^n$ as

$$A_q \bullet X = \sum_{(i,j)\in F} [A_q]_{ij} X_{ij} = \sum_{k=1}^{p} \left(\sum_{(i,j)\in F} [A^k_q]_{ij} X_{ij} \right).$$

Since $A_q^k \in \mathbb{S}^n(C_k)$ we may replace this equality with

$$A_q \bullet X = \sum_{k=1}^{p} \left(A_q^k \bullet X(C_k) \right). \tag{2.7}$$

By using (2.7) we have

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{p} \left(A_0^k \bullet X(C_k) \right) \\ \text{subject to} \quad & \sum_{k=1}^{p} \left(A_q^k \bullet X(C_k) \right) = b_q, \quad q = 1, 2, \ldots, m \\ & X(C_k) \in \mathbb{S}_+^n(C_k), \ k = 1, 2, \ldots, p. \end{aligned} \tag{2.8}$$

Unfortunately this is not a standard SDP as the submatrices in the semidefinite constraints $X(C_k) \succeq 0$ share elements and so are not independent. To convert this into a standard SDP Fukuda *et. al.* introduce new independent variables $X_1, X_2, \ldots, X_p$ where $X_k \in \mathbb{S}_+^n(C_k)$ for $k = 1, 2, \ldots, p$, and additional constraints that ensure equality between the overlapping elements [36]. The SDP problem then becomes

$$\begin{aligned} \text{minimize} \quad & \sum_{k=1}^{p} \left(A_0^k \bullet X_k \right) \\ \text{subject to} \quad & \sum_{k=1}^{p} \left(A_q^k \bullet X_k \right) = b_q, \quad q = 1, 2, \ldots, m \\ & U_{ij} \bullet X_k - U_{ij} \bullet X_l = 0, \ (i, j, k, l \in \Lambda) \\ & X_k \in \mathbb{S}_+^n(C_k), \ k = 1, 2, \ldots, p \end{aligned} \tag{2.9}$$

where the constraint $U_{ij} \bullet X_k - U_{ij} \bullet X_l = 0$ enforces that the variable X_{ij} shared by maximal cliques k and l must be equal. Note that the single large semidefinite

constraint in (2.1) has been replaced by multiple semidefinite constraints on smaller matrix variables and that the SDP is now in a block diagonal form. This is important because SDP solvers can exploit this block diagonal form to calculate the next iterate of the Newton step more efficiently.

2.3.2 Exploiting Chordal Sparsity in the Dual SDP

Again following following [43], we now decompose the dual SDP using Theorem 2.2.3. Consider (2.2). Under our assumption on the aggregate sparsity pattern of the LMI the feasibility constraints of the dual SDP are

$$\sum_{i=1}^{q} y_q A_q + Z = A_0,\ Z \in \mathbb{S}^n_+(E,0). \tag{2.10}$$

Let $G_{ch} = (V,F)$ be a chordal extension of $G = (V,E)$, with set of maximal cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$. Using this chordal extension we can write (2.10) in the equivalent form

$$\sum_{q=1}^{m} y_q A_q + Z = A_0,\ Z \in \mathbb{S}^n_+(F,0). \tag{2.11}$$

Note that the constraint $Z_{ij} = 0$ if $(i,j) \in F \setminus E$ is implicit in these constraints since $A(y) = A_0 - \sum_{q=1}^{m} y_q A_q \in \mathbb{S}^n(E,0)$ for all $y \in \mathbb{R}^m$. Let A_q^k for $k = 1,2,\ldots,p$ and $q = 0,1,2,\ldots,m$ be matrices that satisfy

$$\sum_{k=1}^{p} \left(A_0^k - \sum_{q=1}^{m} y_q A_q^k \right) = A_0 - \sum_{i=1}^{q} y_q A_q.$$

The following proposition uses Theorem 2.2.3 to decompose the constraints of the dual SDP.

Proposition 2.3.1 The set of constraints (2.11) is feasible if and only if there exist matrices $Z_k \in \mathbb{S}^n(C_k)$ and $L_k(\theta) \in \mathbb{S}^n(C_k)$ for $k = 1, 2, \ldots, p$ that satisfy

$$Z_k = A_0^k - \sum_{q=1}^{m} y_q A_q^k - L_k(\theta),\ k = 1, 2, \ldots, p$$

$$Z_k \in \mathbb{S}_+^n(C_k),\ k = 1, 2, \ldots, p$$

where $L_k(\theta)$ are as defined in Theorem 2.2.3.

Proof 2.3.1 Let $Z = \sum_{k=1}^{p} Z_k$ and note that $\sum_{k=1}^{p} L_k(\theta) = 0$ to give the first condition. The second condition then follows from Theorem 2.2.3,

$$Z \in \mathbb{S}_+^n(F, 0) \Leftrightarrow Z = \sum_{k=1}^{p} Z_k,\ Z_k \in \mathbb{S}_+^n(C_k).$$

With these equivalent constraints the decomposition of the dual SDP is

$$\begin{aligned} \text{maximize} \quad & b^T y \\ \text{subject to} \quad & \sum_{q=1}^{m} y_q A_q^k + L_k(\theta) + Z_k = A_0^k,\ k = 1, 2, \ldots, p \\ & Z_k \in \mathbb{S}_+^n(C_k),\ k = 1, 2, \ldots, p. \quad \square \end{aligned} \tag{2.12}$$

The SDPs (2.9) and (2.12) are in fact Lagrange duals of one another and the derivation is given in the following subsection. For each equality constraint $U_{ij} \bullet X_k - U_{ij} \bullet X_l = 0$ in the primal problem there is an associated dual variable θ_{ijkl}. We note that this decomposition has been exploited to decompose problems in

distributed robust stability analysis using IQCs [58].

For further information on the conversion process we refer the reader to papers [36, 44, 43]. Solvers that exploit chordal sparsity in general SDPs are available, in particular SMCP [54] and SDPA-C [37]. Also available is SparseCoLO which automates the reformulation of SDPs with chordal sparsity to facilitate solution using standard solvers [43].

2.3.3 Lagrangian Duality Between the Decomposed Problems

To complete the picture of the chordal decomposition of sparse SDPs we will briefly give our own derivation of the fact that the decomposed primal and decomposed dual SDPs are themselves dual problems. First, we write (2.12) in the form

$$\begin{aligned} &\text{maximise} \quad b^T y \\ &\text{subject to} \quad A_0^k - \sum_{q=1}^{m} y_q A_q^k - L_k(\theta) \in \mathbb{S}_+^n(C_k),\ k = 1, 2, \ldots, p. \end{aligned}$$

The Lagrangian function for this optimisation problem is

$$L(y, \theta, X) = b^T y + \sum_{k=1}^{p} \left(A_0^k - \sum_{q=1}^{m} y_q A_q^k - L_k(\theta) \right) \bullet X_k$$

with the dual variables $X_k \in \mathbb{S}_+^n(C_k)$ for $k = 1, 2, \ldots, p$. To find the dual function $g(X)$ we maximise the Lagrangian function over y and θ. Since the Lagrangian function is an affine function in terms of y and θ the dual function can be evaluated

as

$$g(X) = \begin{cases} \sum_{k=1}^{p} A_0^k \bullet X_k & \text{if } \frac{\partial L}{\partial y} = 0 \text{ and } \frac{\partial L}{\partial \theta} = 0 \\ \infty & \text{otherwise} \end{cases}.$$

Then minimising the dual function the dual problem is equivalent to

$$\begin{aligned} \text{minimise} \quad & \sum_{k=1}^{p} A_0^k \bullet X_k \\ \text{subject to} \quad & \sum_{k=1}^{p} \left(A_q^k \bullet X_k\right) = b_q, \ q = 1, 2, \ldots, m \\ & U_{ij} X_k - U_{ij} X_l = 0, \ (i, j, k, l) \in \Lambda \\ & X_k \in \mathbb{S}_+^n(C_k) \text{ for } k = 1, 2, \ldots, p \end{aligned}$$

which is the decomposed primal SDP that resulted from applying Grone's theorem.

2.4 Chapter Summary

In summary, the body of work that began with Grone's and Agler's theorems allows us to decompose the primal and dual forms of a sparse SDP. The decomposed primal and dual problems are then also Lagrange duals of one another, bringing into focus an elegant picture where the duality between the primal and dual SDP is mirrored in the duality between Grone's and Agler's theorems. This is depicted in Figure 2.7.

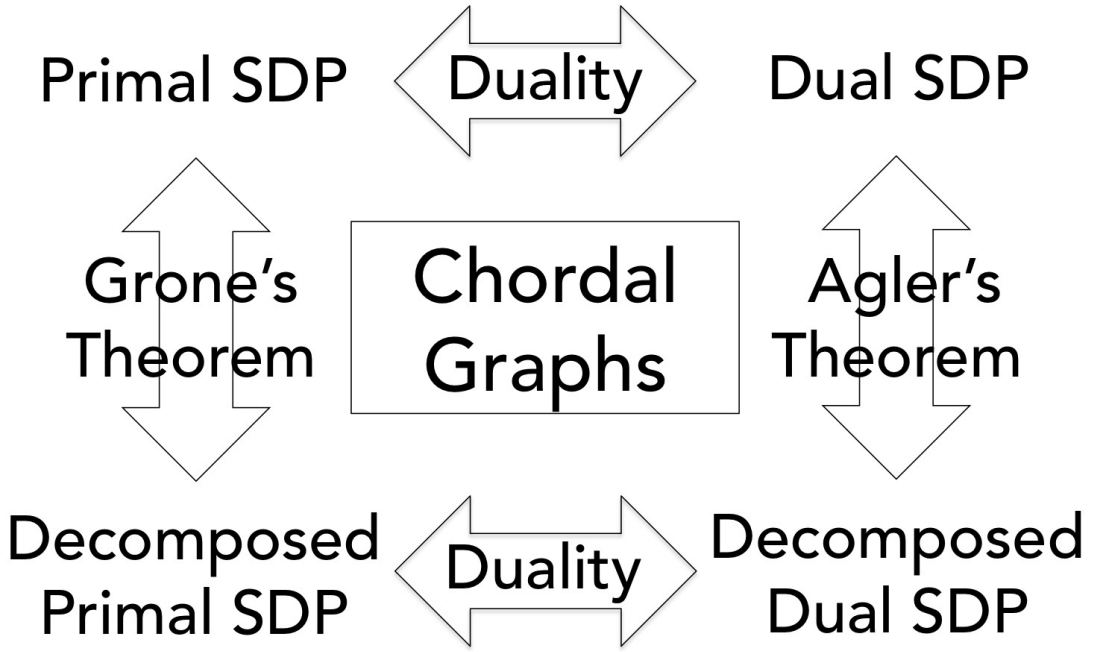


Figure 2.7: Summary of the duality relationships between the primal and dual SDPs and the decomposed primal and dual SDPs.

Chapter 3

Analysis of Sparse Linear Systems

3.1 Introduction

Analysis questions in control and systems theory are often formulated as Linear Matrix Inequalities (LMIs) and solved using convex optimisation algorithms. For large LMIs it is important to exploit the structure or sparsity within a problem in order to solve the associated SDPs efficiently. In this chapter we consider how to exploit chordal sparsity in the SDPs that arise when constructing Lyapunov functions and calculating bounds on the H_∞ norm for LTI systems in both the continuous and discrete-time cases.

These SDPs are challenging to solve for general systems of large dimension, and therefore we consider how to address the problem for large but sparse systems, i.e., when the data matrices describing the dynamical system are sparse. By designing the matrix variables in the problem so as to have a chordal graphical structure we can convert the semidefinite constraints in the problem into an equivalent set

of smaller semidefinite constraints with some additional equality constraints. For sparse systems this approach can improve the efficiency with which SDPs are solved and can allow larger systems to be analysed using these methods. The work in this chapter was published in [48].

3.2 Continuous-Time Lyapunov Stability

We begin this chapter with a review of Lyapunov stability theory in continuous time. Consider the autonomous dynamical system

$$\dot{x}(t) = f\left(x(t)\right), \; x(0) = x_0, \tag{3.1}$$

where $f : D \to \mathbb{R}^n$ is locally Lipschitz in a domain $D \subset \mathbb{R}^n$. Suppose there exists a point x^* such that $f(x^*) = 0$, then we say that x^* is an equilibrium point for (3.1). Without loss of generality, we may perform a simple change of coordinates to place a given equilibrium point at the origin. We will be concerned with the stability properties of equilibria and hence we now define the notions of stability and asymptotic stability.

Definition 3.2.1 The system (3.1) is stable about $x = 0$ if for every $\epsilon > 0$ there exists $\delta = \delta(\epsilon) > 0$ such that if $\|x(0)\| < \delta$, then $\|x(t)\| < \epsilon$ for all $t \geq 0$.

Definition 3.2.2 The system (3.1) is asymptotically stable about $x = 0$ if it is stable and, additionally, there exists $\delta > 0$ such that if $\|x(0)\| < \delta$, then $\lim_{t\to\infty} \|x(t)\| = 0$.

Theorem 3.2.1 [59] Consider (3.1), and let $D \subset \mathbb{R}^n$ be a neighbourhood of the origin. If there is a continuously differentiable function $V : D \to \mathbb{R}$ such that the following two conditions are satisfied:

$$\begin{aligned} &1)\ V(x) > 0 \text{ for all } x \in D \setminus \{0\} \text{ and } V(0) = 0, \\ &2)\ \dot{V}(x) = \frac{\partial V}{\partial x} f(x) \leq 0 \text{ for all } x \in D, \end{aligned}$$

then the origin is a stable equilibrium point. If in condition (2) above, $\dot{V}(x) < 0$ for all $x \in D$, then the origin is asymptotically stable. If $D = \mathbb{R}^n$ and $V(x)$ is radially unbounded, i.e., $V(x) \to \infty$ as $\|x\| \to \infty$, then the result holds globally.

For a linear time-invariant system $\dot{x} = Ax$ the conditions for global asymptotic stability reduce to checking the feasibility of the following LMI, which we will call the Lyapunov LMI:

$$P \succ 0, \quad Q = A^T P + PA \prec 0. \tag{3.2}$$

It is well known that this problem can be solved by picking a $Q \prec 0$ and solving for P using linear algebra [60]. Our motivation for studying the Lyapunov LMI is that it appears as a block within many key LMIs in control and systems theory that cannot be solved using linear algebra alone.

For example, a generalisation of the Lyapunov LMI problem that cannot be solved using linear algebra is the following robust stability problem. Given matrices $A_1, A_2, \ldots, A_m \in \mathbb{R}^{n \times n}$, construct a Lyapunov function for the system

$$\dot{x} = Ax,\ A \in \text{conv}(A_1, A_2, \ldots, A_m), \tag{3.3}$$

where conv$(\cdot)$ denotes the convex hull. It is well-known that the existence of a Lyapunov function that holds simultaneously for all of the vertices of the convex hull i.e., $\{A_1, A_2, \ldots, A_m\}$ is sufficient for the system in (3.3) to be asymptotically stable [10].

We next outline how chordal sparsity can be exploited in the Lyapunov LMI when the matrix A describing the system dynamics is sparse. Let $P = P^T$ and write the Lyapunov LMI in the form

$$\begin{bmatrix} P & 0 \\ 0 & -(A^T P + PA) \end{bmatrix} \succ 0. \tag{3.4}$$

Let $W_1, W_2, \ldots, W_m \in \mathbb{S}^n$ be symmetric basis matrices that correspond to the $m = n(n+1)/2$ free variables in P and define the matrices $B_1, B_2, \ldots, B_m \in \mathbb{S}^{2n}$ by

$$B_i = \begin{bmatrix} W_i & 0 \\ 0 & -(A^T W_i + W_i A) \end{bmatrix}, \ i = 1, 2, \ldots, m. \tag{3.5}$$

Let $I \subset \{1, 2, \ldots, m\}$ be an index set that selects which elements of P are allowed to be nonzero. We may then formulate a sparse LMI feasibility problem as

$$B_I(y) = \sum_{i \in I} y_i B_i \succ 0. \tag{3.6}$$

For a given I, let E be the aggregate sparsity pattern for (3.6) and suppose that the graph $G = (V, E)$ is a chordal graph, we then say that the matrix variable $B_I(y)$ has a chordal sparsity pattern. In the next subsection we consider some special cases where the sparsity pattern of A makes it possible to pick I, i.e., design

the sparsity pattern of P, so that the aggregate sparsity pattern of (3.6) is chordal.

In the following subsections we consider four classes of A matrices: banded, cyclic, tree and Metzler. We choose a P matrix with a special structure for each class so that both P and $Q = A^T P + PA$ have a chordal sparsity pattern. In addition, for the banded and cyclic cases, we show that there exist a sequence of chordal P matrices of increasing density up to a complete graph that induce a sequence of chordal Q matrices. Therefore in these cases it is possible to test a sequence of increasingly dense SDPs whilst exploiting chordal sparsity until a Lyapunov function is found.

3.2.1 Banded Matrices

The first special case that we consider is when A is a banded matrix. We say that $A \in \mathbb{R}^{n \times n}$ is a banded matrix of bandwidth d if

$$A_{ij} = 0 \text{ if and only if } |i - j| > d$$

where $d \geq 0$ is an integer. Banded matrices are well known to be chordal and the multiplication of two banded matrices is again banded. Therefore we may generate a sequence of chordal Q matrices by setting P to be a banded matrix of bandwidth $d = 0, 1, 2, \ldots, n-2$.

To generalise this we may allow for some entries within the bandwidth to be zero. We say that $A \in \mathbb{R}^{n \times n}$ is a generalised banded matrix of bandwidth d if there

exists an integer $d \geq 0$ such that

$$A_{ij} = 0 \text{ if } |i - j| > d.$$

Then the graph describing the sparsity pattern of a generalised banded matrix can always be extended to the graph describing the sparsity pattern of a banded matrix of bandwidth d. Hence the same approach for constructing a sequence of chordal Lyapunov functions can be used for the generalised banded case as in the banded case.

3.2.2 Cyclic Matrices

We next define another class of matrices for which we can find P and Q with chordal sparsity patterns. If an $n \times n$ matrix A has a sparsity pattern given by

$$A_{ij} = \begin{cases} 1 & \text{if } |i - j| \leq 1 \\ 1 & \text{if } (i, j) = (1, n) \text{ or } (n, 1) \\ 0 & \text{otherwise,} \end{cases} \tag{3.7}$$

then we call it a cyclic matrix. Let the sparsity pattern of P be defined by

$$P_{ij} = \begin{cases} 1 & \text{if } i = j \\ 1 & \text{if } i + j = n + 1 \\ 1 & \text{if } i + j = n + 2 \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

We will call the sparsity pattern of P defined above a zig-zag pattern and we say that the zig-zag is of length $n-1$.

Theorem 3.2.2 Given $A \in \mathbb{R}^{n\times n}$ with a cyclic sparsity pattern, the sparsity pattern of P defined in (3.8) is chordal and results in a $Q = A^TP + PA$ with a chordal sparsity pattern.

Proof 3.2.1 Let A and P have sparsity patterns as defined in (3.7) and (3.8). Note that P can be rearranged to be a banded matrix and hence has a chordal sparsity pattern. The sparsity pattern of $Q = A^TP + PA$ is given by

$$Q_{ij} = \begin{cases} 1 & \text{if } |i-j| \leq 1 \\ 1 & \text{if } n \leq i+j \leq n+3 \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

The proof that Q as defined above is chordal is by induction. We use the notation Q^n to denote the Q matrix for an n node cycle. For $n = 1, 2, 3$ there cannot be a cycle of length greater or equal to 4 so these graphs must be chordal. We first consider the even case and take Q^4 as our base case. Since it is a complete graph it is chordal. Next consider Q^n with n even and assume that it is chordal. Construct Q^{n+2} by relabelling the nodes of Q^n by incrementing $v_i := v_{i+1}$, for $i = 1, 2, \ldots, n$ and then adding two new nodes $\{v_1, v_{n+2}\}$ to the periphery of the network and connecting them according to (3.9), see Figure 3.1.

Let the perfect elimination ordering for Q^n be denoted by α. If we can find a perfect elimination ordering for Q^{n+2} of the form $\langle v_1, v_{n+2}, \alpha\rangle$ then we have shown

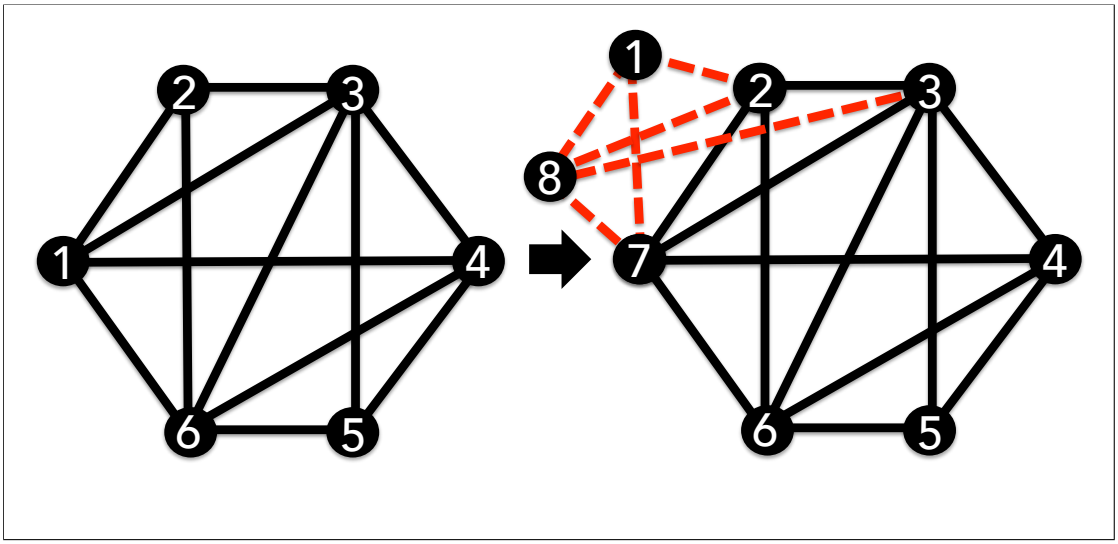


Figure 3.1: Constructing Q^8 from Q^6.

that Q^{n+2} is chordal. Consider the first row of Q^{n+2}

$$Q^{n+2}_{1,k} = 1, \text{ for } k \in \{1, 2, n+1, n+2\}, \ 0 \text{ otherwise.}$$

In order for node 1 to be simplicial, nodes $2, n+1$ and $n+2$ must form a clique. From (3.9) we see that $Q^{n+2}_{ij} = 1$ for $n+2 \leq i+j \leq n+5$ and hence these nodes do indeed form a clique. Similarly for node $n+2$, after eliminating node 1 the $n+2$th row of Q^{n+2} is

$$Q^{n+2}_{n+2,k} = 1, \text{ for } k \in \{2, 3, n+1, n+2\}, \ 0 \text{ otherwise.}$$

Again using the definition of Q^{n+2} we see that the nodes $2, 3$ and $n+2$ form a clique and hence $n+2$ is a simplicial vertex. Hence the ordering $\langle v_1, v_{n+2}, \alpha \rangle$ is a perfect elimination ordering, and by induction Q^n is chordal for all even n. The same argument can be applied to the odd n case, which completes the proof. □

Next we show by example that a sequence of chordal P and Q matrices exists for the cyclic case. The basis of the argument is that P and Q can be expressed as generalised banded graphs by a particular relabelling of the nodes. Consider Figure 3.2, which shows $G(P)$ and $G(A)$ with the nodes labelled according to the following procedure: For $G(P)$, start by picking a node at the end of the zig-zag pattern and label it node 1, then label the unique unlabelled node connected to node 1 with node 2, and so on up to node n. For $G(A)$, label the nodes so that they have the same labels as those in $G(P)$. Now consider the adjacency matrices for $G(P)$ and $G(A)$

$$P = \begin{bmatrix} p_{11} & p_{12} & 0 & 0 & 0 & 0 \\ p_{12} & p_{22} & p_{23} & 0 & 0 & 0 \\ 0 & p_{23} & p_{33} & p_{34} & 0 & 0 \\ 0 & 0 & p_{34} & p_{44} & p_{45} & 0 \\ 0 & 0 & 0 & p_{45} & p_{55} & p_{56} \\ 0 & 0 & 0 & 0 & p_{56} & p_{66} \end{bmatrix}, \quad A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 \\ a_{12} & a_{22} & 0 & a_{24} & 0 & 0 \\ a_{13} & 0 & a_{33} & 0 & a_{35} & 0 \\ 0 & a_{24} & 0 & a_{44} & 0 & a_{46} \\ 0 & 0 & a_{35} & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & a_{46} & a_{56} & a_{66} \end{bmatrix}.$$

We see that these permuted P and A are generalised banded matrices and that this pattern extends to cycles of any finite length. Therefore we can construct a sequence of candidate Lyapunov functions using banded P matrices of increasing bandwidth exactly as in the case of generalised banded matrices.

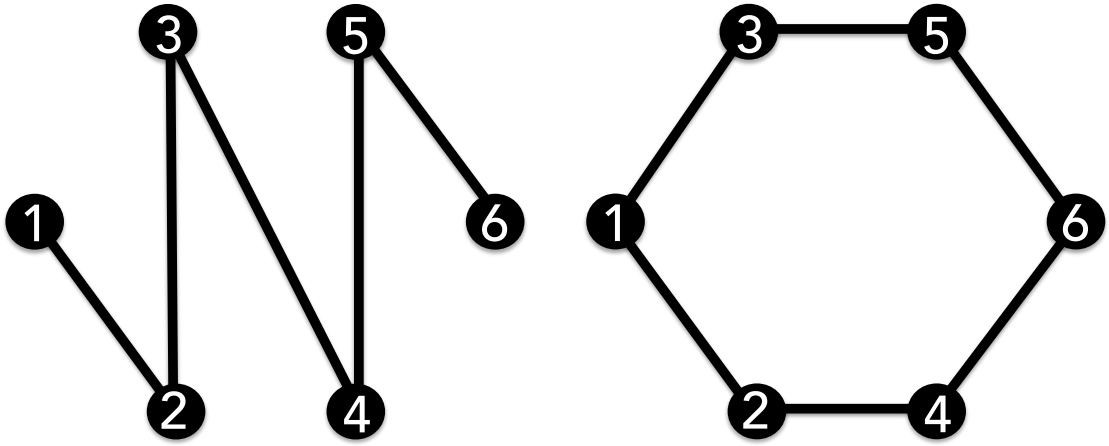


Figure 3.2: $G(P)$ (left) and $G(A)$ (right) for the cyclic case with the nodes relabelled.

3.2.3 Tree Matrices

The third special case that we consider is when A has a sparsity structure given by a tree graph i.e., a graph $T = (V, E)$ that is connected but has no cycles. We next give some basic terminology used to describe the nodes in the tree. For any tree T we may arbitrarily pick one node to be the root of the tree and label it r. A node u is the parent of node v and v is the child of u if u is the unique node adjacent to v on the unique path from r to v. We define the siblings of a node u to be the set of all nodes with the same parent as u (including u). Given node $i \in V$, we denote its parent by $\mathrm{Par}(i)$, its children by $\mathrm{Ch}(i)$ and its siblings by $\mathrm{Sib}(i)$. We also number the nodes in a topological ordering, meaning that each node in the graph is assigned a number from $1, \ldots, n$ such that each node has a higher number in the ordering than its children.

Given a tree $T = (V, E)$ consisting of n nodes, let $M(i) = \{i\} \cup \mathrm{Par}(i) \cup \mathrm{Ch}(i)$ for

$i = 1, 2, \ldots, n$ and define the sparsity pattern of A by

$$A_{ij} = \begin{cases} 1 & \text{if } j \in M(i) \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

Let the sparsity pattern of P be defined by

$$P_{ij} = \begin{cases} 1 & \text{if } j \in \text{Sib}(i) \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

Theorem 3.2.3 Given $A \in \mathbb{R}^{n \times n}$ with a sparsity pattern defined by a tree, the sparsity pattern of P defined in (3.11) is chordal and results in a $Q = A^T P + PA$ with a chordal sparsity pattern.

Proof 3.2.2 Let A and P have sparsity patterns as defined in (3.10) and (3.11). Note that P can be rearranged to be block diagonal and hence has a chordal sparsity pattern. The element-wise equation for the sparsity pattern of Q is

$$Q_{ij} \neq 0 \Leftrightarrow \sum_{k=1}^{n} A_{ik} P_{kj} + \sum_{k=1}^{n} P_{ik} A_{kj} \neq 0$$

where we have used the fact that the sparsity pattern of A is symmetric. From the definitions of the sparsity pattern of A and P we conclude that the sparsity pattern of Q is given by

$$Q_{ij} = \begin{cases} 1 & \text{if } (M(i) \cap \text{Sib}(j) \neq \emptyset) \text{ or } (\text{Sib}(i) \cap M(j) \neq \emptyset) \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

Let the topological ordering $\alpha = \{1, 2, \ldots, n\}$ be a candidate for a perfect elimina-

tion ordering. The subgraph induced by $\{v_i, \ldots, v_n\}$ cannot include the children of v_i or the children of its siblings, since these nodes have a lower number in the topological ordering. The remaining nodes that v_i is connected to are: its siblings with a higher number in the ordering, its parent and the siblings of its parent. It remains to be shown that all of the siblings of i are adjacent to all of the siblings of the parent of node i. Let $j \in \mathrm{Sib}(i)$ and $k \in \mathrm{Sib}(\mathrm{Par}(i))$. Using (3.12) we have

$$M(j) \cap \mathrm{Sib}(k) = \mathrm{Par}(i) \neq \emptyset \Rightarrow Q_{jk} = 1, Q_{kj} = 1.$$

Hence the siblings of the parent of node i are adjacent to all of the siblings of i. Therefore v_i is a simplicial vertex and α is a perfect elimination ordering. We conclude that Q is a chordal graph. $\square$

The sparsity pattern for P given above does not guarantee that a Lyapunov function of this form exists for tree matrices, only that we have a candidate Lyapunov function that we can test efficiently by exploiting chordal sparsity. Unlike the banded and cyclic cases, for the tree cases we do not know if a there exists a sequence of chordal P and Q matrices and so this will be a question for future research.

3.2.4 Metzler and Triangular Matrices

Our final special cases are Metzler and Triangular matrices. A matrix $A \in \mathbb{R}^{n \times n}$ is said to be Metzler if all its off diagonal elements are nonnegative i.e, $a_{ij} \geq 0$ $\forall i \neq j$.

Proposition 3.2.1 [47] Let $A \in \mathbb{R}^{n \times n}$ be a Metzler matrix. Then the following

statements are equivalent:

i) the matrix A is Hurwitz

ii) there exists a positive definite, diagonal matrix $P \succ 0$ such that $Q = A^T P + PA \prec 0$.

If we further assume that A is sparse, then by choosing P to be diagonal we ensure that the structure of A is preserved in Q. Let $G = (V, E)$ be the graphical structure of Q, then we can find a chordal extension $G = (V, F)$ where $E \subseteq F$. Therefore, given a sparse Metzler matrix we can apply the chordal decomposition to the SDP so that we can efficiently test whether the system is stable.

Remark 3.2.1 It is known that stable triangular matrices also admit a diagonal Lyapunov function [61]. Therefore, when a matrix is triangular and sparse we can use the same method to decompose the problem as for Metzler matrices.

3.2.5 Numerical Results

Next we present some numerical results for these special cases that demonstrate the improvements in efficiency that are possible using the chordal sparsity approach. In our experiments, we make use of the MATLAB package SparseCoLO [43] which detects chordal sparsity in an LMI and preprocesses the data so that the matrix variables are block diagonal and then calls either SeDuMi, SDPA or SDPT3 to solve the problem.

To test the chordal sparsity approach, we generated sparse banded, cyclic, tree, Metzler and sparse triangular A matrices with negative eigenvalues. We then compared the time taken for SeDuMi and SparseCoLO+SeDuMi to solve the Lyapunov

LMI. For all of our experiments the SDP was considered to be solved when the primal-dual gap had been reduced to less than $\epsilon = 10^{-9}$. The experiments were run on a MacBook Pro with a 2.9GHz processor and 8GB of RAM.

For the case of banded matrices we chose A and P to be banded with order $d = 5$. To generate the tree matrices we simply started with the trivial tree consisting of a single node, and then at each step we added an undirected edge to a node not in the tree, repeating until all of the nodes were connected to the tree, see Appendix A.3.2 for details. For the Metzler and triangular cases we generated sparse non-chordal graphs using the method described in Appendix A.3.3.

To generate the stable A matrices required for each experiment we first generated a mask matrix S with the desired sparsity pattern i.e., for the banded matrix experiment S was chosen to be a banded with order $d = 5$. Then given n (`n`) and a shift parameter $\kappa > 0$ we generated sparse random A matrices in MATLAB using the following code:

```
A = rand(n,n)-0.5;

A = A.*S;

lambda = max(real(eig(A)));

A = A -(lambda+κ)*eye(n);
```

Note that increasing κ shifts the spectrum of A further into the left-half plane.

Tables 3.1-3.5 show a comparison of the CPU time in seconds required to solve check the feasibility of the Lypapunov LMI using SeDuMi and SparseCoLO for

banded, cyclic, tree, Metzler and triangular matrices respectively. Each experiment was repeated 100 times for each value of n and κ was set to a value of 1. The variable maxC corresponds to the largest maximal clique in the sparsity pattern of the LMI. Note that MaxC was kept constant as the number of nodes was increased. The numbers in brackets e.g., (230,17) denote the size and the number of blocks of the Schur complement matrix respectively.

We see that the size of the Schur complement matrix is larger when we apply the chordal decomposition method. This is due to the extra variables that are introduced (the dual variables of the equality constraints). However, the problem is decomposed into a number of blocks of smaller size which an SDP solver, in this case SeDuMi, can exploit to solve the problem more efficiently. Hence the chordal decomposition method is significantly quicker in these sparse cases with small maximal cliques, this is shown most clearly in Figure 3.3 which collects together the data from Tables 3.1-3.5.

As the number of nonzero elements in the matrix A increases, the density of the matrices in the LMI also tends to increase. This leads to larger maximal cliques, with more overlapping elements, which requires us to introduce more dual variables and increases the size of the Schur complement matrix in the chordal sparsity approach. Table 3.6 shows the way in which increasing maximal clique sizes affects the time taken to solve the Lyapunov LMI for Metzler matrices with $n = 400$. We see that for small maxC the chordal decomposition is more efficient than the standard dense method, but as maxC is made larger the increased size of the Schur complement matrix outweighs the advantages of decomposing the

Banded Matrices (maxC = 11)		
n	SeDuMi	SparseCoLO+SeDuMi
100	3.1 (585,2)	2.3 (925,15)
200	10.7 (1185,2)	4.4 (1865,27)
300	34.3 (1785,2)	7.8 (2805,39)
400	65.3 (2385,2)	8.3 (3745,51)
500	122.6 (2985,2)	11.8 (4685,63)
600	197.9 (3585,2)	12.7 (5640,76)
700	373.6 (4185,2)	14.2 (6595, 89)
800	679.4 (4785,2)	17.5 (7535,101)

Table 3.1: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time in seconds to check feasibility of the Lyapunov LMI for banded matrices, (size of Schur complement matrix, no. of blocks).

problem into blocks.

Cyclic Matrices (maxC = 4)		
n	SeDuMi	SparseCoLO+SeDuMi
100	3.5 (199,2)	1.5 (244,18)
200	9.2 (399,2)	2.1 (489,33)
300	26.0 (599,2)	2.9 (734,48)
400	52.0 (799,2)	3.1 (986,65)
500	117.5 (999,2)	4.2 (1231,80)
600	152.0 (1199,2)	5.2 (1476,95)
700	347.8 (1399,2)	6.5 (1727,111)
800	418.6 (1599,2)	7.1 (1973,127)

Table 3.2: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time in seconds to check feasibility of the Lyapunov LMI for cyclic matrices, (size of Schur complement matrix, no. of blocks).

Tree Matrices (maxC = 15)		
n	SeDuMi	SparseCoLO+SeDuMi
100	2.8 (199,2)	1.1 (280,34)
200	13.6 (393,2)	3.3 (643,73)
300	51.9 (588,2)	5.8 (842,110)
400	88.3 (755,2)	7.7 (1063,146)
500	224.7 (980,2)	8.7 (1540,180)
600	290.2 (1184,2)	9.9 (1684,217)
700	379.4 (1367,2)	10.4 (1965,253)
800	466.4 (1598,2)	13.9 (2502,289)

Table 3.3: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time in seconds to check feasibility of the Lyapunov LMI for tree matrices, (size of Schur complement matrix, no. of blocks).

Sparse Metzler (maxC = 10)		
n	SeDuMi	SparseCoLO+SeDuMi
100	2.8 (100,2)	0.9 (146,14)
200	15.8 (200,2)	1.6 (230,17)
300	39.6 (300,2)	3.2 (411,30)
400	66.9 (400,2)	3.3 (496,37)
500	155.0 (500,2)	4.0 (651,50)
600	296.0 (600,2)	4.3 (732,53)
700	324.6 (700,2)	6.6 (827,61)
800	391.0 (800,2)	6.9 (982,71)

Table 3.4: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time in seconds to check feasibility of the Lyapunov LMI for sparse Metzler matrices, (size of Schur complement matrix, no. of blocks).

Sparse Triangular (maxC = 10)		
n	SeDuMi	SparseCoLO+SeDuMi
100	3.8 (100,2)	0.9 (128,11)
200	14.0 (200,2)	1.9 (291,24)
300	31.4 (300,2)	2.3 (391,27)
400	71.2 (400,2)	2.7 (533,41)
500	187.2 (500,2)	4.2 (645,43)
600	237.1 (600,2)	4.7 (931,57)
700	336.6 (700,2)	5.1 (936,58)
800	450.3 (800,2)	5.5 (1023,68)

Table 3.5: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time in seconds to check feasibility of the Lyapunov LMI for sparse triangular matrices, (size of Schur complement matrix, no. of blocks).

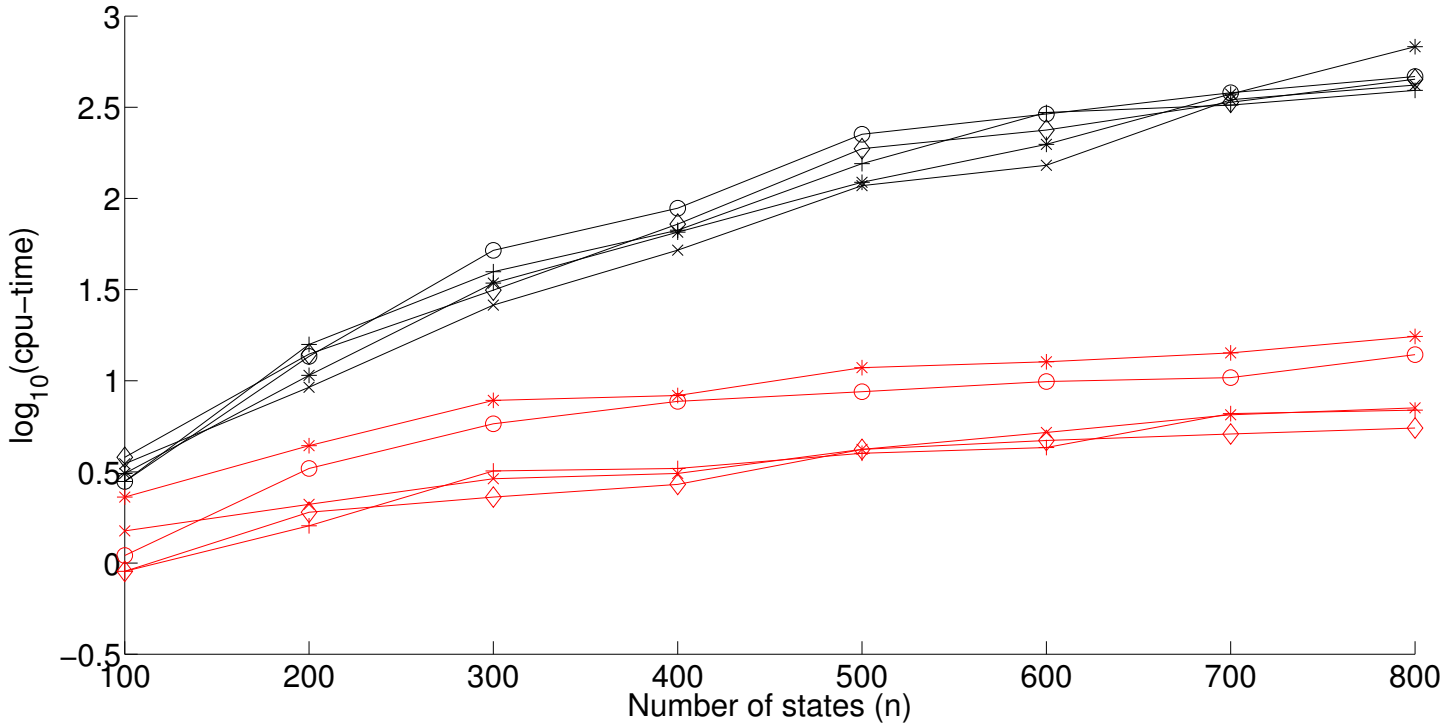


Figure 3.3: Logarithm of the cpu-time (in seconds) vs the problem size for Banded (*), Cyclic (x), Tree (o), Metzler (+), Triangular (diamond) matrices. The black data correspond to instances that were solved using SeDuMi and the red lines correspond to instances that were solved using SparseCoLO.

Sparse Metzler (n= 400)			
maxC	nnz(A)	SeDuMi	SparseCoLO+SeDuMi
6	2022	57.5 (400,2)	3.3 (486,35)
10	2606	65.1 (400,2)	3.4 (554,36)
17	4874	69.4 (400,2)	5.3 (980,44)
20	5240	72.0 (400,2)	14.9 (3418,192)
27	9260	70.0 (400,2)	129.9 (7182,172)

Table 3.6: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time in seconds to solve the Lyapunov LMI for Metzler matrices with $n = 400$ nodes whilst varying the size of the maximal cliques, (size of Schur complement matrix, no. of blocks).

3.2.6 General Case

In this subsection we consider the case in which we have been given a large, sparse system $\dot{x} = Ax$ that does not fall into one of the special cases covered previously. In this situation we need a method of generating a sparsity pattern for P that will allow us to exploit chordal sparsity. We next show that picking a sparsity pattern for P so that the resulting SDP is sparse is a difficult problem in general, and thus propose an iterative method based on chordal extensions to circumvent this issue.

Given a sparse LTI system $\dot{x} = Ax$, let $Y \in \mathbb{R}^{n \times n}$ be defined by

$$Y_{ij} = \begin{cases} 1 & \text{if } A_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

For $k = 1, 2, \ldots, m$, where $m = n(n+1)/2$, let W_k be the symmetric basis matrix associated with the kth free variable in P and define $B_k \in \mathbb{S}^n$ by

$$[B_k]_{ij} = \begin{cases} 1 & \text{if } [Y^T W_k + W_k Y]_{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases}.$$

We can now state the problem of selecting the sparsity pattern of P more precisely: given an integer $0 < \lambda < m$, choose $y_1, y_2, \ldots, y_m \in \{0, 1\}$ to minimise the number of nonzero entries in the matrix $B(y) = \sum_{i=1}^{m} B_i y_i$, subject to the constraint that $\sum_{i=1}^{m} y_i = \lambda$. This can be expressed as the following optimisation problem

$$\begin{aligned} \text{minimise} \quad & \text{card}(B(y)) \\ \text{subject to} \quad & B(y) = \sum_{i=1}^{m} B_i y_i \\ & \sum_{i=1}^{m} y_i = \lambda \\ & y_i \in \{0, 1\}, \quad i = 1, 2, \ldots, m, \end{aligned} \tag{3.13}$$

where card() is the cardinality function. By relaxing the integer constraints in (3.13) we may obtain a problem with convex constraints

$$\begin{aligned} \text{minimise} \quad & \text{card}(B(y)) \\ \text{subject to} \quad & B(y) = \sum_{i=1}^{m} B_i y_i \\ & \sum_{i=1}^{m} y_i = \lambda \\ & 0 \leq y_i \leq 1, \quad i = 1, 2, \ldots, m. \end{aligned} \tag{3.14}$$

Let $x = \text{vec}(B(y))$, i.e., a n^2-dimensional vector formed by stacking the columns of $B(y)$, then, since convexity is preserved under linear transformations we may write this problem as

$$\begin{aligned} \text{minimise} \quad & \text{card}(x) \\ \text{subject to} \quad & x \in C, \end{aligned} \tag{3.15}$$

where C is a convex set. Problem 3.15 is well known to NP-hard, and so we can conclude that the problem of minimising the cardinality of the Q matrix in the Lyapunov equation is a challenging problem.

A commonly used approximation to the cardinality minimisation problem is obtained by replacing the nonconvex cardinality function with the L_1 norm. However, minimising the L_1 norm does not guarantee that the resulting matrices have sparse chordal extensions which is required for our purposes. We instead present a heuristic for picking a sequence of P matrices to minimise the the number of edges that must be added in chordal extensions of Q. We call the heuristic the Chordal Powers algorithm because basic idea is to pick a a sequence of P matrices that have the same sparsity pattern as increasing powers of A.

The motivation for picking this sequence of P matrices comes from the notion of the power of a graph. For a graph $G = (V, E)$, we denote by $G^k = (V, F)$ the graph with the same vertices as G and edge set defined by

$$F = \{ (i, j) \mid d(i, j) \leq k \}$$

where $d(i, j)$ is the standard graph distance in G. The graph G^k is then said to be the kth power of G. In terms of adjacency matrices, if $G(A)$ is the graph corresponding to a sparse symmetric matrix A, then the power graph $G^k(A) = G(A^k)$.

Let us now return to the Lyapunov equation, and consider for the case of a sparse A matrix. By selecting P to have the same sparsity pattern as the kth power of

A, then, by the definition of the powers of a graph, Q will have the same sparsity pattern as $G^{k+1}(A)$. This implies that any two nodes that were a distance further apart than $k+1$ in $G(A)$ will not be adjacent in $G(Q)$. Based on this observation we propose the following algorithm, which we call the Chordal Powers algorithm, for checking a sequence of increasingly dense quadratic Lyapunov functions.

Chordal Powers Algorithm: Given $A \in \mathbb{R}^{n\times n}$. Pick maximum clique threshold $M > 0$. Initialise $k, m = 0$

while $m < M$

$S_k := (A^T + A)^k, \;\; G(P_k) := G_{ch}(S_k) = (V, E_k)$

$R_k := A^T S_k + S_k A, \;\; G(Q_k) := G_{ch}(R_k) = (V, F_k)$

solve SDP:

$$\begin{bmatrix} P_k & 0 \\ 0 & Q_k \end{bmatrix} \succeq \epsilon I$$

$Q_k = -(A^T P_k + P_k A)$

$P_k \in \mathbb{S}^n_+(E_k, 0), \;\; Q_k \in \mathbb{S}^n_+(F_k, 0)$

if SDP is feasible

Terminate: A Lyapunov function has been found

else

$m := \text{maxC}(G(Q_k))$

$k := k + 1$

end

end

An example of the output of this algorithm for a banded matrix example is given in Appendix A.2. Note that this algorithm is different to that proposed in [48], which used a sequence of banded P matrices of increasing bandwidth. We next provide some numerical results regarding the time taken to construct a Lyapunov function using the Chordal Power algorithm. We compare the total time required to solve the sequence of SDPs in the Chordal Power algorithm using SeDuMi and SparseCoLO+SeDuMi. The purpose of this comparison is to demonstrate the benefits of exploiting chordal sparsity for the case when the system is sparse, as well as some of the limitations of the approach.

We first briefly introduce some notation that will be used in the tables of results. The variable 'dim.Schur' is the dimension of the Schur Complement matrix; 'max.Bl' is the maximum block size in the SDP problem i.e., the size of the largest semidefinite constraint in the problem; 'av.k' is the average of the variable k in the Chordal Power algorithm required to find a feasible solution across all experiments for systems of a particular dimension. For example, $k = 0$ is a diagonal Lyapunov function, $k \geq 1$ is a Lyapunov function with the same sparsity pattern as the matrix $(A^T + A)^k$.

For our first experiment we wished to test the chordal sparsity approach in the most favourable setting - sparse matrices where the sparsity pattern is close to a chordal sparsity pattern. With this in mind, we first generated chordal graphs with maximal cliques bounded by a threshold of 10 nodes using a function called `chordalGen(n,threshold)`, see Appendix A.3.1 for details. We then added some sparse random entries using the MATLAB command `S = sprand(n,n,rho)` with

$\rho = 0.1/n$. We then subtracted the largest eigenvalue of the matrix plus a positive parameter, e.g., $\kappa = 0.1$ to ensure that the matrices were Hurwitz. The code used for generating the matrices was as follows

```
S = chordalGen(n,threshold)+ sprand(n,n,rho)

S = abs(S)>0;

A = rand(n,n)-0.5;

A = A.*S;

lambda = max(real(eig(A)));

A = A -(lambda+κ)*eye(n);
```

Table 3.7 shows results comparing the time taken to construct a Lyapunov function using the standard dense approach (SeDuMi) and the chordal sparsity approach (SparseCoLO+SeDuMi) for the class of matrices described above. We find that the chordal sparsity approach is significantly faster than the standard approach for this type of system. This can be explained by the fact that the largest matrix blocks in the chordal sparsity SDP are relatively small (20-40 nodes) compared to the size of the matrix block in the standard dense SDP which are of size n. This gives the chordal sparsity approach an advantage because SeDuMi can exploit this block diagonal sparsity pattern to speed up the calculation of the Newton step within the interior-point method.

For our next set of experiments we tested the performance of the Chordal Power algorithm on sparse, random systems. In this case the graph describing the sparsity pattern of $A^T P + PA$ is not necessarily close to being chordal and so may

maxC =10	SeDuMi				SparseCoLO+SeDuMi				
n	mean	std	dim.Sch	max.Bl	mean	std	dim.Sch	max.Bl	av. k
100	6.0	2.3	260	100	1.6	1.2	2594	26	0.6
200	37.1	18.0	499	200	5.9	4.2	2936	26	0.7
300	73.0	34.9	514	300	6.5	6.1	4675	25	0.4
400	233.1	87.2	982	400	23.3	13.1	6306	41	0.8
500	313.1	168.7	1072	500	76.6	96.1	9182	48	0.6

Table 3.7: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time (in seconds) to find Lyapunov functions for sparse chordal A matrices with maximum cliques bounded by 10 plus sparse random entries, with $\kappa = 0.1$.

require a significant number of edges be added in order to extend it to be chordal. We therefore expect the algorithm to perform less well on these examples. The systems used in our experiments were generated in the following way: Given n (`n`) and ρ (`rho`) we generated sparse random A matrices in MATLAB using the following code:

```
S = sprand(n,n,rho) + eye(n);

S = abs(S)>0;

A = rand(n,n)-0.5;

A = A.*S;

lambda = max(real(eig(A)));

A = A -(lambda+κ)*eye(n);
```

where `sprand(n,n,rho)` is a random, n-by-n, sparse matrix with approximately ρn^2 uniformly distributed nonzero entries.

In Table 3.8 and Table 3.9 we show the total time required to run the Chordal Power algorithm on these randomly generated examples for $\rho = 0.1/n$ and $\rho =$

$\rho = 0.1/n$	SeDuMi				SparseCoLO+SeDuMi				
n	mean	std	dim.Sch	max.Bl	mean	std	dim.Sch	max.Bl	av.k
100	5.7	2.3	267	100	1.7	1.0	2105	25	0.65
200	32.9	13.0	488	200	5.9	4.1	3185	27	0.65
300	90.0	39.2	647	300	11.3	10.6	5018	32	0.55
400	177.1	78.6	898	400	20.5	17.3	7011	37	0.6

Table 3.8: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time (in seconds) to find Lyapunov functions for sparse random A matrices with $\rho = 0.1/n$ and $\kappa = 0.1$.

$0.2/n$ respectively. We see that for $\rho = 1/n$ the chordal sparsity approach is on average significantly faster than the standard dense approach. However, when the density of the random matrices is doubled to $\rho = 0.2/n$, the chordal sparsity approach performs worse than the standard dense approach as n grows large. This is due to the fact that increasing the density of the system matrix also tends to increase the size of the maximal cliques in the decomposed problem. These experiments therefore point to the importance of examining the sparsity pattern of a given problem after the chordal extension process has been carried out before deciding whether to apply the chordal sparsity approach or not. One must consider the size of the Schur complement matrix along with its sparsity pattern in order to estimate the time required to complete an iteration of the Newton step algorithm.

3.3 Discrete-Time Lyapunov Stability

In this section we consider a discrete-time system $x_{k+1} = Ax_k$ where $A \in \mathbb{R}^{n\times n}$ is assumed to be a sparse matrix. The stability of a discrete-time system can be verified by constructing a Lyapunov function that satisfies the following require-

$\rho = 0.2/n$	SeDuMi				SparseCoLO+SeDuMi				
n	mean	std	dim.Sch	max.Bl	mean	std	dim.Sch	max.Bl	av.k
100	5.5	2.1	240	100	2.1	1.6	2148	26	0.55
200	33.6	9.1	581	200	16.4	11.4	4300	43	0.8
300	91.4	38.0	729	300	63.6	56.6	5614	51	0.65
400	235.1	131.0	990	400	293.5	360.9	8146	68	0.7

Table 3.9: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time (in seconds) to find Lyapunov functions for sparse random A matrices with $\rho = 0.2/n$ and $\kappa = 0.1$.

ments

$$V(x_k) > 0,\ \forall x_k \in \mathbb{R}^n \setminus \{0\}$$

$$V(x_{k+1}) - V(x_k) < 0,\ \forall x_k, x_{k+1} \in \mathbb{R}^n.$$

For a quadratic Lyapunov function $V(x_k) = x_k P x_k$, $P \succ 0$, the conditions in (3.3) reduce to

$$P \succ 0,\ Q = A^T P A - P \prec 0. \tag{3.16}$$

It is well-known that the problem of checking the robust stability of discrete-time system with polytopic uncertainty can be formulated as an LMI and solved using an SDP. This result is stated in the following theorem.

Theorem 3.3.1 Given matrices $A_1, A_2, \ldots, A_m \in \mathbb{R}^{n\times n}$, if there exists $P \succ 0$ such that

$$A_i^T P A_i - P \prec 0, \quad i = 1, 2, \ldots, m \tag{3.17}$$

then $x_{k+1} = Ax_k$, $A \in \text{conv}(A_1, A_2, \ldots, A_m)$ is asymptotically stable.

For robust stability analysis the size of the semidefinite constraint in the SDP grows as nm, and hence it might be useful to decompose this using the chordal sparsity method.

We next consider how to exploit chordal sparsity when constructing Lyapunov functions for discrete-time systems. The following theorem by Balakrishnan *et. al.* will be very useful for this purpose.

Theorem 3.3.2 [62] If G is chordal and k is odd, then G^k is chordal.

This theorem can be applied to the discrete-time problem in the following way. Given $A \in \mathbb{R}^{n\times n}$, let $|A|$ be the $n \times n$ matrix with elements equal to the absolute values of the elements of A, i.e., $|A|_{ij} = |A_{ij}|$. Let $G(A) = (V, E)$ be the undirected graph with edge set

$$E = \{(i, j) \mid (|A| + |A^T|)_{ij} \neq 0\}.$$

Let $G_{ch}(A) = (V, E)$ be a chordal extension of A, then by Theorem 3.3.2, we have that $G^3_{ch}(A) = (V, H)$ is a chordal graph. Let $P \in \mathbb{S}^n(E, 0)$ and $G(Q) = (V, F)$ where $Q = A^T PA - P$, then $F \subset H$. Hence we can chordal extend $G(Q)$ to $G^3_{ch}(A)$. We next propose a discrete-time version of the Chordal Power algorithm that uses this property of odd powers of chordal graphs so that only a single chordal extension is required in the process of searching for a Lyapunov function.

Chordal Powers Algorithm: Given $A \in \mathbb{R}^{n\times n}$. Pick maximum clique threshold $M > 0$. Initialise $m = 0,\ k = 1$.

$S := (A^T + A),\ \ G_{ch}(S) = (V, H)$

$T \in \mathbb{S}^n(H, 0),\ \ T_{ij} \in \{0, 1\}$

while $m < M$

$G(P_k) := G(T^k) = (V, E_k)$

$G(Q_k) := G(T^{2+k}) = (V, F_k)$

solve SDP:

$$\begin{bmatrix} P_k & 0 \\ 0 & Q_k \end{bmatrix} \succeq \epsilon I$$

$Q_k = -(A^T P_k A - P_k)$

$P_k \in \mathbb{S}^n_+(E_k, 0),\ \ Q_k \in \mathbb{S}^n_+(F_k, 0)$

if SDP is feasible

Terminate: A Lyapunov function has been found

else

$m := \mathrm{maxC}(G(Q_k))$

$k := k + 2$

end

end

To test the Chordal Powers algorithm in the discrete-time case we generated sparse

maxC= 10	SeDuMi				SparseCoLO+SeDuMi				
n	mean	std	dim.Sch	max.Bl	mean	std	dim.Sch	max.Bl	k
100	2.9	0.4	100	100	1.0	0.1	1261	23	0
200	15.4	0.9	200	200	2.0	0.7	3072	27	0
300	44.0	2.6	300	300	4.0	0.7	3253	31	0
400	90.1	6.3	400	400	7.9	1.4	4687	39	0
500	172.2	13.3	500	500	13.8	6.5	5756	45	0

Table 3.10: SeDuMi CPU time vs SparseCoLO+SeDuMi CPU time in seconds to find Lyapunov functions for sparse chordal A matrices with random noise for discrete time case with $\kappa = 1$.

random A matrices using the same approach as in the continuous-time case i.e., generating chordal sparsity pattern and adding a small number of random edges. To ensure stability, we modified the spectrum of the matrix to lie within the unit circle by dividing the entries of the matrix by the factor $(\lambda + \kappa)$ where λ is the absolute value of the largest eigenvalue in A. Table 3.10 summarises the results for instances with where the maximal cliques of $G(A)$ were bounded by 10, $\rho = 0.1/n$ and $\kappa = 1$.

In our experiments we found that as the value of κ is reduced towards zero the chordal powers algorithm tends to require a large number of iterations, meaning that the SDP becomes increasingly dense to the point where solving the SDP using SparseCoLO requires more time than simply solving it without exploiting chordal sparsity. This negative result is disappointing, but for large sparse examples we expect that the benefits of exploiting sparsity will begin to dominate again, but this will remain a topic for future research.

3.4 The KYP Lemma

In this section we consider how to use the methods described in the previous section to exploit chordal sparsity in the KYP lemma. The KYP lemma is a central result in robust control theory that gives an equivalence between the feasibility of an LMI or Riccati equation and a bound on the infinity norm of the transfer function matrix. The SDP that must be solved to compute this bound are typically of large dimension and so methods for exploiting sparsity are of particular interest.

Consider the continuous-time system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{3.18}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$ and $D \in \mathbb{R}^{p\times m}$. The KYP lemma allows us to compute a bound on the worst-case gain of the system from the input $u(t)$ to the output $y(t)$, which is useful in many applications where the input represents a disturbance that is norm-bounded. We next state the KYP lemma.

Theorem 3.4.1 [14] Let $G(s) = C(sI-A)^{-1}B+D$ be the transfer function matrix for the state-space system in (3.18). The following are equivalent conditions.
i) The eigenvalues of A satisfy $\mathrm{Re}(\lambda_i) < 0$ for $i = 1, 2, \ldots, n$ and $\|G\|_\infty < \gamma$.
ii) There exists a matrix $P \succ 0$ such that

$$\begin{bmatrix} A^TP + PA + C^TC & PB + C^TD \\ B^TP + D^TC & D^TD - \gamma^2 I \end{bmatrix} \prec 0.$$

To exploit chordal sparsity it is necessary to reformulate the LMI using the Schur complement formula in order to give the following optimisation problem

$$\begin{aligned} &\text{minimise} \quad \alpha \\ &\text{subject to} \quad P \succ 0, \qquad M(P) = \begin{bmatrix} A^TP + PA & PB & C^T \\ B^TP & -\alpha I & D^T \\ C & D & -I \end{bmatrix} \prec 0, \end{aligned} \tag{3.19}$$

where $\alpha = \gamma^2$. By reformulating the problem in this way, we see that $Q = A^TP + PA$ is now in the upper left block of $M(P)$, and so we can directly apply the methods proposed in the previous section on continuous-time Lyapunov stability to exploit chordal sparsity for this block. On the other hand, we must now consider the sparsity pattern of the matrix M in the inequality, this is summarised in the following result.

Theorem 3.4.2 Let M be as defined in (3.19) and let $Q = A^TP + PA$. Suppose that $G(Q)$ has r maximal cliques and the cardinality of the largest maximal clique is equal to d. Then the worst-case number of maximal cliques in $G(M)$ is mpr and the cardinality of the largest maximal clique in $G(M)$ is $d + 2$.

Proof 3.4.1 Consider the worst-case scenario where B, C, D are dense matrices. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$ be the set of maximal cliques of $G(Q)$ and suppose that d is the cardinality of the largest maximal clique in $\mathcal{C}$. Let $G(I)$ be the graph corresponding to the sparsity pattern of the $-\alpha I$ block in M, which consists of m nodes without connections between the nodes. We combine $G(Q)$ and $G(I)$ together to form a new graph $G(J) = G(Q) \cup G(I)$. The connections between the nodes in the two components $G(Q)$ and $G(I)$ are determined by the sparsity

pattern of the PB block. When PB is a dense block, every node in $G(Q)$ is connected to every node in $G(I)$, which implies that the maximal cliques of $G(J)$ are given by

$$C_i \cup j, \;\; i \in \{1, 2, \ldots, r\}, \; j \in \{n+1, n+2, \ldots, n+m\}.$$

Hence in the worst case, the number of maximal cliques in $G(J)$ is mr and each maximal clique in $G(J)$ has cardinality less than or equal to $d+1$. By repeating the same procedure for dense C and D we see that the worst case number of maximal cliques in $G(M)$ is mpr and the cardinality of the largest maximal clique in $G(M)$ is $d+2$, which completes the proof. $\square$

To test the chordal decomposition method on the continuous-time KYP lemma we first generated sparse, random, stable systems using the method described in Appendix A.3.4. We then solved the SDP in (3.19) in two different ways. First we started with a diagonal P matrix and solved the optimisation problem using SparseCoLO. If the result returned by the solver was infeasible, we then increased the bandwidth of P and repeated the process until the problem became feasible. This gave us the value α_1 which is the bound on the H_∞ norm of the system achievable for this sparse Lyapunov function. Then to allow us to compare the chordal sparsity approach with the standard approach, we set P to be a fully dense matrix and solved the optimisation problem using SeDuMi to obtain a second bound α_2.

Table 3.11 summarises the results from the continuous-time KYP experiments. We see that the bound computed using the sparse approach is on average a few percent greater than the bound computed using a dense matrix and is orders of

Continuous-time KYP results ($m = p = 10$, $\rho = 3/n$)									
n	20	30	40	50	60	70	80	90	100
%	4.0	3.7	2.6	2.5	1.8	2.4	0.9	1.8	0.9
σ	(3.5)	(3.8)	(2.3)	(4.0)	(2.7)	(4.6)	(1.2)	(3.4)	(1.6)
t_1	0.7	1.0	2.6	3.8	5.0	6.4	8.3	11.0	14.1
σ_1	(0.3)	(0.4)	(0.8)	(0.7)	(0.8)	(1.2)	(1.4)	(2.0)	(2.3)
t_2	1.6	3.0	6.6	13.0	29.7	59.8	127.4	281.1	675.4
σ_2	(0.7)	(0.7)	(1.0)	(2.0)	(4.4)	(9.6)	(19.4)	(81.6)	(138.4)

Table 3.11: Results for the continuous-time KYP lemma experiments. The first row, marked %, gives the average percentage increase in the norm bound between using the chordal sparsity approach and the standard dense approach. The second row marked σ gives the standard deviation of the percentage increase. The variable t_1 is the time taken to solve the sparse optimisation problem in SeDuMi after pre-processing using SpareCoLO, and t_2 is the average time taken in seconds to solve the standard dense optimisation problem using SeDuMi.

magnitude quicker for the largest examples.

We now carry out the same analysis for the discrete-time KYP lemma. Consider the following LTI discrete-time system

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k, \end{aligned} \tag{3.20}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, and $D \in \mathbb{R}^{p\times m}$. The analogue to the continuous-time KYP lemma in discrete-time is stated in the following theorem.

Theorem 3.4.3 Suppose $G(z) = C(zI-A)^{-1}B+D$ is the transfer function matrix for the state-space system in (3.20). The following are equivalent conditions.

i) The eigenvalues of A satisfy $|\lambda_i| < 1$ for $i = 1, 2, \ldots, n$ and $\|G\|_\infty < \gamma$.

ii) There exists a matrix $P \succ 0$ such that

$$\begin{bmatrix} A^TPA - P + C^TC & A^TPB + C^TD \\ B^TPA + D^TC & B^TPB + D^TD - \gamma^2 I \end{bmatrix} \prec 0.$$

Similarly to the continuous-times case, we apply the Schur complement to expand the inequality in Theorem 3.4.3 and restate it in terms of the following optimisation problem

$$\begin{aligned} &\text{minimise} \quad \alpha \\ &\text{subject to} \quad P \succ 0, \qquad M(P) = \begin{bmatrix} A^TPA - P & A^TPB & C^T \\ B^TPA & B^TPB - \gamma^2 I & D^T \\ C & D & -I \end{bmatrix} \prec 0. \end{aligned} \tag{3.21}$$

By reformulating Theorem 3.4.3 in this way we see that $Q = A^TPA - P$ is now in the upper left block of $M(P)$, and so we can directly apply the methods proposed in the previous section on discrete-time Lyapunov stability to exploit chordal sparsity for this block. On the other hand, we must now consider the sparsity pattern of the matrix M in the inequality. Once again we will consider the worst case scenario where B, C, D are dense and we shall see that the situation is less favourable in the discrete-time case than in the continuous-time case. This is summarised in the following result (the proof of which follows from identical arguments as in the continuous-time case).

Theorem 3.4.4 Let $M(P)$ be as defined in (3.21) and let $Q = A^TPA - P$. Suppose that $G(Q)$ has r maximal cliques and the cardinality of the largest maximal clique is equal to d. Then the worst-case number of maximal cliques in $G(M)$ is

pr and the cardinality of the largest maximal clique in $G(M)$ is $d + m + 1$.

3.5 Chapter Summary

We have shown that when the matrices describing an LTI dynamical system are sparse we can exploit results on chordal sparsity to decompose the semidefinite constraints in the SDPs that arise in linear control theory. This makes it a promising approach when applied to large, sparse systems which are often found in engineering applications. We also draw attention to the fact that that the techniques discussed in this chapter can also be applied to other problems in linear control theory such as checking whether a given LTI system is dissipative or passive.

Chapter 4

Sparse State Feedback Synthesis

4.1 Introduction

The design of decentralised feedback controllers for interconnected systems has been an active area of research in recent times [63, 64, 65, 66]. Some examples of problems involving interconnected systems include the control of power networks [67], formation flying of UAVs [68], and vibration control of large structures [69, 70]. In the static linear feedback case, decentralised control corresponds to a sparse feedback gain matrix, and in this chapter we consider methods of exploiting the properties of chordal graphs when designing sparse static state feedback controllers.

Approaches to the design of sparse controllers can be divided according to whether the controller structure is allowed vary, or whether the controller is assumed to have a fixed sparsity pattern. The choice of which of these two approaches is taken is typically determined by the stage at which the controller is being added to the

system, with the fixed sparsity pattern case corresponding to adding a controller to a pre-existing plant, and the the variable sparsity case corresponding to an early design phase. The two approaches are also often used in conjunction, since after a suitable structure for the controller has been found, we may then carry out a second optimisation process with this structure fixed in order to fine tune the parameters.

In the variable structure case, sparsity in the controller is typically introduced by optimising a performance objective along with some additional regularisation term on the cardinality of the controller. For example, in [71], the authors first design a sparse controller for an LTI system without constraints on the controller structure by solving an optimal control problem with an additional L_1 norm regularisation term. By varying the weight of the regularisation term they search for an acceptable trade off between sparsity and performance.

The problem synthesising controllers with a-priori structural constraints is known to be an NP-hard problem in general [72], and hence attention has been focussed on characterising when the problem is computationally tractable [73]. Approaches to the general problem include finding approximate solutions via convex relaxations [75, 76], or using non-convex optimisation methods such as the alternating direction method of multipliers [71].

In this chapter we follow the well known approach to controller design of first specifying the the performance specification in terms of a Lyapunov function, and then, via an appropriate transformation of the controller variables, reducing the

problem to a LMI [77]. We propose two methods of exploiting the properties of chordal graphs when designing sparse static state feedback controllers for linear time-invariant systems. In the first half of the chapter, we apply a result called the maximum-determinant completion theorem to synthesise controllers with a fixed sparsity pattern. In the second half of the chapter, we allow the controller structure to vary and consider how to exploit chordal sparsity within the problem of minimising the H_∞ norm of a system.

4.1.1 Background

In this subsection, we first summarise some well known results on static state feedback synthesis, and then state a theorem from [41] called the maximum-determinant completion theorem that will made use of in the following subsection. We focus on the state equation

$$\dot{x}(t) = Ax(t) + Bu(t), \;\; x(0) = x_0 \tag{4.1}$$

where $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$. Under a static state feedback control law, $u(t) = Kx(t)$ with $K \in \mathbb{R}^{m\times n}$, the state space equation (4.1) reduces to the autonomous system

$$\dot{x}(t) = (A + BK)x(t), \;\; x(0) = x_0.$$

It is well known that the point $x = 0$ is asymptotically stable if and only if there exists a function of the form $V(x) = x^T Px$ that satisfies the Lyapunov conditions

in Theorem 3.2.1). These conditions are equivalent to

$$P \succ 0, \;\; (A+BK)^T P + P(A+BK) \prec 0. \tag{4.2}$$

Let $P = Q^{-1}$ and $R = KQ$. A well known necessary and sufficient condition for (4.2) to be feasible is

$$\begin{bmatrix} -Q & 0 \\ 0 & QA^T + R^T B^T + AQ + BR \end{bmatrix} \prec 0, \tag{4.3}$$

and the controller can be recovered from $K = RQ^{-1}$.

It is often desirable to design decentralised controllers that operate independently based on locally available information. One reason for this is that in many cases it is impractical for a centralised controller to have access to all of the states of a system and so decentralisation becomes a requirement in the design process. This motivates our interest in the following theorem that links sparse inverses of positive definite matrices to chordal graphs.

Theorem 4.1.1 [41]. Suppose $X \in \mathbb{S}^n_{++}(E, ?)$ is a partial positive matrix with a chordal sparsity pattern. Then there exists a unique matrix W in the set of all positive definite completions of X that maximises the determinant function, i.e.

$$\begin{aligned} W = \text{argmax} \quad & \det M \\ \text{such that} \quad & M_{ij} = X_{ij}, \; \forall (i,j) \in E \\ & M \succ 0. \end{aligned} \tag{4.4}$$

Furthermore, the inverse of W satisfies $[W^{-1}]_{ij} = 0$ if $(i, j) \notin E$.

In other words, this theorem states that the inverse of the maximum determinant completion of a partial positive matrix with a chordal sparsity pattern inherits the sparsity pattern of the partial positive matrix. We note that the constraints in the maximum-determinant completion problem are equivalent to an LMI, and that if we replace the objective function of maximising the determinant of M with the equivalent objective function of maximising the log determinant of M we convert this problem into a convex optimisation problem [10].

For general sparsity patterns, the maximum-determinant completion problem must be solved using iterative methods, but for chordal sparsity patterns the solution can be calculated via linear algebra using what is known as the sparse clique-factorisation formula [36]. By using the clique-factorisation formula we can efficiently complete a partial positive matrix to one with a sparse inverse. This makes this theorem a relevant tool for constructing matrices with sparse inverses in order to design sparse state feedback controllers. We next introduce some new notation before stating the clique-factorisation formula.

For any given $X \in \mathbb{S}^n(E, 0)$ we may set the zero entries to be unspecified entries in order to obtain a partial matrix $\hat{X} \in \mathbb{S}^n(E, ?)$. Or in the other direction, we may fill the unspecified entries of a partial matrix with zeros to obtain a sparse matrix. We use the double arrow notation $X \leftrightarrow \hat{X}$ to denote this transformation between partial and sparse matrices. For the sake of simplicity, we will also use the phrase 'the maximum-determinant completion of X' to mean the maximum-determinant

completion of $\hat{X}$. In addition, for every pair (S, T), where $S, T \subset V$, we use the notation X_{ST} for the submatrix formed from all rows $i \in S$ and all columns $j \in T$. We are now ready to state the clique-factorisation formula.

Theorem 4.1.2 [36] Let $G = (V, E)$ be a chordal graph with set of maximal cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$, where the maximal cliques are assumed to be ordered such that they satisfy the running intersection property. Let $\hat{X} \in \mathbb{S}^n_{++}(E, ?)$ and apply the transformation $\hat{X} \leftrightarrow X$. Let P be a permutation matrix that has been chosen so that $(1, 2, \ldots, n)$ is a perfect elimination ordering for the graph $G(PXP^T)$. Then the maximum-determinant completion of X can be expressed in terms of the sparse clique-factorisation formula

$$PWP^T = L_1^T L_2^T \cdots L_{p-1}^T D L_{p-1} \cdots L_2 L_1, \tag{4.5}$$

where $L_1, L_2, \ldots, L_{p-1}$ are lower triangular matrices and D is a positive definite block-diagonal matrix consisting of p diagonal blocks. More explicitly, let

$$S_r = C_r \setminus (C_{r+1} \cup C_{r+2} \cup \cdots \cup C_p) \text{ for } r = 1, 2, \ldots, p$$
$$U_r = C_r \cap (C_{r+1} \cup C_{r+2} \cup \cdots \cup C_p) \text{ for } r = 1, 2, \ldots, p.$$

Then

$$[L_r]_{ij} = \begin{cases} 1 & \text{if } i = j \\ (X^{-1}_{U_r U_r} X_{U_r S_r})_{ij} & \text{if } i \in U_r \text{ and } j \in S_r \\ 0 & \text{otherwise} \end{cases}$$

for $r = 1, 2, \ldots, p-1$, and

$$D = \begin{bmatrix} D_{S_1S_1} & & & \\ & D_{S_2S_2} & & \\ & & \ddots & \\ & & & D_{S_pS_p} \end{bmatrix}$$

with

$$D_{S_rS_r} = \begin{cases} X_{S_rS_r} - X_{S_rU_r}X_{U_rU_r}^{-1}X_{U_rS_r} & \text{for } r = 1, 2, \ldots, p-1 \\ X_{S_rS_r} & \text{for } r = p. \end{cases}$$

4.1.2 Maximum determinant completions and the S-Procedure

We now further discuss the relevance of Theorem 4.1.1 and Theorem 4.1.2 to the control problem of designing sparse feedback controllers and develop a novel approach based on the S-procedure. Let $G = (V, E)$ be a chordal graph and suppose that $Q \in \mathbb{S}^n_{++}(E, 0)$ and $R \in \mathbb{R}(F, 0)$ satisfy (4.3), where $\mathbb{R}(F, 0)$ is the set of $\mathbb{R}^{m\times n}$ matrices with sparsity pattern given by some set of indices F. Now transform $Q \in \mathbb{S}^n_{++}(E, 0)$ into a partial positive matrix $\hat{Q} \in \mathbb{S}^n(E, ?)$. This transformation is valid since every principle submatrix of a positive definite matrix is positive definite, and hence each submatrix of Q that corresponds to a maximal clique of $G = (V, E)$ must be positive definite. Now let W be the maximum determinant completion of Q. In the light of Theorem 4.1.1 it is of interest to ask whether W and R satisfy

$$WA^T + AW + BR + R^TB^T \prec 0, \tag{4.6}$$

since if W and R satisfy this inequality the sparse feedback matrix $K = RW^{-1}$ stabilises the system. Unfortunately W and R will not be feasible in general, and so this motivates us to find ways to modify the LMI to improve the likelihood of this being the case.

The general approach that we will take is to exchange robustness to perturbations of the system matrix A for robustness to perturbations of Q. More explicitly, let $A + \Delta A$ be a perturbed system with $\Delta A \in \Omega$ where $\Omega \subset \mathbb{R}^{n \times n}$ and suppose that Q and R are such that

$$Q \succ 0,\ Q(A + \Delta A)^T + (A + \Delta A)Q + BR + R^T B^T \prec 0,\ \Delta A \in \Omega. \tag{4.7}$$

We may write the maximum-determinant completion of Q as $W = Q + \Delta Q$ with $\Delta Q \in \mathbb{S}^n(\overline{E}, 0)$, where $\overline{E} = \{(i,j) \in V \times V \mid (i,j) \notin E\}$ is the complement of E. Consider the following equation which equates perturbations in A to perturbations in Q

$$Q(A + \Delta A)^T + (A + \Delta A)Q = (Q + \Delta Q)A^T + A(Q + \Delta Q). \tag{4.8}$$

If there exists a matrix $\Delta A \in \Omega$ such that (4.8) is satisfied we can conclude that the maximum-determinant completion of Q is a feasible point and recover the sparse state-feedback controller $K = RW^{-1}$.

Next we use the S-procedure to formulate an optimisation problem in which we maximise the set of possible perturbation in A that can be tolerated for a given

sparsity pattern of Q. Consider the following system with a nonlinear perturbation

$$\dot{x}(t) = Ax(t) + h(x(t)) + Bu(t), \quad h(x(t)) \in \Omega_\alpha$$

where Ω_α is the set of perturbations

$$\Omega_\alpha = \{h : \mathbb{R}^n \to \mathbb{R}^n \mid h^T h \leq \alpha^2 x^T H^T H x\}.$$

This set of perturbations can be characterised by the following inequality

$$\begin{bmatrix} x & h \end{bmatrix} \begin{bmatrix} -\alpha^2 H^T H & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x \\ h \end{bmatrix} \leq 0.$$

We now apply state feedback $u = Kx$ to the system so that the dynamics are given by

$$\dot{x}(t) = (A + BK)x(t) + h(x(t)), \quad h(x(t)) \in \Omega_\alpha.$$

To check the stability of this closed-loop system we consider the candidate Lyapunov function $V(x) = x^T P x$ with $P \succ 0$. The derivative condition can then be written as

$$\dot{V}(x) = \begin{bmatrix} x & h \end{bmatrix} \begin{bmatrix} A^T P + PA + PBK + K^T B^T P & P \\ P & 0 \end{bmatrix} \begin{bmatrix} x \\ h \end{bmatrix} \leq 0.$$

The S-Procedure is known to be lossless in the case of one quadratic objective function subject to one quadratic constraint. Let $\tau > 0$, the following is a necessary and sufficient condition for the closed loop system to be stable for all perturbations

$h \in \Omega_\alpha$:

$$
\begin{aligned}
&P \succ 0 \\
&\begin{bmatrix} A^TP + PA + PBK + K^TB^TP + \tau\alpha^2 H^TH & P \\ P & -\tau I \end{bmatrix} \prec 0.
\end{aligned}
$$

Let $Q = \tau P^{-1}$, then, using the Schur complement, this is equivalent to

$$
\begin{aligned}
&Q \succ 0 \\
&\begin{bmatrix} QA^T + AQ + BKQ + QK^TB^T & I & QH^T \\ I & -I & 0 \\ HQ & 0 & -\gamma I \end{bmatrix} \prec 0,
\end{aligned}
$$

where $\gamma = 1/\alpha^2$. This inequality can then be reformulated as an LMI by the change of variable $KQ = R$. The largest possible set of perturbations that can be tolerated can be found by minimising γ, resulting in the following SDP

$$
\begin{aligned}
&\text{minimise} \quad \gamma \\
&\text{subject to} \quad Q \in \mathbb{S}^n_{++}(E, 0), \;\; R \in \mathbb{R}^{n\times n}(F, 0) \\
&\qquad \begin{bmatrix} QA^T + AQ + BR + R^TB^T & I & QH^T \\ I & -I & 0 \\ HQ & 0 & -\gamma I \end{bmatrix} \prec 0.
\end{aligned}
$$

Solving this SDP therefore returns a controller $K = RQ^{-1}$ that stabilises the system for all perturbations in the set Ω_α. This is a desirable objective because maximising the magnitude of perturbations that can be tolerated increases the likelihood that the maximum determinant completion of Q is feasible in (4.6).

We make one final modification to the optimisation problem above to ensure that the magnitude of the elements in the feedback gain matrix K are suitably normalised using a method proposed by Siljak *et. al.* [78]. Consider the constraints on Q and R

$$R^T R \prec \beta_R I,\ \beta_R > 0$$

$$Q^{-1} \prec \beta_Q I,\ \beta_Q > 0,$$

which are equivalent to the LMIs

$$\begin{bmatrix} -\beta_R I & R^T \\ R & -I \end{bmatrix} \prec 0, \qquad \begin{bmatrix} \beta_Q I & I \\ I & Q \end{bmatrix} \succ 0.$$

These LMIs provide the bound

$$K^T K = Q^{-1} R^T R Q^{-1} \prec \beta_R Q^{-1} Q^{-1} \prec \beta_R \beta_Q^2 I,$$

and hence can be used to normalise the magnitude of the entries in K. Using these constraints we may formulate the problem as maximising the robustness of the closed-loop system to perturbations given an upper bound on the magnitude of feedback control gains that we are willing to tolerate. Hence we pick β_Q and β_R

to be suitable so that $\beta_R\beta_Q^2$ is acceptable and solve the optimisation problem

$$
\begin{aligned}
&\text{minimise} \quad \gamma \\
&\text{subject to} \quad Q \in \mathbb{S}^n_{++}(E,0), \quad R \in \mathbb{R}^{m\times n}(F,0) \\
&\qquad \begin{bmatrix} -(QA^T + AQ + BR + R^TB^T) & -I & -QH^T \\ -I & I & 0 \\ -HQ & 0 & \beta I \end{bmatrix} \succ 0 \qquad (4.9) \\
&\qquad \begin{bmatrix} \beta_R I & -R^T \\ -R & I \end{bmatrix} \succ 0, \qquad \begin{bmatrix} \beta_Q I & I \\ I & Q \end{bmatrix} \succ 0.
\end{aligned}
$$

4.1.3 Numerical Results

The objective of the previous section was to develop a method of synthesising sparse feedback controllers using the maximum determinant completion property. In order to maximise the likelihood of returning a sparse feedback controller we modified the feasibility problem in (4.3) by adding an objective function to give the optimisation problem in (4.9). In this subsection we give a comparison between this modified problem and the standard approach of solving the feasibility problem. To ensure a like-for-like comparison we first adjoin some additional constraints to the feasibility problem in (4.3) to give:

$$
\begin{aligned}
&Q \in \mathbb{S}^n_{++}(E,0), \quad R \in \mathbb{R}^{m\times n}(F,0) \\
&QA^T + AQ + BR + R^TB^T \prec 0 \\
&\begin{bmatrix} \beta_R I & -R^T \\ -R & I \end{bmatrix} \succ 0, \qquad \begin{bmatrix} \beta_Q I & I \\ I & Q \end{bmatrix} \succ 0.
\end{aligned} \qquad (4.10)
$$

By setting the parameters β_Q and β_R to be equal in (4.9) and (4.10) we ensure that the magnitudes of the feedback gain matrices are constrained in the same way in both problems. From now on we will refer to (4.9) as the Modified Problem and (4.10) as the Standard Problem. For a given system (A, B) and choice of sparsity pattern (E, F), and assuming that the problems are feasible, we denote the solutions to the Modified and Standard Problems by (Q_1, R_1) and (Q_2, R_2) respectively.

We next give a description of how the experiments were carried out. For each instance we first generated a random matrix $A \in \mathbb{R}^{n\times n}(E, 0)$ with a chordal sparsity pattern (with largest maximal clique restricted to 4 nodes). We computed the maximal cliques of $G(A)$ and m maximal cliques were chosen at random to define the sparsity pattern of $B \in \mathbb{R}^{n\times m}(F, 0)$. We normalised A so that the largest eigenvalue was of magnitude one and shifted the spectrum according to $A + \delta I$, where $\delta \in \mathbb{R}$ is a parameter such that $\max_i \mathrm{Re}(\lambda_i(A)) = \kappa > 0$. Then, for simplicity, we set $H = I$ and the sparsity patterns of the matrix variables were selected to match the sparsity patterns of the randomly generated system, i.e., $Q \in \mathbb{S}^n_{++}(E, 0)$ and $R \in \mathbb{R}^{n\times m}(F, 0)$.

Each random instance of the Modified and Standard Problems was then solved using SparseCoLO and the resulting Q_1 and Q_2 were completed to their respective maximum determinant completions W_1 and W_2 using the clique factorisation formula (4.5). We then compared the proportion of controllers $K_1 = R_1 W_1^{-1}$ and $K_2 = R_2 W_2^{-1}$ that stabilised the closed loop system. The objective being to demonstrate the improvement in the success rate of designing a stabilising controller that results from maximising the set of perturbations that can be tolerated.

The success rates for returning sparse stabilising feedback controllers and the 95% confidence intervals for the two optimisation problems are shown in Table 4.1 (instances in which (4.10) and (4.9) were infeasible were discarded). We see that for randomly generated systems of this type the Modified approach provides a significant improvement in the success rate of synthesising a sparse stabilising state feedback controller. In particular, for examples involving $n = 10$ nodes the Modified approach improves the success rate by approximately a factor of two compared to the Standard approach. For the examples involving $n \geq 20$ states the Modified approach maintains a high success rate whilst the Standard approach does not perform as well. We can also see that increasing κ tends to reduce the probability of successfully synthesising sparse state feedback controllers. In addition, in Appendix A.4 we give an example in which the sparsity patterns were chosen to give a banded feedback matrix.

4.2 H_∞ State Feedback Synthesis

In this section we consider a method of exploiting chordal sparsity when designing a static state feedback controller to minimise the H_∞ norm of a system from a disturbance to an output. The approach that we will take is to restrict the structure of the Q matrix to be diagonal and decompose the problem using chordal sparsity. Although this method is conservative it will enable us to synthesise state feedback controllers for systems with larger dimensions than is possible using the standard dense method.

		n				
κ	Problem	10	20	30	40	50
0.1	Modified	0.92 ± 0.06	0.90 ± 0.07	0.91 ± 0.07	0.78 ± 0.10	0.84 ± 0.08
0.1	Standard	0.40 ± 0.10	0.15 ± 0.09	0.08 ± 0.06	0.03 ± 0.03	0.05 ± 0.05
0.15	Modified	0.92 ± 0.04	0.85 ± 0.06	0.83 ± 0.06	0.78 ± 0.07	0.74 ± 0.07
0.15	Standard	0.43 ± 0.08	0.12 ± 0.06	0.10 ± 0.05	0.04 ± 0.03	0.02 ± 0.02
0.2	Modified	0.89 ± 0.05	0.84 ± 0.06	0.81 ± 0.07	0.77 ± 0.08	0.75 ± 0.09
0.2	Standard	0.46 ± 0.08	0.09 ± 0.05	0.05 ± 0.04	0.03 ± 0.03	0.02 ± 0.03
0.25	Modified	0.90 ± 0.05	0.81 ± 0.07	0.79 ± 0.08	0.71 ± 0.09	0.74 ± 0.10
0.25	Standard	0.41 ± 0.09	0.06 ± 0.04	0.04 ± 0.04	0.04 ± 0.04	0.02 ± 0.02
0.3	Modified	0.89 ± 0.06	0.76 ± 0.09	0.69 ± 0.10	0.56 ± 0.12	0.68 ± 0.11
0.3	Standard	0.44 ± 0.09	0.07 ± 0.05	0.05 ± 0.05	0.05 ± 0.05	0.03 ± 0.03

Table 4.1: Success rates and 95% confidence intervals for the Modified and Standard approaches to synthesising sparse feedback controllers for varying n and κ.

Consider the state space system

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + Fw(t) \\ y(t) &= Cx(t) + Du(t) + Hw(t), \end{aligned} \tag{4.11}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the input, $w(t) \in \mathbb{R}^q$ is a disturbance and $y(t) \in \mathbb{R}^p$ is the output. In order to exploit chordal sparsity we will first focus on the simpler state equation

$$\dot{x}(t) = Ax(t) + Bu(t),$$

with the assumption that A and B are sparse matrices and that the system is stabilisable. We wish to represent the sparsity pattern of this system using an

undirected graph that takes into account the sparsity patterns of both A and B and to facilitate this we require some further definitions.

Given $A \in \mathbb{R}^{n\times n}$, let $|A|$ be the $n \times n$ matrix with elements equal to the absolute values of the elements of A, i.e., $|A|_{ij} = |A_{ij}|$. The sparsity pattern graph of A, denoted by $G(A) = (V, E)$ is the graph with set of nodes $V = \{1, 2, \ldots, n\}$ and edge set

$$E = \{(i, j) \mid (|A| + |A^T|)_{ij} \neq 0\}.$$

Let $I_k = \{j \in V \mid B_{jk} \neq 0\}$ for $k = 1, 2, \ldots, m$, so that each set I_k represents the subset of the nodes that are directly influenced by input $u_k(t)$. We are now ready to define a graph that we will use to summarise the sparsity of the system.

Let $G_{ch}(A, B)$ be a chordal extension of $G(A)$ with the additional constraint that

$$\exists C_i \in \mathcal{C} \text{ such that } I_k \subset C_i \text{ for } k = 1, 2, \ldots, m,$$

where $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$ is the set of maximal cliques of $G_{ch}(A, B)$. This constraint can be enforced by taking $G(A) = (V, E)$ and extending the edge set according to

$$E' = \cup_{k=1}^{m} (I_k \times I_k) \cup E$$

and then finding a chordal extension of this augmented graph. Each maximal clique of $G_{ch}(A, B)$ corresponds to a subset of the states of the system and we interpret each maximal clique as a subsystem of the large sparse system. We next give a simple example of a sparse system to help explain the purpose of the

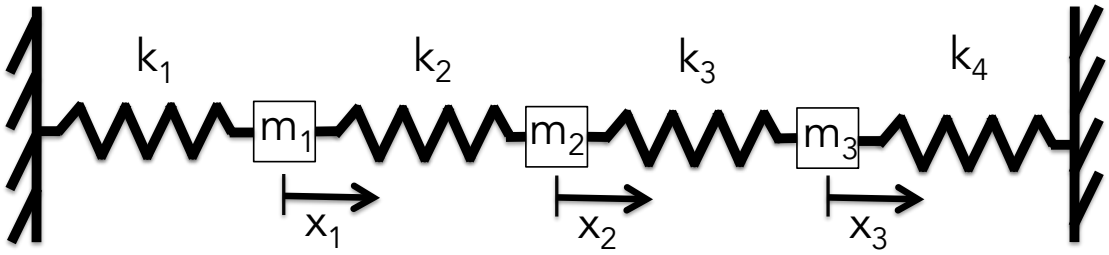


Figure 4.1: A mass-spring system consisting of three masses.

definitions above.

Example 4.2.1 Consider the mass-spring system shown in Figure 4.1 where each mass m_i is equipped with an actuator that can apply a force $u_i(t)$. The state-space representation for this system is

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \ddot{x}_1(t) \\ \ddot{x}_2(t) \\ \ddot{x}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ t_{11} & t_{12} & 0 & 0 & 0 & 0 \\ t_{21} & t_{22} & t_{23} & 0 & 0 & 0 \\ 0 & t_{32} & t_{33} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/m_1 & 0 & 0 \\ 0 & 1/m_2 & 0 \\ 0 & 0 & 1/m_3 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix}$$

where $t_{11} = -(k_1 + k_2)/m_1,\ t_{12} = k_2/m_1, t_{21} = k_1/m_2,\ t_{22} = -(k_2 + k_3)/m_2,\ t_{23} = k_3/m_2, t_{32} = k_{3/m_3},\ t_{33} = -(k_3 + k_4)/m_3$.

Figure 4.2 shows a diagram of $G(A)$ which has the following set of maximal cliques $\mathcal{C} = \{\{1,4\},\{1,5\},\{2,4\},\{2,5\},\{2,6\},\{3,5\},\{3,6\}\}$. The set of indices for the inputs are $I_1 = \{4\}$, $I_2 = \{5\}$ and $I_3 = \{6\}$, which are all subsets of the maximal

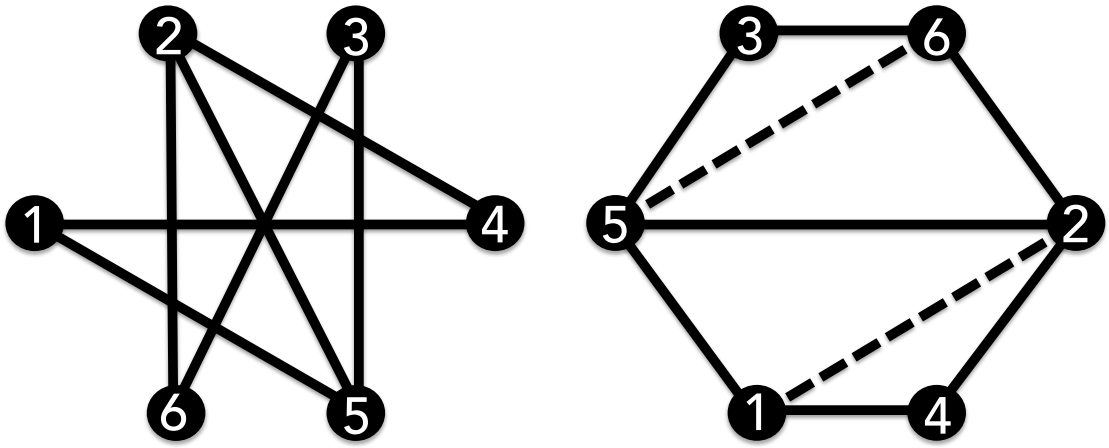


Figure 4.2: $G(A)$ and a $G_{ch}(A, B)$ for the mass-spring example.

cliques of $G(A)$ and so no further edges must be added to account for the sparsity pattern of B. However, we see that $G(A)$ is not a chordal graph, and hence we extend the graph to make it chordal using the edges $(1, 2)$ and $(5, 6)$ (shown in dotted lines), as shown on the right of Figure 4.2.

We now return to the full state space system in (4.11) and consider how to formulate the H_∞ synthesis problem so that the sparsity pattern described by $G_{ch}(A, B)$ can be exploited. Under the control law $u(t) = Kx(t)$ the state and output equations for the system become

$$\begin{aligned} \dot{x}(t) &= A_{cl}x(t) + Fw(t) \\ y(t) &= C_{cl}x(t) + Hw(t), \end{aligned}$$

where $A_{cl} = A + BK$ and $C_{cl} = C + DK$. Then using the KYP lemma we may formulate the problem of minimising the H_∞ norm using a static state feedback

controller as the following optimisation problem

$$\begin{aligned} &\text{minimise} \quad \alpha \\ &\text{subject to} \quad P \succ 0 \\ &\qquad M(P,K) = \begin{bmatrix} (A+BK)^T P + P(A+BK) & PF & (C+DK)^T \\ F^T P & -\alpha I & H^T \\ C+DK & H & -I \end{bmatrix} \prec 0. \end{aligned}$$

However, in its current form the problem is not an SDP since the matrix variable $M(P,K)$ does not depend affinely on P and K. Therefore we need to find a transformation of variables to convert this into a convex optimisation problem. Focussing on $M(P,K)$ and setting $P = Q^{-1}$ we have

$$\begin{bmatrix} A^T Q^{-1} + K^T B^T Q^{-1} + Q^{-1} A + Q^{-1} BK & Q^{-1} F & C^T + K^T D^T \\ F^T Q^{-1} & -\alpha I & H^T \\ C+DK & H & -I \end{bmatrix} \prec 0.$$

Multiplying this matrix inequality left and right by the positive definite matrix $\mathrm{diag}(Q, I_q, I_p)$, where I_q and I_p are identity matrices of dimension q and p respectively, we obtain

$$\begin{bmatrix} QA^T + QK^T B^T + AQ + BKQ & F & QC^T + QK^T D^T \\ F^T & -\alpha I & H^T \\ CQ + DKQ & H & -I \end{bmatrix} \prec 0,$$

After expanding terms and setting $R = KQ$ we arrive at a form that is affine in Q and R and so the problem of minimising the H_∞ norm can be expressed as the

following SDP

$$\begin{aligned}
&\text{minimise} \quad \alpha \\
&\text{subject to} \quad Q \succ 0 \\
&\begin{bmatrix} QA^T + R^TB^T + AQ + BR & F & QC^T + R^TD^T \\ F^T & -\alpha I & H^T \\ CQ + DR & H & -I \end{bmatrix} \prec 0 \qquad (4.12) \\
&\begin{bmatrix} \beta_R I & -R^T \\ -R & I \end{bmatrix} \succ 0, \qquad \begin{bmatrix} \beta_Q I & I \\ I & Q \end{bmatrix} \succ 0,
\end{aligned}$$

where we have introduced some extra constraints on the magnitude of the elements in K. Next we state a result that will be useful for decomposing the top left matrix block in this SDP, and will guide our choice of sparsity patterns for R.

Proposition 4.2.1 Let $A \in \mathbb{R}^{n\times n}$ and let $\mathcal{C} = \{C_1, C_2 \ldots, C_p\}$ be the set of maximal cliques of $G(A)$. We can factor the sparsity pattern as $G(A) = G(XX^T)$ where $X \in \mathbb{R}^{n\times p}$ is given by

$$X_{ij} = \begin{cases} 1 & \text{if } i \in C_j \\ 0 & \text{otherwise.} \end{cases}$$

Proof 4.2.1 Let $Y = XX^T$. The ij element of Y is given by

$$Y_{ij} = \left(XX^T\right)_{ij} = \sum_{k=1}^{p} X_{ik}(X^T)_{kj} = \sum_{k=1}^{p} X_{ik}X_{jk}.$$

Let $G(Y) = (V, F)$ be the sparsity pattern graph of Y. We have that

$$(i,j) \in F \Leftrightarrow Y_{ij} \neq 0 \Leftrightarrow \exists k \text{ s.t. } i,j \in C_k \Leftrightarrow (i,j) \in E.$$

Hence $F = E$ and $G(XX^T) = G(A)$. $\square$

We next explain the relevance of this proposition to the sparsity pattern of the matrix variables in the state feedback LMI. Suppose that we restrict the sparsity pattern of Q to be diagonal, this ensures that $QA^T + AQ$ has the same sparsity pattern as $A^T + A$. Suppose that we sample at random m maximal cliques $C_1, C_2, \ldots, C_m$ from $G_{ch}(A, B)$, and let $R \in \mathbb{R}^{m \times n}$ have the sparsity pattern defined by

$$R_{ij} = \begin{cases} r_{ij} & \text{if } j \in C_i \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

where the r_{ij}s are variables that can be optimised. Using this choice of sparsity pattern for R and Proposition 4.2.1 the graph $G(S)$ of the matrix $S = BR + R^T B^T$ is covered by $G_{ch}(A, B)$ and hence we can exploit chordal sparsity in the top left block of the LMI in (4.12). Equally importantly, with Q constrained to be diagonal, the state feedback matrix $K = RQ^{-1}$ has the same sparsity pattern as R.

However, having a fixed sparsity pattern for R can be too limiting since (4.12) may not be feasible for that specific sparsity pattern. In order to generalise this approach we find a sequence of R matrices of increasing density i.e., R^k for $k = 0, 1, 2, \ldots, q$, that define a sequence of increasingly dense SDPs (note that in this case the superscript in R^k is not intended to denote a power). This approach is useful when we wish to strike a balance between decentralisation and performance.

In order to find such a sequence of R matrices we will use an operation on the maximal cliques of the graph $G_{ch}(A, B)$ which we will denote 'merging cliques' [79]. The idea is to combine two maximal cliques in order to obtain a single, larger maximal clique. We will show that this type of sequence allows us to place an upper bound on the number of edges that must be added by any chordal extension.

Lemma 4.2.1 [79] Suppose that $G^0 = (V, E)$ is a chordal graph and let C_i and C_j be any maximal cliques of G^0 such that $i \neq j$ and $C_i \cap C_j \neq \emptyset$. Now define a new edge set $E \cup F$ where

$$F = \{(i, j) \mid i, j \in C_i \cup C_j\}.$$

Then the graph $G_1 = (V, E \cup F)$ is chordal.

Proof 4.2.2 G_0 is chordal and hence there exists a clique tree that satisfies the clique intersection property. Note that the merging operation combines two adjacent nodes in the clique tree to form a new clique tree that also satisfies the clique intersection property. Therefore there exists a clique tree that satisfies the clique intersection property for G_1 and hence G_1 is a chordal graph. □

A sequence of merges γ can be represented by a sequence of pairs of indices $\gamma = [(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)]$ that define which maximal cliques are merged together. Suppose that G^0 is a chordal graph with set of maximal cliques $\mathcal{C}^0 = \{C_1^0, C_2^0, \ldots, C_m^0\}$ and a sequence of merges γ of length $k < m$. The sequence γ defines a mapping from C^0 to another set of maximal cliques $\mathcal{C}^k = \{C_1^k, C_2^k, \ldots, C_{m-k}^k\}$. Note that each $C_i^0 \in \mathcal{C}^0$ will satisfy $C_i^0 \subseteq C_l^k$ for a unique $C_l^k \in \mathcal{C}^k$ and we use this to define

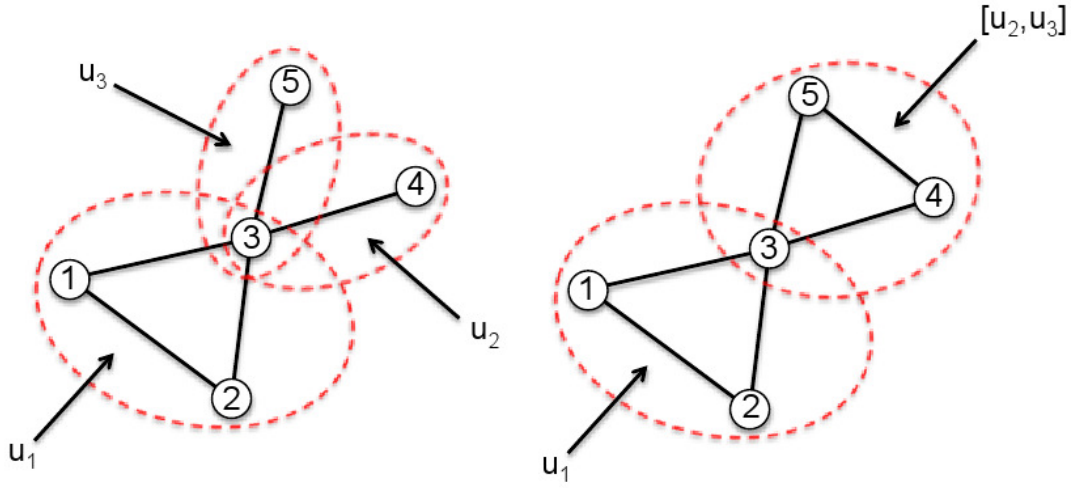


Figure 4.3: Merging two maximal cliques in a graph

$R^k \in \mathbb{R}^{m \times n}$ as

$$R^k_{ij} = \begin{cases} r_{ij} & \text{if } \exists l \text{ such that } j \in C^k_l \text{ and } C^0_i \subseteq C^k_l \\ 0 & \text{otherwise.} \end{cases} \tag{4.14}$$

Example 4.2.2 Consider the graph shown on the left of Figure 4.3. The graph is chordal and has set of maximal cliques

$$\mathcal{C}^0 = \{\{1,2,3\},\{3,4\},\{3,5\}\}.$$

The sparsity pattern for R^0 is then given by

$$R^0 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Now merge cliques C^0_2 and C^0_3 together, as shown on the right of Figure 4.3, so

that $\mathcal{C}^1 = \{\{1, 2, 3, \}, \{3, 4, 5\}\}$. This gives

$$R^1 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

By now merging C_1^1 and C_2^1 we obtain a single maximal clique, i.e., $\mathcal{C}^2 = \{1, 2, 3, 4, 5\}$, and R^2 is a fully dense matrix.

Proposition 4.2.2 Given $G_{ch}(A) = (V, E^0)$ with set of maximal cliques $\mathcal{C}^0 = \{C_1^0, C_2^0, \ldots, C_m^0\}$. Suppose that γ is a sequence of merges of length $k < m$ that generates the set of maximal cliques $\mathcal{C}^k = \{C_1^k, C_2^k, \ldots, C_{m-k}^k\}$. Assume that Q has a diagonal structure and that R^k is as defined as in (4.14) and let $Y = QA^T + (R^k)^T B^T + AQ + BR^k$. The number of edges ζ that must be added in any chordal extension of $G(Y)$ satisfies $\zeta \leq |E^k \setminus E^0|$, where

$$E^k = \{(i, j) \,|i, j \in C_l^k \text{ for some } l \in \{1, 2, \ldots, m - k\}\}.$$

Proof 4.2.3 Let $G^k = (V, E^k)$ be the graph defined by

$$(i, j) \in E^k \Leftrightarrow i, j \in C_l^k$$

for some $l \in \{1, 2, \ldots, m - k\}$. Since $G^k = (V, E^k)$ was formed by a sequence of merges of the maximal cliques of $G_{ch}(A)$, by Lemma 4.2.1 it is also a chordal graph and we have that $E^0 \subset E^k$. Hence we may always extend $G(Y)$ to equal G^k and so we have an upper bound on the number of new variables that must be added to the LMI. $\square$

Suppose that we are given a set of maximal cliques $\mathcal{C}^k = \{C_1^k, C_2^k, \ldots, C_{m-k}^k\}$ that was generated under the assumptions in Proposition 4.2.2. In order to keep the controller implementation local and minimise the number of free variables in the LMI we use the following heuristic: find the smallest maximal clique in the set; then find the smallest maximal clique adjacent to this maximal clique, and merge them together. The motivation for this algorithm being to minimise the number of edges that we are adding to the graph of the sparsity pattern of the LMI by merging together small maximal cliques.

4.2.1 Numerical Results

We next demonstrate the effectiveness of this approach on a number of examples. First we consider the sparse system consisting of 10 states defined by the set of matrices (A, B, C, D, F, H) given in Appendix A.5. For these experiments we initialised R so as to have the same sparsity pattern as B^T, and then we solved the SDP in (4.12) for the sequence of R matrices produced by merging maximal cliques according to the greedy algorithm. In each experiment the parameters were set as $\beta_R = \beta_Q = 50$. The results are shown in Figure 4.4, where we have plotted heat maps of the first eight K matrices, the scale to the right of each figure shows the size of the elements in each K matrix.

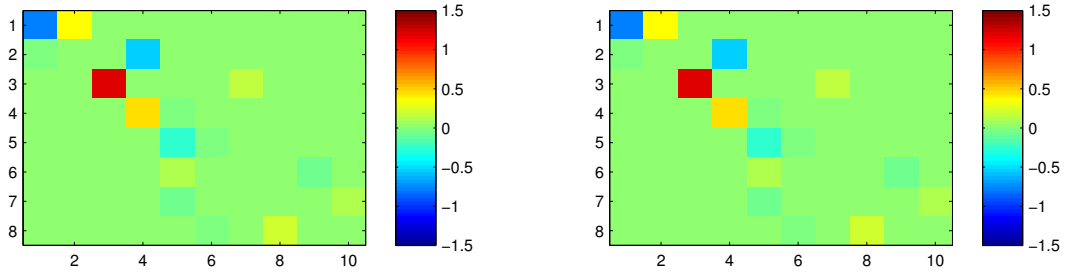
The numerical results of the experiments are collected in Table 4.2. For each experiment $\text{nnz}(K)$ is the number of non-zero entries in K; γ is the bound on the H_∞ norm of the system under the controller $K = RQ^{-1}$; t is the CPU time in seconds to solve the SDP using SparseCoLO; $\dim(A)$ is the dimension of the coefficient

matrix in the SDP; m is the number of semidefinite cones in the SDP, and maxC is the size of the largest maximal clique/SDP cone in the SDP.

K_i	nnz(K)	γ	t	dim(A)	m	maxC
1	8	0.3571	1.8	19 x 292	27	14
2	17	0.3281	0.6	32 x 431	24	14
3	24	0.3274	0.9	37 x 384	24	14
4	29	0.3274	1.1	41 x 439	23	14
5	35	0.3273	1.2	53 x 456	24	14
6	42	0.3271	1.7	60 x 500	24	16
7	50	0.3269	1.1	62 x 511	23	16
8	59	0.3263	1.4	80 x 520	23	15
9	65	0.3263	1.7	76 x 570	22	18
10	70	0.3263	2.8	81 x 570	22	18
11	74	0.3263	1.1	85 x 570	22	18
12	77	0.3263	1.8	88 x 570	22	18
13	79	0.3263	1.8	90 x 570	22	18
14	80	0.3263	2.6	91 x 570	22	18
Full	80	0.3240	1.1	145 x 777	13	18

Table 4.2: 10 Node Example. $K_1, K_2, \ldots, K_{14}$ are controllers that were synthesised by solving (4.12) for a sequence of increasingly dense R matrices generated via the maximal clique merging heuristic. The final row of the table, marked 'Full' displays the result of solving the SDP with Q and R set to be fully dense matrices, this shows how the performance of the sparse controllers compares to the best achievable performance.

Next we consider a larger example consisting of 50 states, repeating the experiment using the same methodology as in the example with 10 states. In Table 4.3 we see that when the density of the controller is increased, the performance improves, and that by the 15th step of the algorithm the performance of the sparse controller is within 10% of the best possible performance achievable using a dense controller. Finally, in Table 4.4 we show the results for a 100 state example to demonstrate the scalability of the approach. We observe that each step of the algorithm requires a



(a) K_1 (b) K_2

(c) K_3 (d) K_4

(e) K_5 (f) K_6

(g) K_7 (h) K_8

Figure 4.4: Heatmaps $K_1, K_2, \ldots, K_8$ from the 10 Node example (Table 4.2) showing the increasing density of the controllers.

few tens of seconds to solve for these sparse controllers, whereas 100 state example was too large for us to solve for a controller using a dense Q and R matrix.

K_i	nnz(K)	γ	t	dim(A)	m	maxC
1	94	0.6008	7.1	180 x 5929	117	69
2	97	0.5953	7.9	181 x 6151	115	69
3	101	0.5952	6.7	185 x 6071	115	69
4	106	0.5929	7.9	205 x 6187	115	69
5	112	0.5928	8.4	210 x 6098	114	69
6	116	0.5926	7.3	212 x 6284	112	69
7	123	0.5924	6.9	205 x 6189	113	69
8	131	0.5918	11.3	195 x 6242	115	70
9	134	0.5815	7.9	265 x 5930	118	67
10	136	0.5787	7.8	266 x 5977	117	67
11	138	0.5783	7.6	277 x 6018	117	67
12	140	0.5777	9.6	283 x 6060	116	67
13	142	0.5766	7.9	280 x 6037	116	67
14	144	0.5763	6.6	266 x 6130	115	68
15	153	0.5760	7.8	230 x 6253	114	70
Full	2250	0.5199	217.0	3654 x 19155	55	95

Table 4.3: 50 node example.$K_1, K_2, \ldots, K_{15}$ are controllers that were synthesised by solving (4.12) for a sequence of increasingly dense R matrices generated via the maximal clique merging heuristic. The final row of the table, marked 'Full' displays the result of solving the SDP with Q and R set to be fully dense matrices, this shows how the performance of the sparse controllers compares to the best achievable performance.

4.3 Chapter Summary

In the first half of this chapter we considered how the sparse inverse property of chordal partial matrices could be used to design static state feedback controllers with desired sparsity patterns. The approach taken was to solve the associated SDP with sparse Q and R matrices whilst maximising the set of uncertain systems that the resulting controller could stabilise. By maximising the robustness of the

K_i	nnz(K)	γ	t	dim(A)	m	maxC
1	197	1.1397	28.3	402 x 22196	229	139
2	199	1.1396	32.0	406 x 22062	231	139
3	202	1.1295	23.2	408 x 22133	230	139
4	206	1.1267	22.4	411 x 22328	229	139
5	211	1.1091	15.5	337 x 22461	227	140
6	217	1.1071	16.2	343 x 22389	227	140
7	224	1.1049	24.0	437 x 22390	228	139
8	232	1.1040	23.0	444 x 22230	231	139
9	241	1.0938	29.4	484 x 22285	231	139
10	243	1.0934	22.4	486 x 22325	231	139
11	245	1.0895	24.0	487 x 22340	230	139
12	247	1.0894	20.4	491 x 22254	232	139
13	250	1.0886	21.8	493 x 22402	229	139
14	254	1.0881	19.9	495 x 22540	227	139
15	259	1.0865	17.1	504 x 22404	231	139
Full	4500	-	-	-	-	-

Table 4.4: 100 node example. $K_1, K_2, \ldots, K_{15}$ are controllers that were synthesised by solving (4.12) for a sequence of increasingly dense R matrices generated via the maximal clique merging heuristic. The final row of the table, marked 'Full' displays the result of solving the SDP with Q and R set to be fully dense matrices, in this case the fully dense problem was too large to solve numerically.

closed-loop system we improved the likelihood that the maximum determinant completion of Q is a feasible solution to the SDP.

In the second half of this chapter we decomposed the SDP associated with the H_∞ static state feedback synthesis problem using the chordal decomposition method. We proposed a sequence of R matrices that provided an upper bound on the number of free variables that must be added to the SDP. This allows the computational difficulty of the problem to be increased incrementally so that a sequence of LMIs can be tested for feasibility.

Chapter 5

Stability Analysis of Sparse Nonlinear Systems

5.1 Introduction

In chapter 3 we considered how to exploit chordal sparsity in the SDPs that arise when constructing Lyapunov functions for linear systems. The steps in this process were: describe the sparsity pattern of the matrix variable in the SDP in terms of a graph, find a chordal extension of this graph, and then decompose the SDP according to the maximal cliques of the extended graph. This process allowed a large, sparse matrix variable to be decomposed into multiple smaller matrix variables with some additional equality constraints.

In this chapter we focus on exploiting chordal sparsity when constructing Lyapunov functions using Sum of Squares (SOS) polynomials. For large SOS problems the scale of the matrix variables in the SDP relaxation means that the first step of

describing the sparsity pattern of the matrix variable is rendered computationally intractable if carried out directly. The main contribution in this chapter is a method that circumvents this problem by exploiting chordal sparsity without explicitly forming the matrix variable in the SDP by using the notion of correlative sparsity and Agler's theorem.

The notion of correlative sparsity was originally introduced by Waki *et. al.* in [16] where the authors proposed a method of exploiting sparsity in polynomial optimisation problems. The approach taken by Waki involved expressing the sparsity pattern of a polynomial in terms of a graph, chordal extending this graph and then using the maximal cliques of the extended graph to propose support sets of polynomials. This leads to a sparse SOS problem that can be converted into a primal SDP problem and then be decomposed using Grone's theorem. In this chapter, we show that the sparse SOS problem proposed by Waki can be more directly decomposed by applying Agler's theorem to the dual SDP problem.

We first give some background on SOS polynomials and explain how this theory can be applied to the problem of constructing Lyapunov functions for systems with polynomials vector fields. We then consider systems where the dynamics are given by sparse polynomials and develop a method for efficiently constructing Lyapunov functions for this class of systems using chordal sparsity.

5.2 Sum of Squares Polynomials

Let $x \in \mathbb{R}^n$, $\alpha \in \mathbb{N}^n$ and let $x^\alpha = \prod_{i=1}^n x_i^{\alpha_i}$ denote a monomial in x of degree $|\alpha| = \sum_{i=1}^n \alpha_i$. For an integer $d \in \mathbb{N}$, let $\mathbb{N}_d^n = \{\alpha \in \mathbb{N}^n | \sum_{i=1}^n \alpha_i \leq d\}$ and let

$$v_d(x) := \left(1, x_1, x_2, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_{n-1}x_n, x_n^2, \ldots, x_1^d, \ldots, x_n^d\right)^T \tag{5.1}$$

be the vector of all monomials of degree less than or equal to d with dimension $s(d) = \binom{n+d}{d}$. A (real) polynomial $p(x)$ is a linear combination of a finite set of monomials of x with coefficients in $\mathbb{R}$:

$$p(x) = \sum_\alpha p_\alpha x^\alpha = \sum_\alpha p_\alpha x_1^{\alpha_1} x_2^{\alpha_2}, \cdots, x_n^{\alpha_n}, \quad p_\alpha \in \mathbb{R}.$$

The degree of a polynomial is the maximum over the degrees of all the monomials in it. We denote the ring of multivariate polynomials with real coefficients by $\mathbb{R}[x]$. Given a polynomial $p \in \mathbb{R}[x]$ of degree d we can uniquely identify p with the vector of coefficients $\mathbf{p} = \{p_\alpha\} \in \mathbb{R}^{s(d)}$ and write it as

$$p(x) = \sum_{\alpha \in \mathbb{N}_d^n} p_\alpha x^\alpha = \langle \mathbf{p}, v_d(x) \rangle.$$

The support of a polynomial is defined by

$$\text{supp}(p) = \{\alpha \in \mathbb{N}^n \mid p_\alpha \neq 0\}.$$

In many applications it is of interest to know whether a given polyomial $p(x)$ is nonnegative, that is $p(x) \geq 0$ for all $x \in \mathbb{R}^n$. For example, as we shall see later in

this chapter, the problem of constructing a Lyapunov function for systems with polynomial vector fields can be reduced to finding a polynomial that satisfies a number of nonnegativity conditions.

Checking whether a given polynomial is nonnegative is an NP-hard problem for polynomials of degree greater than or equal to four [80]. We therefore cannot expect to discover scalable algorithms that solve this problem in general. Faced with this problem, researchers have instead sought relaxations of nonnegativity constraints that are more computationally tractable, such as the Sum of Squares relaxation.

The basic concept of the SOS relaxation is to take a nonnegativity constraint on a polynomial and replace it with the constraint that the polynomial must be expressible as a sum of squares of other polynomials. This motivates the following definition of SOS polynomials.

Definition 5.2.1 (Sum of Squares Polynomial): A polynomial $p \in \mathbb{R}[x]$ of degree $2d$ is a sum of squares (SOS) if there exist polynomials $q_i \in \mathbb{R}[x]$, $i = 1, 2, \ldots, m$, such that

$$p(x) = \sum_{i=1}^{m} q_i^2(x).$$

We write the set of SOS polynomials as $\Sigma[x]$ and note that expressing a polynomial as a sum of squares provides a certificate or proof that the polynomial is nonnegative.

The next question that arises is whether membership to the set of SOS polynomials

can be checked in polynomial time. A crucial step to answering this question is to first recognise that a polynomial is an SOS polynomial if and only if it can be expressed as a quadratic form involving a semidefinite matrix.

Theorem 5.2.1 [17] A polynomial $p \in \mathbb{R}[x]$ of degree $2d$ has a sum of squares decomposition (or is SOS) if and only if there exists a real symmetric and positive semidefinite matrix $Q \in \mathbb{S}_+^{s(d)}$ such that $p(x) = v_d(x)^T Q v_d(x)$ for all $x \in \mathbb{R}^n$.

Parrilo realised that this characterisation of SOS polynomials allows us to test whether a given polynomial is a sum of squares polynomial by solving an SDP [81]. More specifically, suppose that $p \in \mathbb{R}[x]$ is a polynomial of degree $2d$ that we wish to test for membership of $\Sigma[x]$, then for each $\alpha \in \mathbb{N}_{2d}^n$ we define the matrices $B_\alpha \in \mathbb{S}^{s(d)}$ by the equation

$$v_d(x)v_d(x)^T = \sum_{\alpha \in \mathbb{N}_{2d}^n} B_\alpha x^\alpha.$$

We may now formulate the problem of checking whether $p(x)$ is an SOS polynomial as the following SDP feasibility problem: Find $Q \in \mathbb{S}^{s(d)}$ such that

$$\begin{aligned} &Q \bullet B_\alpha = p_\alpha, \ \forall \alpha \in \mathbb{N}_{2d}^n \\ &Q \succeq 0. \end{aligned} \tag{5.2}$$

Therefore the decision problem of checking whether a given polynomial can be represented as a sum of squares can be solved in polynomial time using interior-point methods [82]. However, in practice, since the dimension of the matrix variables in the SDP is given by $s(d) = \binom{n+d}{d}$ the memory required to store the problem and the time required to compute a solution becomes impractical for large n, d unless

some structure or sparsity within the problem can be exploited.

One approach to mitigate this scaling issue is to reformulate the problem so that it involves matching the value of the polynomial at a number of sample points rather than matching coefficients of the polynomial. This approach leads to rank one coefficient matrices in the linear constraints of the SDP which can be exploited to improve the efficiency with which the gradient and Hessian of the logarithmic barrier function is computed [83].

Another possible approach is to focus on the sparsity of the polynomials to reduce the dimension of the matrix variable. Work in this direction includes the Newton Polytope method [84], sparse Lagrange multipliers [85, 86] and correlative sparsity [16]. In this chapter we will consider the notions of correlative sparsity and chordal graphs to decompose the semidefinite constraints found in SOS problems without having to explicitly form the Q matrix. This is achieved by constructing the maximal cliques of a chordal graph that is a chordal extension of the graph describing the sparsity pattern of Q.

We next give a brief example of one of the applications of SOS polynomials. Consider the following global optimisation problem:

$$\text{minimise } \; p(x) \; \text{ s.t. } \; x \in \mathbb{R}^n,$$

where $p(x)$ is a multivariate polynomial of even degree. By introducing a free variable γ we may express this minimisation problem as the following maximisation

problem with a nonnegativity constraint

$$\text{maximise} \quad \gamma \quad \text{subject to} \quad r(x,\gamma) = p(x) - \gamma \geq 0, \quad \forall x \in \mathbb{R}^n. \tag{5.3}$$

We may relax the nonnegativity constraint in (5.3) by constraining $r(x,\gamma)$ to be expressible as a sum of squares [87]

$$\text{maximise} \quad \gamma \quad \text{subject to} \quad r(x,\gamma) = p(x) - \gamma \in \Sigma[x].$$

Since by definition an SOS polynomial is nonnegative the constraint above is sufficient for $r(x,\gamma)$ to be nonnegative and by maximising γ we place a bound on the global minimum of $p(x)$.

Sum of Squares polynomials also play an important role in a central result in real algebraic geometry called the Positivstellensatz. This result gives an equivalence between the emptiness of a set defined in terms of a collection of polynomial equalities and inequalities and the existence of a solution to an algebraic condition. A sparse version of the Positivstellensatz with a direct connection to chordal graphs via the running intersection property was derived by Lasserre [57].

5.3 Polynomial Lyapunov Functions

We next focus our attention on dynamical systems with polynomial vector fields and consider how SOS polynomials can be applied to stability analysis problems.

Consider the following continuous-time system with a polynomial vector field

$$\dot{x} = f(x),\ f(0) = 0,$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is of the form $f(x) = [f_1(x), f_2(x), \ldots, f_n(x)]^T$ with $f_1, f_2, \ldots, f_n \in \mathbb{R}[x]$. For ease of reference we restate Lyapunov's theorem which was given in Chapter 3. Suppose that $g_1, g_2, \ldots, g_m \in \mathbb{R}[x]$ define a basic semi-algebraic set

$$D = \{\, x \in \mathbb{R}^n \mid g_j(x) \geq 0,\ j = 1, 2, \ldots, m \,\}$$

that contains the origin. Then if there exists a continuously differentiable function $V : D \to \mathbb{R}$ such that

$$\begin{aligned} &1)\ V(x) > 0 \text{ for all } x \in D \setminus \{0\} \text{ and } V(0) = 0, \\ &2)\ -\dot{V}(x) = \nabla V(x)^T f(x) \geq 0 \text{ for all } x \in D, \end{aligned} \tag{5.4}$$

then the origin is locally stable. To formulate the positive definite condition above as an SOS problem we first define the positive definite function $\varphi(x)$ as

$$\varphi(x) = \sum_{i=1}^{n} \sum_{j=1}^{d} \epsilon_{ij} x_i^{2j},\ \ \sum_{j=1}^{d} \epsilon_{ij} \geq \gamma,\ i = 1, 2, \ldots, n,\ \ \gamma > 0,\ \ \epsilon_{ij} \geq 0\ \forall i, j.$$

This makes $\varphi(x) > 0$, i.e., is positive definite. If we impose $V(x) - \varphi(x)$ to be SOS we have that

$$V(x) - \varphi(x) \geq 0 \Rightarrow V(x) \geq \varphi(x) > 0,$$

i.e., the SOS condition that ensures that $V(x)$ is positive definite. We may then write the conditions for local stability as [88]

$$\begin{aligned} V(x) - \sum_{j=1}^{m} q_j(x) g_j(x) - \varphi(x) &\in \Sigma[x] \\ -\nabla V(x)^T f(x) - \sum_{j=1}^{m} r_j(x) g_j(x) &\in \Sigma[x] \\ q_j(x) &\in \Sigma[x], \quad j = 1, 2, \ldots, m \\ r_j(x) &\in \Sigma[x], \quad j = 1, 2, \ldots, m. \end{aligned} \tag{5.5}$$

In principle this problem can now be formulated as an SDP and solved in polynomial time. However, in practice the dimension of the semidefinite constraints and the number of free variables in the problem makes constructing Lyapunov functions in this way prohibitively costly in terms of memory and computation time with currently available methods. This motivates us to consider systems with sparse polynomial vector fields in order to establish whether we can exploit chordal sparsity in these cases.

5.4 Correlative Sparsity and Sparse Lyapunov Functions

Given a polynomial $p \in \mathbb{R}[x]$ in n variables, the correlative sparsity pattern matrix of p is defined as the $n \times n$ symmetric matrix where the i, jth entry is nonzero if p contains monomials involving both x_i and x_j terms and zero otherwise. A polynomial is then said to be correlatively sparse if the associated correlative sparsity pattern matrix is sparse [16]. The definition of correlative sparsity implies a higher level view of the sparsity of a polynomial in terms of the interaction of independent

variables rather than of the microstructure of individual monomials. We will use correlative sparsity to define what it means for a dynamical system with polynomial dynamics to be sparse, and as a tool to design candidate Lyapunov functions to ensure that the resulting SDPs are sparse.

Waki *et. al.* first introduced the concept of correlative sparsity in the context of sparse polynomial optimisation problems involving SOS polynomials. When a polynomial objective function is correlatively sparse it is possible to derive a sparse SDP relaxation to the optimisation problem. These sparse SDP relaxations are derived from the graph (or the chordal extension of the graph) describing the correlative sparsity pattern by using the maximal cliques of the graph to define the sets of supports used for the SOS polynomials in the relaxed problem.

Let $f(x) = [f_1(x), f_2(x), \ldots, f_n(x)]^T$ describe the system dynamics and $\{g_1(x), \ldots, g_m(x)\}$ define D in (5.5). Let $\mathcal{S}$ be the set of monomials in the support of either $f_1(x), f_2, \ldots, f_n(x)$ or $g_1(x), \ldots, g_m$, i.e.,

$$\mathcal{S} = \{\alpha \in \mathbb{N}^n \mid \exists k \text{ s.t. } \alpha \in \text{supp}(f_k) \text{ or } \exists j \text{ s.t. } \alpha \in \text{supp}(g_j)\}.$$

Using this set of monomials we define the correlative sparsity pattern matrix for the local stability problem as

$$R_{ij} = \begin{cases} \star & \text{if } i = j \\ \star & \text{if } \alpha_i, \alpha_j \geq 1 \text{ for some } \alpha \in \mathcal{S}, \\ 0 & \text{otherwise} \end{cases}$$

where the symbol $\star$ denotes a nonzero entry in the matrix. We then say that a problem is correlatively sparse if the correlative sparsity pattern matrix is sparse. We next consider how to structure a candidate Lyapunov function in order to preserve this correlative sparsity in the conditions in (5.5), and hence ensure that the resulting SDP is sparse.

Let $G(R)$ be the graph describing the sparsity pattern of the correlative sparsity pattern matrix of $f(x)$ and D and let $\Sigma[x, \mathcal{C}]$ be the set of SOS polynomials with support given by the set of maximal cliques of $G(R)$. Just as in the linear case, at the one extreme we have diagonal Lyapunov functions and at the other we have fully dense Lyapunov functions. An intermediate choice is to set $V(x)$ to be a polynomial with the same correlative sparsity pattern matrix as R_{ch}, i.e., pick a candidate Lyapunov function of the form

$$V(x) = \sum_{i=1}^{p} V_i(x)$$

where each $V_i(x) \in \mathbb{R}[x, C_i]$ for $i = 1, 2, \ldots, p$. By introducing this structure into the Lyapunov function the conditions for local asymptotic stability become

$$\begin{aligned} \sum_{i=1}^{p} V_i(x) - \sum_{k=1}^{m} q_k(x) g_k(x) - \varphi(x) \in \Sigma[x, \mathcal{C}] \\ -\left(\sum_{i=1}^{p} \nabla V_i(x) \right)^T f(x) - \sum_{k=1}^{m} r_k(x) g_k(x) \in \Sigma[x, \mathcal{C}'] \end{aligned}$$

where $q_k \in \Sigma[x, C_k]$ for $k = 1, \ldots, p$ and $r_k \in \Sigma[x, C'_k]$ for $k = 1, \ldots, s$.

Next we give a simple example of a correlatively sparse system and a structured Lyapunov function that preserves the correlative sparsity in the derivative condition. Consider the following system and its correlative sparsity pattern matrix

$$f(x) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2, x_3) \\ f_3(x_2, x_3, x_4) \\ f_4(x_3, x_4, x_5) \\ f_5(x_4, x_5, x_6) \\ f_6(x_5, x_6) \end{bmatrix}, \quad R_f = \begin{bmatrix} \star & \star & \star & 0 & 0 & 0 \\ \star & \star & \star & \star & 0 & 0 \\ \star & \star & \star & \star & \star & 0 \\ 0 & \star & \star & \star & \star & \star \\ 0 & 0 & \star & \star & \star & \star \\ 0 & 0 & 0 & \star & \star & \star \end{bmatrix}.$$

The correlative sparsity pattern matrix for this system is banded and hence is a chordal sparsity pattern. The maximal cliques of the graph describing the sparsity pattern of the correlative sparsity pattern matrix has the following set of maximal cliques $\{1, 2, 3\}$, $\{2, 3, 4\}$, $\{3, 4, 5\}$ and $\{4, 5, 6\}$. If we choose a candidate Lyapunov function of the form

$$V(x) = V_1(x_1, x_2, x_3) + V_2(x_2, x_3, x_4) + V_3(x_3, x_4, x_5) + V_4(x_4, x_5, x_6)$$

then the polynomial $\dot{V}(x)$ is of the form

$$\dot{V}(x) = \begin{bmatrix} \frac{\partial V_1(x_1,x_2,x_3)}{\partial x_1} \\ \frac{\partial V_1(x_1,x_2,x_3)}{\partial x_2} + \frac{\partial V_2(x_2,x_3,x_4)}{\partial x_2} \\ \frac{\partial V_1(x_1,x_2,x_3)}{\partial x_3} + \frac{\partial V_2(x_2,x_3,x_4)}{\partial x_3} + \frac{\partial V_3(x_3,x_4,x_5)}{\partial x_3} \\ \frac{\partial V_2(x_2,x_3,x_4)}{\partial x_4} + \frac{\partial V_3(x_3,x_4,x_5)}{\partial x_4} + \frac{\partial V_4(x_4,x_5,x_6)}{\partial x_4} \\ \frac{\partial V_3(x_3,x_4,x_5)}{\partial x_5} + \frac{\partial V_4(x_4,x_5,x_6)}{\partial x_5} \\ \frac{\partial V_4(x_4,x_5,x_6)}{\partial x_6} \end{bmatrix}^T \begin{bmatrix} f_1(x_1,x_2) \\ f_2(x_1,x_2,x_3) \\ f_3(x_2,x_3,x_4) \\ f_4(x_3,x_4,x_5) \\ f_5(x_4,x_5,x_6) \\ f_6(x_5,x_6) \end{bmatrix}$$

This has the following correlative sparsity pattern matrix

$$R_{\dot{V}} = \begin{bmatrix} \star & \star & \star & \star & 0 & 0 \\ \star & \star & \star & \star & \star & 0 \\ \star & \star & \star & \star & \star & \star \\ \star & \star & \star & \star & \star & \star \\ 0 & \star & \star & \star & \star & \star \\ 0 & 0 & \star & \star & \star & \star \end{bmatrix}.$$

5.5 Exploiting Chordal Sparsity

We now return to the problem of how to exploit chordal sparsity in the SDPs that arise in SOS problems. Consider again the SDP in (5.2). In principle we may examine the monomials in $p(x)$, extract the sparsity pattern of Q and then decompose the problem using the same methods that were used in Chapter 3. However, the dimension of the Q matrix grows as $s(d) = \binom{n+d}{d}$ and this prevents the direct approach from scaling to problems involving large numbers of variables.

This problem motivates us to consider methods of exploiting chordal sparsity without directly forming Q. To do this we break the decomposition process into two steps. In the first step we use information from the correlative sparsity pattern of the polynomial to provide a coarse decomposition. In the second step we apply the chordal decomposition method to each of the blocks in the coarse decomposition arising from the first step. We next introduce some notation to allow us to explain the first step in this process.

Let $p \in \mathbb{R}[x]$ be a sparse polynomial of even degree, consisting of m monomials in n variables with correlative sparsity pattern R. Initially we will assume that the degrees of all of the monomials in p are greater than or equal to two, but later we will relax this assumption. We denote the set of monomials in the support of p by

$$W_p = \{x^\alpha \mid x^\alpha \text{ is a monomial of } p\}.$$

Let $z_d(x)$ denote the following vector of monomials of degree one up to degree d

$$z_d(x) = \left(x_1, x_2, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_{n-1}x_n, x_n^2, \ldots, x_1^d, \ldots, x_n^d\right)^T$$

which has dimension $s(d) - 1$. Let X_d be the symmetric matrix of monomials defined by

$$X_d = z_d(x)z_d(x)^T.$$

Let $V_d = \{1, 2, \ldots, s(d) - 1\}$. We next associate the set of monomials W_p with the

edge set $E_d \subseteq V_d \times V_d$ in the following way

$$E_d = \{(i,j) \mid [X_d]_{ij} \in W_p\}.$$

Let $G(R) = (V, E)$ be the graph describing the correlative sparsity pattern matrix R and let $G_{ch}(R) = (V, E')$ be a chordal extension of $G(R)$ with set of maximal cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$. For a given $C_i \in \mathcal{C}$ we denote by $z_d(x, C_i)$ the vector of monomials of degree one up to degree d that involves the subset of variables $\{x_j \mid j \in C_i\}$. For each maximal clique $C_k \in \mathcal{C}$ we define the hyper-maximal clique of degree d by

$$C_k^d = \{i \in V_d \mid \exists j \text{ s.t. } [z_d(x)]_i = [z_d(x, C_k)]_j\},$$

and for each maximal clique $C_k \in \mathcal{C}$ we define the following edge set

$$F_d = \bigcup_{k=1}^{r} \left(C_k^d \times C_k^d\right).$$

Lemma 5.5.1 Let $p \in \mathbb{R}[x]$ be a polynomial of even degree, with each monomial in p having degree greater than or equal to two. Let R be the correlative sparsity pattern of p and let $G_{ch}(R)$ be a chordal extension of $G(R)$. Then $E_d \subseteq F_d$.

Proof 5.5.1 Denote the set of maximal cliques of $G'(R)$ by $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$ and define the following sets of monomials

$$Y_d(C_i) = \{x^\alpha \mid \exists j,k \text{ s.t. } [z_d(x, C_i) z_d(x, C_i)^T]_{jk} = x^\alpha\},\ i = 1, 2, \ldots, r,$$
$$Y_d(\mathcal{C}) = \bigcup_{i=1}^{r} Y_d(C_i).$$

We next associate $Y_d(\mathcal{C})$ with the edge set $H_d \subseteq V_d \times V_d$ in the following way

$$H_d = \{(i,j) \mid [X_d]_{ij} \in Y_d(\mathcal{C})\}.$$

Since $W_p \subseteq Y_d(\mathcal{C})$ we have that $E_d \subseteq H_d$. It now remains to be shown that $H_d = F_d$

$$\begin{aligned}
(i,j) \in H_d &\Leftrightarrow \exists k,p,q \text{ such that } [X_d]_{ij} = [z_d(x,C_k)z_d(x,C_k)^T]_{pq} \\
&\Leftrightarrow \exists k,p,q \text{ such that } [z_d(x)]_i[z_d(x)]_j = [z_d(x,C_k)]_p[z_d(x,C_k)]_q \\
&\Leftrightarrow \exists k,p,q \text{ such that } [z_d(x)]_i = [z_d(x,C_k)]_p \text{ and } [z_d(x)]_j = [z_d(x,C_k)]_q \\
&\Leftrightarrow (i,j) \in F_d
\end{aligned}$$

Hence $E_d \subseteq F_d$. $\square$

Theorem 5.5.1 Let $p \in \mathbb{R}[x]$ be a polynomial of even degree, with each monomial in p having degree greater than or equal to two. Let R be the correlative sparsity pattern matrix of p and let $G_{ch}(R)$ be a chordal extension of $G(R)$ with set of maximal cliques $\mathcal{C} = \{C_1, C_2, \ldots, C_r\}$. Then F_d is a chordal extension of E_d and has set of maximal cliques given by $\mathcal{C}^d = \{C_1^d, C_2^d, \ldots, C_r^d\}$.

Proof 5.5.2 The fact that F_d is an extension of E_d follows from Lemma 5.5.1 and the hyper-maximal cliques are (by construction) the maximal cliques of the graph $G = (V_d, F_d)$. It therefore remains to be shown that the graph $G = (V_d, F_d)$ is chordal.

For $q \in \mathbb{Z}_+$ let $V_q = \{1, 2, \ldots, s(q) - 1\}$, and denote the set of hyper-maximal

cliques of order q by $\mathcal{C}^q = \{C_1^q, C_2^q, \ldots, C_r^q\}$ where

$$C_k^q = \{i \in V_q \mid \exists j \text{ s.t. } [z_q(x)]_i = [z_q(x, C_k)]_j\},\ k = 1, 2, \ldots, r. \tag{5.6}$$

Given a set of hyper-maximal cliques we define the edge set $F_q \subseteq V_q \times V_q$ in the following way

$$F_q = \bigcup_{k=1}^{r} (C_k^q \times C_k^q)\,.$$

We next show that the graphs $G_{ch}(R)$ and $G = (V_1, F_1)$ are identical and hence $G = (V_1, F_1)$ is chordal. For $q = 1$ the hyper-maximal cliques in (5.6) are the maximal cliques of $G_{ch}(R)$, i.e., $\mathcal{C}^1 = \mathcal{C}$. Since $F_1 = T$ and $V_1 = V$ and (by definition) $G_{ch}(R) = (V, T)$ is a chordal graph we have that $G = (V_1, F_1)$ is a chordal graph.

Now recall that a graph is chordal if and only if there exists a clique tree that satisfies the clique intersection property. Hence for $q = 1$ there exists a clique tree $\mathcal{T}_1 = (\mathcal{C}, \mathcal{E})$ that satisfies the clique intersection property. The clique intersection property means that for any two maximal cliques C_i, C_j in the clique tree, their intersection satisfies $(C_i \cap C_j) \subset C_g$ for all maximal cliques C_g on the unique path between C_i and C_j on the clique tree.

Now consider the clique tree $\mathcal{T}_q = (\mathcal{C}^q, \mathcal{E})$, which has the same edge set as $\mathcal{T}_1$ but where the nodes have been replaced by hyper-maximal cliques of order q. We will next show that the clique intersection property enjoyed by $\mathcal{T}_1$ is invariant for $q \in \mathbb{Z}^+$. Let C_g^q be a hyper-maximal clique on the unique path between C_i^q and C_j^q. Then from the definition in (5.6) and from the fact $(C_i \cap C_j) \subset C_q$ for all C_q

on the unique path between C_i and C_j, we have that

$$\begin{aligned} C_i^q \cap C_j^q &= \{l \in V_q \mid \exists k \text{ s.t. } [z_q(x)]_l = [z_q(x, C_i)]_k \text{ and } [z_q(x)]_l = [z_q(x, C_j)]_k\} \\ &= \{l \in V_q \mid \exists k \text{ s.t. } [z_q(x)]_l = [z_q(x, C_i \cap C_j)]_k\} \\ &\subset \{l \in V_q \mid \exists k \text{ s.t. } [z_q(x)]_l = [z_q(x, C_g)]_k\} = C_g^q \end{aligned}$$

for all $q \in \mathbb{Z}^+$. That is the set of indices in the intersection of C_i^q and C_j^q is a subset of all of the hyper-maximal cliques on the unique path between C_i^q and C_j^q for all $q \in \mathbb{Z}_+$. Therefore by setting $q = d$ we see that the clique intersection property holds for $\mathcal{T}_d$ and so we conclude that $G = (V_d, F_d)$ is a chordal graph. $\square$

We now remove the restriction in Theorem 5.5.1 that the degree of each monomial in p must be greater than or equal to two, and so generalise the theorem to polynomials of even degree. Construct the symmetric $n + 1$ by $n + 1$ matrix S as follows

$$S = \begin{bmatrix} 1 & \mathbf{1}^T \\ \mathbf{1} & R \end{bmatrix}.$$

Let $G(S)$ be the graph describing the sparsity pattern of S. We can construct a chordal extension of $G(S)$ using $G_{ch}(R)$ by taking $G_{ch}(R)$ and adding to it a new node that is connected to every other node in the graph. This means that every hyper maximal clique in $\mathcal{C}^d$ is enlarged by a single node but the graph is chordal. We therefore can decompose the Q matrix according to these maximal cliques without explicitly forming Q. We next give an example to demonstrate how the correlative sparsity decomposition works in practice.

Example 5.5.1 Suppose that we wish to find a lower bound on the minimum of

the following polynomial

$$p(x) = 2x_1^2 + 3x_1x_2 + 7x_2^2 + 4x_2x_3 + 5x_3^2 - 2x_1 + 10.$$

Using the SOS relaxation we may formulate this problem as the following SDP

$$\begin{aligned} &\text{minimise} && q_{11} \\ &\text{subject to} && \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ q_{14} & q_{24} & q_{34} & q_{44} \end{bmatrix} \succeq 0 \\ & && q_{12} = 1,\ q_{13} = 0,\ q_{14} = 0,\ q_{22} = 2,\ q_{23} = 1.5 \\ & && q_{24} = 0,\ q_{33} = 7,\ q_{34} = 2,\ q_{44} = 5. \end{aligned} \tag{5.7}$$

The sparsity pattern for S is given by

$$S = \begin{bmatrix} 1 & \mathbf{1}^T \\ \mathbf{1} & R \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix},$$

which has a chordal sparsity pattern and $G(S)$ has set of maximal cliques $\mathcal{C} = \{\{1,2,3\},\{1,3,4\}\}$. This means that we can apply Agler's theorem in order to

decompose the semidefinite constraint in (5.7) into two semidefinite constraints

$$\begin{array}{ll} \text{minimise} & q_{11} \\ \text{subject to} & A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \succeq 0,\ B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \succeq 0 \\ & q_{11} = a_{11} + b_{11},\ a_{12} = 1,\ a_{13} + b_{12} = 0,\ a_{22} = 2, \\ & a_{23} = 2,\ a_{33} + b_{22} = 7,\ b_{13} = 0,\ b_{23} = 2,\ b_{33} = 5. \end{array} \tag{5.8}$$

We may examine the sparsity pattern of A and B individually, and further decompose B into the following form

$$\begin{array}{ll} \text{minimise} & q_{11} \\ \text{subject to} & \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \succeq 0,\ \begin{bmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{bmatrix} \succeq 0,\ \begin{bmatrix} c_{11} & c_{12} \\ c_{12} & c_{22} \end{bmatrix} \succeq 0 \\ & q_{11} = a_{11} + b_{11},\ a_{12} = 1,\ a_{13} + b_{12} = 0,\ a_{22} = 2,\ a_{23} = 2 \\ & a_{33} + b_{22} + c_{11} = 7,\ c_{12} = 2,\ c_{22} = 5. \end{array} \tag{5.9}$$

5.6 Numerical Results

We implemented the correlative sparsity decomposition presented in the previous section as a collection of MATLAB functions which we have called SOSCHORDAL. In this section, we compare the performance of the standard functions in SOSTOOLS with SOSCHORDAL on two types of problems: checking whether a given polynomial is a SOS, and constructing Lyapunov functions for systems with poly-

nomial vector fields.

The t column is the time in seconds required to formulate and solve the SDP. For example, the data 1.2 (0.4,0.8) denotes that solving the whole problem took 1.2 seconds and 0.4 second were spent formulating the SDP using SOSTOOLS or SOSCHORDAL and 0.8 seconds were spent solving the problem in SeDuMi. The dim(A) column gives the dimensions of the A matrix in the SeDuMi SDP format i.e., $Ax = b$, $x \in K$. The m column gives the number of SDP cones in the SDP, and maxC gives the dimension of the largest semidefinite cone in the SDP. A '-' symbol indicates that the simulation was halted because it required excessive amount of time to terminate. All of experiments in this section were run on a MacBook Pro with a 2.9GHz processor and 8GB of RAM.

5.6.1 Quadratic Polynomial Example

As a first example, consider a quadratic function of the form

$$p(x) = 1 + \sum_{i=2}^{n} \left(100(x_i - x_{i-1})^2 + (1 - x_i)^2\right). \tag{5.10}$$

This polynomial has a banded correlative sparsity pattern of width one, and the graph describing this sparsity pattern has $n - 1$ maximal cliques of size three. Table 5.1 shows results comparing the time required to check that this polynomial is an SOS polynomial with SOSTOOLS and SOSCHORDAL using SeDuMi as the underlying SDP solver in both cases.

	SOSTOOLS/SeDuMi				SOSCHORDAL/SeDuMi			
n	t	dim(A)	m	maxC	t	dim(A)	m	maxC
10	1.0 (0.4/0.6)	66 x 121	1	11	0.7 (0.4/0.3)	30 x 81	9	3
20	1.2 (0.6/0.6)	231 x 441	1	21	1.1 (0.5/0.6)	60 x 171	19	3
30	2.7 (0.8/1.8)	496 x 961	1	31	1.2 (0.6/0.6)	90 x 261	29	3
40	5.9 (1.2/4.7)	861 x 1681	1	41	1.9 (1.1/0.8)	120 x 351	39	3
50	11.2 (1.4/9.8)	1326 x 2601	1	51	1.7 (0.8/0.9)	150 x 441	49	3
60	23.9 (1.7/22.2)	1891 x 3721	1	61	2.2 (1.1/1.0)	180 x 531	59	3
70	59.6 (2.4/57.2)	2556 x 5041	1	71	2.4 (1.4/1.0)	210 x 621	69	3
80	117.1 (2.8/114.3)	3321 x 6561	1	81	2.5 (1.4/1.1)	240 x 711	79	3
90	259.3 (3.8/255.5)	4186 x 8281	1	91	3.5 (1.7/1.8)	270 x 801	89	3
100	645.1 (4.0/641.1)	5151 x 10201	1	101	2.9 (1.7/1.2)	300 x 891	99	3
110	-	-	-	-	3.3 (1.8/1.4)	330 x 981	109	3
120	-	-	-	-	3.6 (2.1/1.5)	360 x 1071	119	3
130	-	-	-	-	4.4 (2.6/1.8)	390 x 1161	129	3
140	-	-	-	-	4.1 (2.4/1.7)	420 x 1251	139	3
150	-	-	-	-	4.5 (2.8/1.8)	450 x 1341	149	3
160	-	-	-	-	5.2 (2.9/2.3)	480 x 1431	159	3
170	-	-	-	-	5.1 (3.1/2.0)	510 x 1521	169	3
180	-	-	-	-	5.4 (3.3/2.0)	540 x 1611	179	3
190	-	-	-	-	5.7 (3.6/2.1)	570 x 1701	189	3
200	-	-	-	-	6.2 (4.0/2.2)	600 x 1791	199	3

Table 5.1: CPU time required to check that the quadratic function in (5.10) is an SOS polynomial using SOSTOOLS and SOSCHORDAL. n is the number of variables, t is the CPU time, dim(A) is the dimension of the A matrix in the SeDuMi SDP format, m is the number of semidefinite cones in the problem, maxC is dimension of the largest semidefinite cone in the SDP.

Table 5.1 shows that the time required to check that the quadratic function in (5.10) is an SOS polynomial increases rapidly for SOSTOOLS/SeDuMi, with the majority of the computation time being spent in SeDuMi solving the SDP. In contrast, SOSCHORDAL/SeDuMi solves this problem significantly more efficiently using the same SDP solver. The main reason for this difference in performance is that SOSCHORDAL exploits the sparsity in the polynomial to decompose the semidefinite constraint in the SDP into $n-1$ semidefinite constraints of size 3. This reduces the size of the A matrix significantly and therefore facilitates the calculation of the Newton step at each iteration of the interior-point method.

5.6.2 Quartic Polynomial Example

As a second example, we consider quartic polynomials of the form

$$p(x) = \sum_{i=1}^{n-1} (x_i^2 + x_{i+1}^2)^2, \tag{5.11}$$

and compare the time required to verify that they are SOS polynomials. Note that this polynomial has an identical correlative sparsity pattern to the quadratic example function but the higher degree of this polynomial means that the hyper maximal cliques in the correlative sparsity decomposition will be larger for this example. Table 5.2 displays the time required to check that this polynomial is an SOS polynomial using SOSTOOLS and SOSCHORDAL.

	SOSTOOLS/SeDuMi				SOSCHORDAL/SeDuMi			
n	t	dim(A)	m	maxC	t	dim(A)	m	maxC
10	5.7 (0.5/5.2)	1001 x 4356	1	66	0.7 (0.4/0.3)	95 x 324	9	6
12	21.2 (0.9/20.3)	1820 x 8281	1	91	0.8 (0.4/0.5)	115 x 396	11	6
14	85.9 (1.8/84.0)	3060 x 14400	1	120	1.1 (0.5/0.5)	135 x 468	13	6
16	464.9 (2.8/462.1)	4845 x 23409	1	153	1.2 (0.6/0.6)	155 x 540	15	6
18	2042.2 (5.0/2037.2)	7315 x 36100	1	190	0.9 (0.6/0.3)	175 x 612	17	6
20	6665.2 (8.8/6656.4)	10626 x 53361	1	231	1.2 (0.7/0.6)	195 x 684	19	6
30	-	-	-	-	1.6 (0.8/0.8)	295 x 1044	29	6
40	-	-	-	-	1.5 (0.9/0.6)	395 x 1404	39	6
50	-	-	-	-	1.7 (1.0/0.7)	495 x 1764	49	6
60	-	-	-	-	2.1 (1.3/0.8)	595 x 2124	59	6
70	-	-	-	-	2.4 (1.5/0.8)	695 x 2484	69	6
80	-	-	-	-	4.3 (1.8/2.5)	795 x 2844	79	6
90	-	-	-	-	5.0 (2.4/2.6)	895 x 3204	89	6
100	-	-	-	-	6.4 (3.3/3.0)	995 x 3564	99	6

Table 5.2: CPU time in seconds required to check that the quartic polynomial in (5.11) is an SOS polynomial. n is the number of variables, t is the CPU time, dim(A) is the dimension of the A matrix in the SeDuMi SDP format, m is the number of semidefinite cones in the problem, maxC is dimension of the largest semidefinite cone in the SDP.

Similarly to the case of the quadratic polynomial example, the output of SOSCHORDAL is a far more compact SDP than the output of SOSTOOLS, and this translates into significant improvements in the rate at which the SDP is solved.

5.6.3 Varying Band Width

Next we examine the effect of varying the size of the maximal cliques in the correlative sparsity pattern matrix on the performance of SOSCHORDAL. We consider quadratic polynomials with variable bandwidths of the form

$$p(x) = \sum_{i=1}^{n-k} (x_i + x_{i+1} + \cdots + x_{i+k})^2.$$

Table 5.3 summarises the results. Each entry is the time taken to check that $p(x)$ is an SOS polynomial using SOSCHORDAL and SeDuMi. We find a trade-off between having a large number of variables with a low bandwidth (small maximal clique sizes) and having a small number of variables with a high bandwidth (large maximal clique sizes).

5.6.4 Local Stability of Banded Systems

We next consider the problem of constructing Lyapunov functions to check for local stability i.e., solving instances of (5.5). We first consider dynamical systems $\dot{x} = f(x)$ and constraint sets $D = \{x \in \mathbb{R}^n \mid g_j(x) \geq 0,\ j = 1, 2, \ldots, m\}$ of a

	k					
n	5	10	15	20	25	30
50	1.8	3.2	3.8	5.4	6.7	7.7
75	4.1	5.3	7.1	9.9	14.1	18.1
100	3.4	6.4	9.6	14.0	20.8	29.8
125	6.0	9.5	14.6	22.6	32.1	45.4
150	6.2	10.4	17.1	29.2	43.5	-
175	8.0	14.1	25.0	44.5	-	-
200	8.6	20.3	37.4	-	-	-
225	11.4	24.3	44.3	-	-	-
250	14.1	29.4	57.1	-	-	-
275	16.5	37.1	-	-	-	-
300	18.7	46.5	-	-	-	-
325	20.9	51.9	-	-	-	-
350	25.3	58.4	-	-	-	-
375	29.0	70.6	-	-	-	-
400	32.1	78.6	-	-	-	-

Table 5.3: CPU time in seconds required to check that a quadratic polynomial is an SOS polynomial using SOSCHORDAL as a function of the number of variables (n) and bandwidth (k). t is the CPU time, $\dim(A)$ is the dimension of the A matrix in the SeDuMi SDP format, m is the number of semidefinite cones in the problem, maxC is dimension of the largest semidefinite cone in the SDP.

banded form

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1, x_2), & g_1(x) &= \gamma - (x_1^2 + x_2^2) \geq 0 \\
\dot{x}_2 &= f_2(x_1, x_2, x_3), & g_2(x) &= \gamma - (x_1^2 + x_2^2 + x_3^2) \geq 0 \\
&\vdots & &\vdots \\
\dot{x}_{n-1} &= f_{n-1}(x_{n-2}, x_{n-1}, x_n), & g_{n-1}(x) &= \gamma - (x_{n-2}^2 + x_{n-1}^2 + x_n^2) \geq 0 \\
\dot{x}_n &= f_n(x_{n-1}, x_n), & g_n(x) &= \gamma - (x_{n-1}^2 + x_n^2) \geq 0.
\end{aligned}$$

Systems of this form have a sparse generalised correlative sparsity matrix that is banded with width two. We generated locally stable systems of degree three and

compared the time required by SOSTOOLS and SOSCHORDAL in order to check the feasibility of diagonal, quadratic Lyapunov functions.

	SOSTOOLS/SeDuMi				SOSCHORDAL/SeDuMi			
n	t	dim(A)	m	maxC	t	dim(A)	m	maxC
4	1.9 (1.3/0.6)	106 x 305	8	15	1.5 (0.8/0.7)	87 x 231	18	10
6	3.5 (2.2/1.3)	273 x 924	12	28	2.1 (1.3/0.8)	153 x 403	40	10
8	6.1 (3.7/2.4)	589 x 2233	16	45	3.1 (1.7/1.3)	245 x 635	69	10
10	11.0 (5.3/5.7)	1130 x 4640	20	66	3.8 (2.2/1.6)	333 x 843	108	10
12	28.2 (7.6/20.6)	1988 x 8649	24	91	5.5 (2.8/2.7)	507 x 1387	151	10
14	94.8 (10.4/84.4)	3271 x 14860	28	120	7.2 (3.5/3.7)	627 x 1659	206	10
16	470.9 (13.9/457.0)	5103 x 23969	32	153	11.0 (5.6/5.3)	763 x 1891	267	10
18	1974.9 (18.9/1956.0)	7624 x 36768	36	190	13.1 (6.6/6.5)	915 x 2263	339	10
20	-	-	-	-	18.5 (7.2/11.3)	1103 x 2651	415	10
30	-	-	-	-	34.5 (10.8/23.7)	2253 x 5167	923	10
40	-	-	-	-	90.0 (16.3/73.7)	3863 x 8867	1631	10
50	-	-	-	-	122.1 (21.9/100.2)	5823 x 12975	2537	10
60	-	-	-	-	179.1 (30.0/149.0)	8213 x 18183	3646	10

Table 5.4: Time required to check the feasibility of a diagonal, quadratic Lyapunov functions using SOSTOOLS and SOSCHORDAL. n is the number of variables, t is the CPU time, dim(A) is the dimension of the A matrix in the SeDuMi SDP format, m is the number of semidefinite cones in the problem, maxC is dimension of the largest semidefinite cone in the SDP.

5.6.5 Local Stability of Sparse Systems

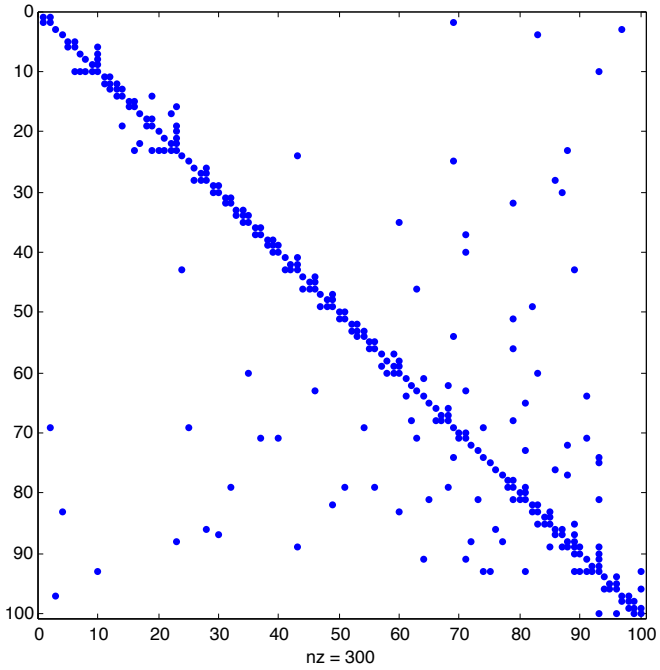
We next consider sparse, randomly generated dynamical systems $\dot{x} = f(x)$ with polynomial dynamics of degree three and stable linearisations about an equilibrium point at zero. For this set of experiments we first generated a correlative sparsity pattern for $f(x)$ using the chordalGen function, see Appendix A.3.1, with the threshold on the maximal clique size set to three. For each experiment, let $G = (V, E)$ be the graph representing this correlative sparsity pattern of $f(x)$ and let $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$ be the set of maximal cliques of this graph. The polynomials $g_1, g_2, \ldots, g_m$ that define the constraint set $D = \{x \in \mathbb{R}^n \mid g_j(x) \geq 0,\ j = 1, 2, \ldots, m\}$ were chosen in the following way

$$g_j(x) = \alpha - \|\mathbf{x}_j\|_2^2$$

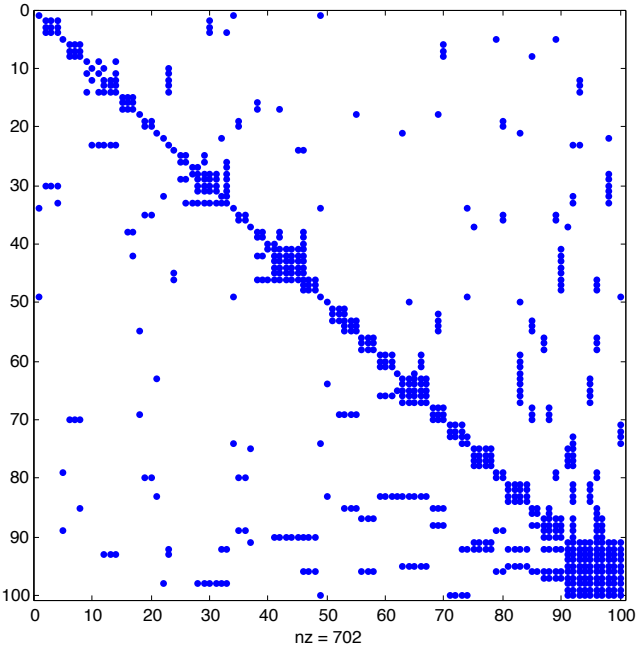
where $\mathbf{x}_j = \{x_i \mid i \in C_j\}$ for $j = 1, 2, \ldots, m$. The candidate Lyapunov function was chosen to be a quadratic polynomial with the same correlative sparsity pattern as $f(x)$. The multiplier polynomials $q_j(x), r_j(x)$ were chosen to be polynomials of appropriate degree with each multiplier being a function of the same set of variables as the associated constraint polynomial $g_j(x)$.

In Figure 5.1 we show a typical sparsity pattern for an example involving 100 states. This example is particularly interesting because a diagonal Lyapunov function was insufficient to show that the system was locally stable and therefore other standard methods would require us to either use denser Lyapunov functions or increase the degree of the Lyapunov function to be quartic. However, we found that by using a correlatively sparse candidate Lyapunov function of the form shown in

Figure 5.1a we were able to construct a Lyapunov function in 85.9s (this experiment was carried out using a server computer as it was too large to be tackled on a desktop computer). Table 5.5 shows a comparison between SOSTOOLS and SOSCHORDAL for a number of smaller examples that were generated using the method above, in each case a diagonal Lyapunov function was insufficient but a correlatively sparse Lyapunov function returned a feasible solution.



(a) Correlative sparsity pattern of $f(x)$ and $V(x) - \sum_{j=1}^{m} q_j(x)g_j(x) - \varphi(x)$.



(b) Correlative sparsity pattern of $\nabla V(x)^T f(x) - \sum_{j=1}^{m} r_j(x)g_j(x)$

Figure 5.1: Correlative sparsity patterns for the polynomials in the 100 state example.

	SOSTOOLS/SeDuMi				SOSCHORDAL/SeDuMi			
n	t	dim(A)	m	maxC	t	dim(A)	m	maxC
4	2.1 (1.5/0.5)	103 x 304	6	15	1.4 (1.0/0.5)	107 x 316	10	15
6	3.7 (2.6/1.0)	270 x 927	10	28	2.3 (1.6/0.7)	227 x 738	29	21
8	6.3 (4.3/2.0)	586 x 2240	14	45	3.5 (2.2/1.3)	306 x 1085	58	15
10	11.7 (6.9/4.8)	1127 x 4651	18	66	5.2 (3.1/2.1)	473 x 1496	91	21
12	27.4 (8.6/18.8)	1985 x 8664	22	91	6.6 (3.7/2.9)	677 x 2130	133	21
14	89.6 (13.2/76.4)	3268 x 14879	26	120	14.8 (5.8/9.0)	1098 x 3832	184	45
18	1775.3 (24.4/1750.9)	7621 x 36795	34	190	19.6 (7.7/11.9)	1197 x 3755	314	28
20	6319.3 (30.5/6288.9)	10987 x 54176	38	231	23.3 (8.7/14.6)	1502 x 4703	387	28
30	-	-	-	-	50.7 (12.9/37.8)	2875 x 8156	880	28
40	-	-	-	-	104.6 (22.2/82.3)	4870 x 13689	1576	45
50	-	-	-	-	173.9 (30.0/143.9)	7322 x 20448	2471	45
60	-	-	-	-	219.4 (37.5/181.9)	9542 x 24540	3567	45

Table 5.5: Time required to check feasibility of sparse, quadratic Lyapunov functions using SOSTOOLS and SOSCHORDAL for sparse dynamical systems of degree three.

5.7 Chapter Summary

SOS problems are computationally intensive to solve because of the characteristic $s = \binom{n+d}{d}$ growth in the the dimension of the SDP cone. In this chapter, we have shown that it is possible to exploit chordal sparsity without forming the $s \times s$ matrix of coefficients Q by using the correlative sparsity of the polynomial as an intermediary step. We used this approach to reformulate a number of sparse SOS problems in order to exploit chordal sparsity. This led to improvements in the scale of SOS problems that could be addressed and we demonstrated that there is potential to exploit chordal sparsity in sparse SOS problems. To develop this approach further it will be necessary to consider SDP solvers that can parallelise the computation tasks or make use of less computationally expensive first order methods. The dependence of SOSTOOLS and SOSCHORDAL on symbolic variables is also undesirable, further improvements could be made by expressing monomials in a more efficient way.

Chapter 6

Conclusion

6.1 Summary

In this concluding chapter, we provide a summary of the thesis and look towards some future research directions. Overall, we have shown that the chordal sparsity approach can be used to decompose the semidefinite constraints that appear within a number of analysis and synthesis problems in control theory. In chapter 3 we considered analysis problems for linear systems and exploited sparsity within the Lyapunov equation. In Chapter 4 we considered two different methods of exploiting chordal sparsity in static state feedback synthesis problems for linear time-invariant systems. The first method made use of a connection between chordal graphs and completions of matrices with sparse inverses. The second method took the more direct approach of describing the sparsity of the state space system using an undirected graph and applying the same techniques as in Chapter 3. Finally, in Chapter 5 we considered the problem of local stability analysis for autonomous systems with polynomial dynamics using the Sum of Squares method.

We showed that by using a chordal extension to this correlative sparsity pattern of a polynomial system we can decompose the semidefinite constraints within the local stability analysis problem using Agler's theorem without forming the coefficient matrix directly.

6.2 Future Research Directions

The majority of the thesis focussed on how to solve optimisation problems related to the construction of Lyapunov functions, and was concerned with how to pick the sparsity pattern of the candidate Lyapunov function so as to minimise the computational burden for the solver. A remaining open question is what the existence of a chordal Lyapunov function means from a systems perspective. In other words, suppose that we know that a system admits a chordal Lyapunov function, what does that tell us about the system? From Agler's theorem it is straightforward to see that a Lyapunov function with a chordal sparsity pattern can be decomposed into a sum of functions that are each positive definite, but further work needs to be carried out to investigate what this implies about the properties of the system under analysis.

In the third chapter we considered the problem of picking a sparsity pattern for a candidate Lyapunov function so as to preserve sparsity in the resulting SDP. We showed that when cardinality constraints are present, this is an NP-hard problem for general sparse linear systems. The chordal powers algorithm addressed this problem in a pragmatic way by using powers of the system matrix to ensure that edges are added between closely connected nodes in the graph. An interesting

question for future research is whether a more systems theoretic approach could provide an alternative method for selecting edges.

An important result from this thesis is that the combination of chordal sparsity and Sums of Squares provides a method for addressing the local stability analysis problem for large, sparse systems with polynomial dynamics. We demonstrated that it is possible to construct a Lyapunov function for a sparse polynomial system involving up to a hundred states using a single computer. One can imagine that the future of solving large sparse SOS problems will involve distributing the processing across multiple computers using a distributed SDP solver, such as was proposed in [89, 90, 91, 92]. Chordal sparsity provides a mechanism by which sparse SDPs can be decomposed into manageable pieces, with each computer/processor in the cluster perhaps handling the semidefinite constraint corresponding to a maximal clique in the graph describing the sparsity pattern of the matrix variables in the problem.

Finally, we note that there is potential to exploit chordal sparsity in the local stability analysis problem in other ways than was proposed in this thesis. Recall that in chapter 5 we chose candidate Lyapunov functions that included all of the monomials up to some degree d that satisfied the desired correlative sparsity pattern. In this case the hyper maximal cliques that feature in Theorem 5.5.1 accurately reflect the sparsity pattern of the coefficient matrices required to represent $V(x)$ and $\nabla V(x)^T f(x)$ in the SDP. However, it is conceivable that this approach could be further refined so that the design of the candidate Lyapunov function is carried out on a monomial by monomial basis to allow for a finer control of the sparsity pattern of the coefficient matrices in the SDP. The sparsity pattern of the coeffi-

cient matrix in this case would be covered by the hyper maximal cliques defined in chapter 5, and so the hyper maximal cliques could be used to constrain the search for nonzero entries during the decomposition step. We suggest the following algorithm to do this: construct a clique tree for the set of hyper maximal cliques, then for each hyper maximal clique in the clique tree examine the sparsity pattern within the submatrix of the coefficient matrix associated with that hyper maximal clique and then use the clique tree to combine these sparsity patterns together to form a single graph. This process is much like taking the individual pages from an ordnance survey book and using the page numbers to piece together a unified map of a whole country.

Appendices

Appendix A

A.1 Chordal Sparsity in Inference Problems

In this appendix we move away from control theory to discuss chordal sparsity in the context of probabilistic inference. We begin with message passing algorithms for graphical models, and explain how chordal graphs play an important role as part of the junction tree algorithm. Finally, we consider approximations to the marginal polytope of a probability distribution using SDPs and an upper bound on entropy function proposed by Wainright *et. al.* in [93, 94]. This shows a connection between probabilistic inference and the work by Lasserre on moment matrices and the duality between Grone's and Agler's theorem.

A.1.1 Graphical Models and Message Passing

For $i \in \{1, 2, \ldots, n\}$ let x_i be a random variable that takes values in the set $\mathcal{X} = \{1, 2, \ldots, r\}$, and for $j \in \{1, 2, \ldots, m\}$ let y_j be a sensor measurement that takes values in the set $\mathcal{Y} = \{1, 2, \ldots, s\}$. Let $G = (V, E)$ be an undirected graph, with $V = \{1, 2 \ldots, n\}$ and $E \subseteq V \times V$. Let the set of maximal cliques of G be denoted by $\mathcal{C} = \{C_1, C_2, \ldots, C_p\}$. We associate each maximal clique $C \in \mathcal{C}$ with a

nonnegative function $\psi_C(x_C, y_C) \geq 0$ called a factor, where x_C and y_C are states and measurements associated with factor C. The joint probability distribution for x and y is assumed to be formed as a product of factors

$$p(x, y) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C, y_C),$$

where Z is a normalisation constant. Inference problems involving graphical models are found throughout science and engineering [6]. They occur frequently in statistical physics, machine learning, digital communications and computer vision problems [93]. Some specific examples of such problems are: inferring objects from 2D arrays of pixels captured by cameras [1], and decoding messages that have been corrupted by noise [95].

A classical problem in inference is to calculate the marginal probability distribution for a particular subset of the variables. In principle this can be carried out by summing over all possible states of the other variables but this exact marginalization process becomes computationally intractable for large numbers of variables. However, when the graph describing the sparsity pattern of the graphical model is a tree, it is possible to perform exact marginalization efficiently using a class of algorithms called message passing algorithms [6]. For a tree structured graph the maximal cliques consist of pairs of neighbouring nodes and so the probability distribution can be factorized as

$$p(x) = \frac{1}{Z} \prod_{i \in V} \psi_i(x_i) \prod_{(i,j) \in E} \psi_{ij}(x_i, x_j),$$

where for notational convenience we have dropped the dependence on sensor measurements from the factors. The marginal distribution, also known as a belief, $b_i(x_i)$ can be calculated using a message passing algorithm called the sum-product algorithm. The messages passed between nodes in graph in the sum-product algorithm are of the form

$$M_{ji}(x_i) \leftarrow \kappa \sum_{x'_j} \left(\psi_{ij}(x_i, x'_j)\psi_j(x'_j) \prod_{k \in N(j)\setminus i} M_{kj}(x'_j) \right),$$

where $\kappa > 0$ is a normalisation constant. A message $M_{ji}(x_i)$ can be thought of as a message from node j to node i that contains the likelihood of each possible state of node i is given the information j has received from all of its neighbours except for i. For tree structured graphs this message passing scheme is guaranteed to converge to a unique fixed point $M^* = \{M^*_{ij}, M^*_{ji}\}$ and the correct marginal distributions can then be calculated from

$$b_i(x_i) = \kappa\psi_i(x_i) \prod_{j \in N(i)} M^*_{ji}(x_i),$$

which can be interpreted as the local likelihood for each state that x_i could take weighted by all of the messages from its neighbours. For graphs with cycles the message passing scheme above is not guaranteed to converge to the correct marginals or even to converge at all, but it can be shown that fixed points of the sum-product algorithm correspond to local minima of the Bethe free energy [96, 97].

The link between message passing algorithms and chordal sparsity is that when the

graph defining the sparsity pattern of the probability distribution is chordal (or can be extended to be chordal) we can form a clique tree and pass messages between the maximal cliques in the clique tree. This algorithm is called the junction tree algorithm and is guaranteed to converge to the correct marginals, but has the drawback that the computational complexity of the algorithm scales exponentially in the size of the largest maximal cliques in the graph [6].

A.1.2 Moments and the Marginal Polytope

We next discuss how SDPs can be used to place outer bounds on the marginal polytope of a probability distribution, and how Grone's theorem can be used to decompose the semidefinite constraints within these problems when sparsity is present. Let $\phi : \mathbb{R}^n \to \mathbb{R}^m$ denote a collection of factors. The exponential family defined by $\phi(x)$ is the set of probability distributions of the form

$$p(x;\theta) = \exp\left\{\langle\theta, \phi(x)\rangle - \Phi(\theta)\right\}, \tag{A.1}$$

where $\theta \in \mathbb{R}^m$ and $\Phi(\theta)$ is the log partition function that normalises the probability distributions

$$\Phi(\theta) = \log \sum_{x \in \mathcal{X}^n} \exp\left\{\langle\theta, \phi(x)\rangle\right\}.$$

The conjugate dual function of the log partition function is defined by

$$\Phi^*(\mu) = \sup_{\theta \in \mathbb{R}^m} \left\{\langle\mu, \theta\rangle - \Phi(\theta)\right\}, \tag{A.2}$$

where the parameters μ are known as dual variables. By taking the derivative of the log partition function with respect to a particular θ_i and setting this derivative

to zero we obtain a relationship between the ith dual variable and the expectation of the ith factor with respect to the probability distribution parameterised by θ

$$\hat{\mu}_i = \mathbb{E}_{\hat{\theta}_i}[\phi_i(x)], \quad i = 1, 2, \ldots, m.$$

Substituting this value $\hat{\mu}$ into the definition of the conjugate dual function we find that the value of the conjugate dual function is given by the negative entropy of the probability distribution parameterised by $\hat{\theta}$

$$\Phi^*(\hat{\mu}) = \sum_{x \in \mathcal{X}} p(x; \hat{\theta}) \log p(x; \hat{\theta}).$$

Since Φ is convex and lower semi-continuous we can recover Φ by taking the conjugate of Φ^*, which by definition is given by

$$\Phi(\theta) = \sup_{\mu \in \mathrm{dom}\Phi^*} \left\{ \langle \theta, \mu \rangle - \Phi^*(\mu) \right\}.$$

This expression prompts us to explicitly define the set of points in the domain of Φ^* which is called the marginal polytope

$$\mathcal{M}(\phi) = \left\{ \mu \in \mathbb{R}^m \mid \exists p(x) \text{ such that } \mathbb{E}_p[\phi(x)] = \mu \right\}.$$

Note that the marginal polytope is determined by the factors describing the exponential family and is exactly the convex hull of the finite set of points $\{\phi(x) \mid x \in \mathcal{X}^n\}$. The marginal polytope can therefore be characterised by either the convex hull of a finite number of vectors (the extreme points) or by the intersection of a finite number of half-spaces.

Since the unique optimum is attained at the exact marginals of $p(x;\theta)$, which we know lie within the marginal polytope, we can replace the supremum with max to give

$$\Phi(\theta) = \max_{\mu \in \mathbb{M}(\phi)} \left\{ \langle \theta, \mu \rangle - \Phi^*(\mu) \right\}. \tag{A.3}$$

Hence by maximising (A.3) we can calculate the desired mean parameters $\mu = \mathbb{E}_p[\phi(x)]$ corresponding to $p(x;\theta)$. However, in general the entropy function and the marginal polytope are complex and difficult to optimise over exactly, and so researchers have sought methods of solving this optimisation problem approximately.

One approach to solving the approximate inference problem is replace the constraint on the dual variables with local consistency requirements between neighbouring nodes. Another approach is to consider more global consistency requirement related to cycles in the graphs [98]. The approach we will follow is to combine a Gaussian bond on the entropy function with an outer bound on the marginal polytope based on semidefinite constraints. We next give some background on moments and moment matrices in order to explain how they are related to the inference problem before considering how to exploit chordal sparsity in the SDPs that arise in this problem.

Given a probability distribution $p(x)$, the moments of $p(x)$ are defined as the sequence of real numbers $y = (y_\alpha)$ such that

$$y_\alpha = \sum_{x \in \mathcal{X}^n} p(x) x^\alpha, \quad \forall \alpha \in \mathbb{N}^n.$$

In other words, the moments are the expectations of the monomials of x over the probability distribution $p(x)$. We next define the set of moment matrices which will be used to provide an outer bound on the marginal polytope which is required for solving inference problems of the form above. For any $d \in \mathbb{Z}_+$ the set of moment matrices is defined as

$$\mathbb{M}_d = \{M \in \mathbb{S}^{s(d)} \mid \exists p(x) \text{ s.t. } M = \sum_{x \in \mathcal{X}^n} p(x)[v_d(x)v_d(x)^T]\}, \tag{A.4}$$

where $v_d(x)$ is the vector of monomials defined in (5.1) and $p(x)$ is a probability distribution.

For inference problems that involve optimising over the marginal polytope it is useful to think of a moment matrix as a function of the moments y. To make this more explicit, given a sequence $y = (y_\alpha)$ we will use the notation $M_d(y)$ to denote the moment matrix of dimension $s(d)$ with rows and columns labeled by the monomials $\alpha \in \mathbb{N}^n_d$ and constructed according to

$$[M_d(y)]_{\alpha,\beta} = y_{\alpha+\beta}, \quad \forall \alpha, \beta \in \mathbb{N}^n_d.$$

For example, let $p(x)$ be a probability distribution for a random variable $x = [x_1, x_2]$ with sequence of moments $y = (y_{00}, y_{10}, y_{01}, \ldots)$ then for $d = 2$ the moment

matrix $M_2(y)$ associated with this probability distribution is given by

$$M_2(y) = \begin{bmatrix} 1 & y_{10} & y_{01} & y_{20} & y_{11} & y_{02} \\ y_{10} & y_{20} & y_{11} & y_{30} & y_{21} & y_{12} \\ y_{01} & y_{11} & y_{02} & y_{21} & y_{12} & y_{03} \\ y_{20} & y_{30} & y_{21} & y_{40} & y_{31} & y_{22} \\ y_{11} & y_{21} & y_{12} & y_{31} & y_{22} & y_{13} \\ y_{02} & y_{12} & y_{03} & y_{22} & y_{13} & y_{04} \end{bmatrix}$$

where we have used the fact that $y_{00} = \sum_{x \in \mathcal{X}^n} p(x) = 1$.

The following proposition gives a necessary condition for the entries of a moment matrix $M_d(y)$ to correspond to the moments of a valid probability distribution.

Proposition A.1.1 [17] If y is a sequence of moments for some probability distribution $p(x)$ then $M_d(y) \succeq 0$.

To see this, let $q \in \mathbb{R}[x]$ be a polynomial of degree d, then taking the expectation of $q(x)^2$ with respect to an arbitrary probability distribution $p(x)$ we have

$$\begin{aligned} \sum_{x \in \mathcal{X}^n} p(x) q(x)^2 \geq 0 &\Leftrightarrow \sum_{x \in \mathcal{X}^n} p(x) \mathbf{q}^T v_d(x) v_d(x)^T q \geq 0 \\ &\Leftrightarrow \mathbf{q}^T \left(\sum_{x \in \mathcal{X}^n} p(x) v_d(x) v_d(x)^T \right) \mathbf{q} \geq 0 \\ &\Leftrightarrow \langle \mathbf{q}, M_d(y) \mathbf{q} \rangle \geq 0 \Rightarrow M_d(y) \succeq 0. \end{aligned}$$

Proposition A.1.1 provides a outer bound on the marginal polytope for this model, since by definition any $\mu \in \mathcal{M}(\phi_{Ising})$ is a truncated sequence of moment for some probability distribution, which implies that for any $d \in \mathbb{Z}_+$ there exists a sequence

of moments $y = (y_\alpha)$ such that $M_d(y) \succeq 0$, where $y_\alpha = \mu_\alpha$ for all $\alpha \in I$ and hence we can define the constraint set

$$\text{SDEF}_d = \{\mu \in \mathbb{R}^m \mid \exists y \text{ s.t. } M_d(y) \succeq 0,\ y_\alpha = \mu_\alpha\ \forall \alpha \in I\} \supseteq \mathcal{M}(\phi_{Ising}).$$

Furthermore, since $M_d(y)$ is a submatrix of $M_{d+1}(y)$ the bounds become tighter and tighter so that the sets SDEF_d have the following nested property [99]

$$\text{SDEF}_1 \supseteq \text{SDEF}_2 \supseteq \cdots \supseteq \text{SDEF}_n = \mathcal{M}(\phi_{Ising}).$$

In [94] the authors propose an outer bound on the marginal polytope and an upper bound on the entropy function for Ising models with pairwise factors. They considered Ising models consisting of n nodes on a complete graph $G = (V, E)$, where each node $i \in V$ represented random variable $x_i \in \{-1, 1\}$ and the factors were of the form

$$\phi_{Ising} = \{x_s \mid s \in V\} \cup \{x_s x_t \mid (s, t) \in E\}. \tag{A.5}$$

For a complete graph there are $m = n + n(n+1)/2$ possible factors and we index each of the m factors according to the exponent of that factor and denote the set of indexes by I. The marginal polytope for the Ising model is then defined by

$$\mathcal{M}(\phi_{Ising}) = \{\mu \in \mathbb{R}^m \mid \exists p(x) \text{ s.t. } \mu_\alpha = \mathbb{E}_p[x^\alpha]\ \ \forall \alpha \in I\}.$$

An upper bound on the entropy function is given by

$$-\Phi^*(\mu) \leq \frac{1}{2}\log\det M_1(\mu) + \frac{1}{3}\text{blkdiag}[0, I_n] + \frac{n}{2}\log\frac{\pi e}{2}.$$

By substituting this expression into the variational problem (A.3) the authors obtained

$$\begin{aligned}\Phi(\theta) \leq \max_{\mu} &\left\{\langle\theta,\mu\rangle + \frac{1}{2}\log\det\left[M_1(\mu) + \frac{1}{3}\text{blkdiag}[0, I_n]\right]\right\} + \frac{n}{2}\log\frac{\pi e}{2} \\ &\text{subject to } M_d(\mu) \succeq 0.\end{aligned} \tag{A.6}$$

The computational complexity of solving this problem grows rapidly due to the dense semidefinite constraint, and so to address this scaling problem Wainwright considered how to relax the semidefinite constraint in [100]. However, relaxing the problem in this way means that it no longer provides an upper bound on the log determinant function. When the graph $G = (V, E)$ describing the sparsity pattern of the graphical model is chordal (or can be extended to be chordal) the semidefinite constraint on the moment matrix can be decomposed using Grone's theorem so that (A.6) can be written as the equivalent problem

$$\begin{aligned}\max_{\mu} &\left\{\langle\theta,\mu\rangle + \frac{1}{2}\log\det\left[M_1(\mu) + \frac{1}{3}\text{blkdiag}[0, I_n]\right]\right\} + \frac{n}{2}\log\frac{\pi e}{2} \\ &\text{subject to } M_d(\mu, C_k) \succeq 0, \quad k = 1, 2, \ldots, p,\end{aligned} \tag{A.7}$$

where $M_d(\mu, C_k)$ is the moment matrix of degree d involving the subsets of variables in maximal clique C_k. This decomposition step and the fact that efficient methods exist for evaluating the gradient of the log determinant function allow the problem to be solved efficiently [5, 54, 101].

A.2 Chordal Powers Example

In this appendix we give an example of a banded system which does not have a quadratic Lyapunov function with the same or smaller bandwidth as the system matrix, but for which a Lyapunov function of bandwidth equal to four can be found using the chordal powers algorithm. Consider the continuous-time system $\dot{x} = Ax$ with

$$A = \begin{bmatrix} -0.3288 & 0.0786 & 0.3620 & 0 & 0 & 0 & 0 & 0 \\ -0.1104 & -0.1174 & -0.3744 & -0.1307 & 0 & 0 & 0 & 0 \\ -0.0099 & 0.1614 & -0.2476 & 0.1613 & -0.1133 & 0 & 0 & 0 \\ 0 & -0.0303 & -0.0627 & 0.0026 & 0.3222 & -0.1662 & 0 & 0 \\ 0 & 0 & 0.1126 & -0.2259 & -0.2978 & -0.2947 & 0.1508 & 0 \\ 0 & 0 & 0 & 0.4979 & 0.2816 & -0.3988 & -0.2627 & 0.1031 \\ 0 & 0 & 0 & 0 & 0.4877 & -0.4688 & -0.4158 & -0.3160 \\ 0 & 0 & 0 & 0 & 0 & 0.3276 & 0.4364 & -0.8057 \end{bmatrix} \prec 0.$$

Using the chordal powers algorithm we find that this example does not admit a quadratic Lyapunov function when the sparsity pattern of P is constrained to be a diagonal matrix, or banded with width two, but on the second iteration of the algorithm we obtain the following P matrix, with bandwidth four, which does

satisfy the Lyapunov conditions.

$$P = \begin{bmatrix} 0.8278 & 0.0933 & 0.1080 & 0.0020 & 0.1055 & 0 & 0 & 0 \\ 0.0933 & 0.8282 & -0.0683 & -0.2479 & -0.3573 & 0.2077 & 0 & 0 \\ 0.1080 & -0.0683 & 1.4787 & -0.1904 & 0.0427 & -0.4544 & 0.2594 & 0 \\ 0.0020 & -0.2479 & -0.1904 & 1.9289 & 0.5867 & -0.0777 & -0.2947 & -0.0391 \\ 0.1055 & -0.3573 & 0.0427 & 0.5867 & 1.3112 & -0.1650 & -0.1565 & -0.0294 \\ 0 & 0.2077 & -0.4544 & -0.0777 & -0.1650 & 1.2661 & -0.2765 & 0.1806 \\ 0 & 0 & 0.2594 & -0.2947 & -0.1565 & -0.2765 & 0.5098 & -0.1182 \\ 0 & 0 & 0 & -0.0391 & -0.0294 & 0.1806 & -0.1182 & 0.5893 \end{bmatrix} \succ 0.$$

A.3 MATLAB Code

In this section of the Appendix we give some MATLAB code which may be use to generate the graphs required to repeat the experimental results in Chapters 3-5.

A.3.1 The ChordalGen Function

```
function [R,MC2,SMC2] = chordalGen(n,thresh)

%create a chordal graph with largest maximal clique size < thresh

%first create a connected tree and then merge adjacent maximal cliques
%together to produce denser chordal graphs.

A = diag(ones(n,1));
inTree = false(n,1);
```

```
10 inTree(1) = true;
11 v = 1:n;
12
13 for i=1:n-1
14     u=v(~inTree);
15     w = v(inTree);
16     r = randi(n-i,1);
17     s = randi(i,1);
18     A(u(r),w(s))=1;
19     A(w(s),u(r))=1;
20     inTree(u(r))=true;
21 end
22
23 %L2 = A - diag(sum(A));
24
25 MC = maximalCliques(A);
26 SMC = sum(MC);
27 [p,q] = size(MC);
28
29 k=1;
30 while max(SMC)<thresh
31     r = randi(q);
32     w =MC(:,r)'*MC(:,1:q);
33     w(r)=0;
34
35     t = find(w);
36     s = randi(length(t));
37     v = find(MC(:,t(s)));
38     z = find(MC(:,r));
39
```

```
40      keep = min([r,t(s)]);
41      del = max([r,t(s)]);
42      MC([v;z],keep)=1;
43      SMC(1,keep) = sum(MC(:,keep));
44
45      if del<q
46      MC(:,1:q-1) = MC(:,[1:del-1,del+1:q]);
47      SMC(:,1:q-1) = SMC(:,[1:del-1,del+1:q]);
48      q= q-1;
49      else
50      MC = MC(:,1:q-1);
51      SMC = SMC(:,1:q-1);
52      q = q-1;
53      end
54  end
55
56  R = zeros(n,n);
57
58  for i=1:q
59      indx = MC(:,i)>0;
60      R(indx,indx) =1;
61  end
62
63  MC2 =MC(:,1:q);
64  SMC2= SMC(1:q);
```

A.3.2 The TreeGen Function

```
 function [B,parents,children] = treeGen(n)

%returns the adjacecy matrix of a connected tree graph with n nodes. The
%nodes are ordered in topological order so that 1 is the root of the tree
%and then children have higher numbers.

%the function also returns the parents of each node in a vector and the
%children of each node in a cell(n,1)

A = diag(ones(n,1));
inTree = false(n,1);
inTree(1) = true;
v = 1:n;

for i=1:n-1
    u=v(~inTree);
    w = v(inTree);
    r = randi(n-i,1);
    s = randi(i,1);
    A(u(r),w(s))=1;
    A(w(s),u(r))=1;
    inTree(u(r))=true;
end

  numbered = zeros(n,1);
    numbered(1) =true;
    toporder = zeros(n,1);
    toporder(1,1) = n;

    for i =n-1:-1:1
```

```
31          u = find(numbered);
32          v = find(~numbered);
33          [a,b] = find(A(u,v)==1);
34          numbered(v(b(1))) =true;
35          toporder(v(b(1))) = i;
36      end
37
38      k=1;
39      reorder = zeros(n,1);
40      for i=n:-1:1
41         u = find(toporder==i);
42         reorder(k)= u;
43         k=k+1;
44      end
45
46
47  B= A(reorder,reorder);
48  %drawNet(B)
49  C =triu(B);
50  C = C-diag(ones(n,1));
51
52  %find parent of each node
53
54  parents = zeros(n,1);
55  for i =1:n
56       u = C(:,i);
57       if all(~u)
58       else
59       parents(i,1) =find(u==1);
60       end
```

```
61 end
62
63 children = cell(n,1);
64 for i =1:n
65      u = C(i,:);
66      if all(~u)
67      else
68      children{i,1} =find(u==1);
69      end
70 end
```

A.3.3 The SparseGen Function

```
1  function R = sparseGen(n,thresh)
2
3  %returns an adjacency matrix for a sparse (not necessarily chordal) graph
4  %the size of the largest maximal clique in the graph is bounded by thresh
5
6  A =zeros(1,n);
7  A(1,1)=1;
8
9  for i =1:n
10     r = randi(size(A,1)); %pick a maximal clique at random
11     u = find(A(r,:)==1); %find elements in maximal clique r
12     q = length(u);
13     k = randi(q,1);
14     v =randperm(q,k); %pick a subset of maximal clique r
15     if k==q
16        A(r,i) = 1;
```

```
17      else
18          z = zeros(1,n);
19          z([i,v]) =1;
20          A= [A;z];
21      end
22  end
23
24
25  while max(sum(A,2))<thresh
26      r = randperm(size(A,1),2); %select two maximal cliques at random
27      A(r(1),:) = or(A(r(1),:),A(r(2),:)); %and merge them together
28  end
29
30  R = zeros(n,n);
31
32  for i =1:size(A,1)
33      indx = find(A(i,:)==1);
34      R(indx,indx) =1;
35  end
```

A.3.4 KYP Example Generation Code

```
1
2  %generate a sparse linear system (A,B,C,D)
3  %for use in the continuous-time KYP experiments
4
5  n=30;   %number of states
6  q = 10; % number of inputs
7  p = 10;  % number of outputs
```

```
8  R= zeros(n,n);
9  H= ones(n,n);
10
11 density = 3/n;   %rho = density +1
12
13 S = sprand(n,n,density) + eye(n);
14 S = abs(S)>0;
15
16 A = rand(n,n)-0.5;
17 A = A.*S;
18
19 lambda = max(real(eig(A)));
20 A = A -(lambda+1)*eye(n);
21
22 S = sprand(n,q,density);
23 S = abs(S)>0;
24
25 B = rand(n,q)-0.5;
26 B = B.*S;
27
28 S = sprand(p,n,density);
29 S = abs(S)>0;
30
31 C = rand(p,n)-0.5;
32 C = C.*S;
33
34 S = sprand(p,q,density);
35 S = abs(S)>0;
36
37 D = rand(p,q)-0.5;
```

```
38 D= D.*S;
```

A.4 Maximum Determinant Completion Controller Example

Consider system $\dot{x}(t) = Ax(t) + Bu(t)$, with A and B as given on the following page. We next give an example of the results of designing a state feedback example using the Max-Det Completion approach discussed in the first half of Chapter 4.

Solving the SDP in (4.9) with $H = I$, $\beta_Q = \beta_R = 50$, $Q \in \mathbb{S}^n(E, 0)$ and $R \in \mathbb{R}^{m\times n}(F, 0)$ returned an optimal value $\gamma = 97.9$ and the following Q and R as solutions in 1.8s. Next we calculated the maximum determinant completion of Q using the clique factorisation formula.

The pair (W, R) is a feasible solution for the original SDP so we expect that the controller $K = RW^{-1}$ will stabilise the system. After thresholding the elements of K so that entries smaller than 1e-05 were set to zero we return the a controller which places the eigenvalues of the closed loop system in the left half plane.

$$A = \begin{bmatrix} -1.1932 & 0.0556 & 0 & -0.2378 & 0 & 0 & 0 & 0 & -0.4109 & 0 \\ -0.4210 & -0.3673 & 0 & 0.1831 & 0 & 0 & 0 & 0 & -0.3849 & 0 \\ 0 & 0 & -1.1653 & 0.4112 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.2953 & 0.2877 & 0.3538 & -0.7032 & 0 & 0 & 0 & 0.2731 & 0.4725 & 0 \\ 0 & 0 & 0 & 0 & -1.0591 & 0 & 0 & 0.4320 & 0 & 0.4785 \\ 0 & 0 & 0 & 0 & 0 & -0.4273 & 0 & -0.3231 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.9926 & -0.4796 & 0 & 0 \\ 0 & 0 & 0 & 0.1263 & -0.0622 & 0.0504 & 0.4075 & -0.4401 & 0 & 0 \\ -0.4366 & -0.2851 & 0 & 0.4483 & 0 & 0 & 0 & 0 & -0.5466 & 0 \\ 0 & 0 & 0 & 0 & 0.2043 & 0 & 0 & 0 & 0 & -0.4109 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & -0.0024 & 0 & 0 & 0 & 0.3070 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 - 0.4430 & 0 & 0.1670 & 0 & 0 & \\ 0 & 0 & 0 & 0 & -0.2860 & 0 & 0 & -0.3181 & 0 & 0 \\ -0.1908 & 0.4671 & 0 & -0.3781 & 0 & 0 & 0 & 0 & -0.4468 & 0 \\ 0 & 0 & -0.4249 & -0.4093 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} 5.2683 & 2.2640 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2.2640 & 6.5952 & 1.5513 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.5513 & 3.2030 & -0.7402 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.7402 & 3.5851 & 1.9375 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.9375 & 115.5920 & 11.7343 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 11.7343 & 49.6242 & 7.8195 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7.8195 & 6.3061 & -1.3774 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.3774 & 4.4443 & -4.2366 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4.2366 & 6.3275 & -0.2918 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.2918 & 15.7792 \end{bmatrix}$$

$$R = \begin{bmatrix} 3.1369 & 0.3858 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3.0882 & 0.6721 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2.6193 & 1.7687 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.7650 & 2.6218 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3.1602 & -0.0460 \end{bmatrix}$$

$$W = \begin{bmatrix} 5.2683 & 2.2640 & 0.5325 & -0.1231 & -0.0665 & -0.0068 & -0.0011 & 0.0002 & -0.0002 & 0.0000 \\ 2.2640 & 6.5952 & 1.5513 & -0.3585 & -0.1937 & -0.0197 & -0.0031 & 0.0007 & -0.0006 & 0.0000 \\ 0.5325 & 1.5513 & 3.2030 & -0.7402 & -0.4000 & -0.0406 & -0.0064 & 0.0014 & -0.0013 & 0.0001 \\ -0.1231 & -0.3585 & -0.7402 & 3.5851 & 1.9375 & 0.1967 & 0.0310 & -0.0068 & 0.0065 & -0.0003 \\ -0.0665 & -0.1937 & -0.4000 & 1.9375 & 115.5920 & 11.7343 & 1.8490 & -0.4039 & 0.3850 & -0.0178 \\ -0.0068 & -0.0197 & -0.0406 & 0.1967 & 11.7343 & 49.6242 & 7.8195 & -1.7080 & 1.6282 & -0.0751 \\ -0.0011 & -0.0031 & -0.0064 & 0.0310 & 1.8490 & 7.8195 & 6.3061 & -1.3774 & 1.3130 & -0.0606 \\ 0.0002 & 0.0007 & 0.0014 & -0.0068 & -0.4039 & -1.7080 & -1.3774 & 4.4443 & -4.2366 & 0.1954 \\ -0.0002 & -0.0006 & -0.0013 & 0.0065 & 0.3850 & 1.6282 & 1.3130 & -4.2366 & 6.3275 & -0.2918 \\ 0.0000 & 0.0000 & 0.0001 & -0.0003 & -0.0178 & -0.0751 & -0.0606 & 0.1954 & -0.2918 & 15.7792 \end{bmatrix}$$

$$K = \begin{bmatrix} 0.6690 & -0.1636 & -0.0320 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.2559 & 1.1819 & 0.4076 & -0.0032 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0124 & -0.0271 & 0.0507 & -0.0549 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.0548 & 0.5064 & 1.7666 & 1.0919 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.3161 & 1.3809 & 0.0063 \end{bmatrix}$$

A.5 H_∞ State Feedback Controller Synthesis Example

$$A = \begin{bmatrix} -1.7366 & -0.0765 & 0 & 0.6329 & 0 & 0 & 0.2263 & 0 & 0 & 0 \\ -0.5307 & -1.4198 & 0 & 0 & 0 & 0 & 0.1596 & 0 & 0 & 0 \\ 0 & 0 & -1.2217 & 0 & 0 & 0 & -0.2154 & 0 & 0 & 0 \\ 0.4410 & 0 & 0 & -0.7974 & -0.4720 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.4372 & -0.8516 & 0.2670 & 0 & 0 & -0.6079 & -0.6369 \\ 0 & 0 & 0 & 0 & -0.5673 & -1.4894 & 0 & 0.2377 & 0 & 0 \\ -0.2030 & 0.0777 & -0.1796 & 0 & 0 & 0 & -0.9385 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6220 & 0 & -0.4560 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.5094 & 0 & 0 & 0 & -0.7156 & 0 \\ 0 & 0 & 0 & 0 & -0.0299 & 0 & 0 & 0 & 0 & -1.3715 \end{bmatrix}$$

$$H = \begin{bmatrix} 0 & 0.0607 \\ 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.3112 & -0.4655 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.1363 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.1574 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.3664 & 0 & -0.2515 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1927 & 0.3214 & -0.4098 & 0.2376 & 0 \\ 0 & 0 & 0 & 0 & -0.2241 & 0 & 0 & 0.3382 \\ 0.2670 & 0 & 0.1556 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.3760 \\ 0 & 0 & 0 & 0 & 0 & -0.3204 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.4607 & 0 \end{bmatrix}$$

$$F = \begin{bmatrix} -0.0615 & 0 \\ 0 & -0.2402 \\ 0 & 0 \\ 0 & 0 \\ 0 & -0.2034 \\ 0 & 0 \\ -0.0163 & 0 \\ -0.0332 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \; C^T = \begin{bmatrix} 0 & 0.0595 \\ 0 & -0.2714 \\ -0.1381 & 0 \\ 0 & 0 \\ 0.0223 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.0622 & 0 \\ -0.1411 & 0 \\ 0 & 0 \end{bmatrix}, \; D^T = \begin{bmatrix} 0 & 0.0171 \\ 0 & 0 \\ 0.0842 & 0 \\ 0 & 0.0885 \\ 0 & 0 \\ 0 & 0 \\ 0.0667 & 0 \end{bmatrix}$$

Bibliography

[1] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," *Exploring artificial intelligence in the new millennium*, vol. 8, pp. 236–239, 2003.

[2] T. A. McKee and F. R. McMorris, *Topics in Intersection Graph Theory*. SIAM, 1999.

[3] D. J. Rose, R. E. Tarjan, and G. S. Lueker, "Algorithmic aspects of vertex elimination on graphs," *SIAM Journal of Computing*, vol. 5, pp. 266–283, 1976.

[4] D. J. Rose, "Triangulated graphs and the elimination process," *Journal of Mathematical Analysis and Applications*, vol. 32, pp. 597–609, 1970.

[5] J. Dahl, L. Vandenberghe, and V. Roychowdhury, "Covariance selection for non-chordal graphs via chordal embedding," *Optimization Methods and Software*, vol. 23, pp. 501–520, 2008.

[6] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.

[7] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[8] F. Alizadeh, "Interior point methods in semidefinite programming with applications to combinatorial optimization," *SIAM Journal on Optimization*, vol. 5, no. 1, pp. 13–51, 1995.

[9] H. Wolkowicz, R. Saigal, and L. Vandenberghe, *Handbook of semidefinite programming: theory, algorithms, and applications*. Springer Science & Business Media, 2000, vol. 27.

[10] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994.

[11] L. Vandenberghe and S. Boyd, "Applications of semidefinite programming," *Applied Numerical Mathematics*, vol. 29, no. 3, pp. 283–299, 1999.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[13] C. Scherer and S. Weiland, *Linear Matrix Inequalities in Control*. DISC course lecture notes, 2004.

[14] G. E. Dullerud and F. Paganini, *A Course in Robust Control Theory*. Springer, 2000.

[15] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

[16] H. Waki, S. Kim, M. Kojima, and M. Muramatsu, "Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity," *SIAM Journal of Optimization*, vol. 17, pp. 218–242, 2006.

[17] J. B. Lasserre, *Moments, Positive Polynomials and their Applications.* Imperial College Press, 2010.

[18] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization," in *Proceedings of the 3rd international symposium on Information processing in sensor networks.* ACM, 2004, pp. 46–54.

[19] A. Simonetto and G. Leus, "Distributed maximum likelihood sensor network localization," *Signal Processing, IEEE Transactions on*, vol. 62, no. 6, pp. 1424–1437, 2014.

[20] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 77–90, 2006.

[21] E. Dall'Anese, H. Zhu, and G. Giannakis, "Distributed optimal power flow for smart microgrids," *Smart Grid, IEEE Transactions on*, vol. 4, no. 3, pp. 1464–1475, 2013.

[22] D. Henrion, S. Tarbouriech, and D. Arzelier, "LMI approximations for the radius of the intersection of ellipsoids," *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 2, pp. 1759–1764, 1998.

[23] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz, "An interior-point method for semidefinite programming," *SIAM Journal on Optimization*, vol. 6, no. 2, pp. 342–361, 1996.

[24] E. D. Klerk, C. Roos, and T. Terlaky, "Initialization in semidefinite programming via a self-dual skew-symmetric embedding," *Operations Research Letters*, vol. 20, pp. 213–221, 1997.

[25] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton, "Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results," *SIAM J.Optimization*, vol. 8, pp. 746–768, 1998.

[26] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM Publications, 1994.

[27] J. F. Sturm, "Using SeDuMi 1.02, a matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, pp. 625–653, 1999.

[28] M. Yamashita, K. Fujisawa, and M. Kojima, "Implementation and evaluation of SDPA 6.0 (semidefinite programming algorithm 6.0)," *Optimization Methods and Software*, vol. 18, no. 4, pp. 491–505, 2003.

[29] K. Toh, M. Todd, and R. Tutuncu, "SDPT3: a matlab software package for semidefinite programming," *Optimization Methods and Software*, vol. 11, pp. 545–581, 1999.

[30] S. J. Benson and Y. Ye, "DSDP5: Software for semidefinite programming," Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, Tech. Rep. ANL/MCS-P1289-0905, September 2005.

[31] B. Borchers, "CSDP, A C library for semidefinite programming," *Optimization methods and Software*, vol. 11, no. 1-4, pp. 613–623, 1999.

[32] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[33] Z. Wen, D. Goldfarb, and W. Yin, "Alternating direction augmented lagrangian methods for semidefinite programming," *Mathematical Programming Computation*, vol. 2, no. 3-4, pp. 203–230, 2010.

[34] X.-Y. Zhao, D. Sun, and K.-C. Toh, "A Newton-CG augmented lagrangian method for semidefinite programming," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1737–1765, 2010.

[35] E. de Klerk, "Exploiting special structure in semidefinite programming: a survey of theory and applications," *European Journal of Operational Research*, vol. 201, pp. 1–10, 2010.

[36] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion I: general framework," *Society for Industrial and Applied Mathematics*, vol. 11, pp. 647–674, 2000.

[37] K. Fujisawa, M. Fukuda, M. Kojima, K. Nakata, and M. Yamashita, "SDPA-C (semidefinite programming algorithm – completion method) user's manual — version 6.10," *Research Report B-409, Dept. Math. and Comp. Sciences, Tokyo Institute of Technology*, 2004.

[38] S. J. Benson, Y. Ye, and X. Zhang, "Solving large-scale sparse semidefinite programs for combinatorial optimization," *SIAM Journal on Optimization*, vol. 10, no. 2, pp. 443–461, 2000.

[39] Y. Kanno, M. Ohsaki, K. Murota, and N. Katoh, "Group symmetry in interior-point methods for semidefinite program," *Optimization and Engineering*, vol. 2, no. 3, pp. 293–320, 2001.

[40] K. Gatermann and P. A. Parrilo, "Symmetry groups, semidefinite programs, and sums of squares," *Journal of Pure and Applied Algebra*, vol. 192, no. 1, pp. 95–128, 2004.

[41] R. Grone, C. R. Johnson, E. M. Sá, and H. Wolkowicz, "Positive definite completions of partial Hermitian matrices," *Linear Algebra and its Applications*, vol. 58, pp. 109–124, 1984.

[42] J. Agler, J. W. Helton, and S. McCullough, "Positive semidefinite matrices with a given sparsity pattern," *Linear Algebra and its Applications*, vol. 107, pp. 101–149, 1988.

[43] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita, "Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion," *Mathematical Programming Series B*, vol. 129, pp. 33–68, 2011.

[44] K. Nakata, K. Fujitsawa, M. Fukuda, M. Kojima, and K. Murota, "Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results," *Mathematical Programming Series B*, vol. 95, pp. 303–327, 2003.

[45] S. Boyd and Q. Yang, "Structured and simultaneous Lyapunov functions for system stability problems," *International Journal of Control*, vol. 49, pp. 2215–2240, 1989.

[46] M. Arcak, "Diagonal stability on cactus graphs and application to network stability analysis," *IEEE Transactions of Automatic Control*, vol. 56, pp. 2766–2777, 2011.

[47] A. Rantzer, "Distributed control of positive systems," *50th IEEE Conference on Decision and Control and European Control Conference*, pp. 6608–6611, 2011.

[48] R. Mason and A. Papachristodoulou, "Chordal sparsity, decomposing SDPs and the Lyapunov equation," *In proceedings of IEEE ACC 2014*, pp. 531 – 537, 2014.

[49] F. Gavril, "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 180–187, 1972.

[50] M. Yannakakis, "Computing the minimum fill-in is NP-complete," *SIAM Journal on Algebraic Discrete Methods*, vol. 2, no. 1, pp. 77–79, 1981.

[51] J. R. S. Blair and B. Peyton, "An introduction to chordal graphs and clique trees," in *Graph Theory and Sparse Matrix Computation*, A. George, J. R. Gilbert, and J. W. Liu, Eds. New York: Springer, 1993.

[52] D. R. Fulkerson and O. A. Gross, "Incidence matrices and interval graphs," *Pacific Journal of Mathematics*, vol. 15, pp. 835–855, 1965.

[53] R. E. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hyper graphs, and selectively reduce acyclic graphs," *SIAM Journal on Computation*, vol. 13, pp. 566–579, 1984.

[54] M. S. Andersen, J. Dahl, and L. Vandenberghe, "Implementation of non-symmetric interior-point methods for linear optimization over sparse matrix cones," *Mathematical Programming Computation*, vol. 2, pp. 167–201, 2010.

[55] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[56] N. Kakimura, "A direct proof for the matrix decomposition of chordal-structured positive semidefinite matrices," *Linear Algebra and its Applications*, vol. 433, pp. 819–823, 2010.

[57] J. B. Lasserre, "Convergent sdp-relaxations in polynomial optimization with sparsity," *SIAM J. Optim.*, vol. 17, pp. 822–843, 2006.

[58] M. S. Andersen, A. Hansson, S. K. Pakazad, and A. Rantzer, "Distributed robust stability analysis of interconnected uncertain systems," *Proceedings of the 51st IEEE Conference on Decision and Control (CDC), Hawaii*, vol. 1, pp. 1548–1553, 2012.

[59] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 1996.

[60] S. J. Hammarling, "Numerical solution of the stable, non-negative definite Lyapunov equation," *IMA Journal of Numerical Analysis*, vol. 2, pp. 303–323, 1982.

[61] G. Barker, A. Berman, and R. Plemmons, "Positive diagonal solutions to the Lyapunov equations," *Linear and Multilinear Algebra*, vol. 5, pp. 249–256, 1978.

[62] R. Balakrishnan and P. Paulraja, "Powers of chordal graphs," *J. Australian Mathematical Society*, vol. 35, pp. 211–217, 1983.

[63] R. D'Andrea and G. E. Dullerud, "Distributed control design for spatially interconnected systems," *Automatic Control, IEEE Transactions on*, vol. 48, no. 9, pp. 1478–1495, 2003.

[64] C. Langbort, R. S. Chandra, and R. D'Andrea, "Distributed control design for systems interconnected over an arbitrary graph," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1502–1519, 2004.

[65] V. Gupta, B. Hassibi, and R. M. Murray, "On the synthesis of control laws for a network of autonomous agents," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 6. IEEE, 2004, pp. 4927–4932.

[66] D. Šiljak and A. Zečević, "Control of large-scale systems: Beyond decentralized feedback," *Annual Reviews in Control*, vol. 29, no. 2, pp. 169–179, 2005.

[67] F. DörIer, M. R. Jovanovic, M. Chertkov, and F. Bullo, "Sparsity-promoting optimal wide-area control of power networks," *Power Systems, IEEE Transactions on*, vol. 29, no. 5, pp. 2281–2291, 2014.

[68] D. M. Stipanović, G. Inalhan, R. Teo, and C. J. Tomlin, "Decentralized overlapping control of a formation of unmanned aerial vehicles," *Automatica*, vol. 40, no. 8, pp. 1285–1296, 2004.

[69] F. Palacios-Quiñonero, J. Rodellar, and J. M. Rossell, "Sequential design of multi-overlapping controllers for longitudinal multi-overlapping systems," *Applied Mathematics and Computation*, vol. 217, no. 3, pp. 1170–1183, 2010.

[70] F. Palacios-Quiñonero, J. Rubió-Massegú, J. M. Rossell, and H. R. Karimi, "Discrete-time multioverlapping controller design for structural vibration control of tall buildings under seismic excitation," *Mathematical Problems in Engineering*, vol. 2012, 2012.

[71] F. Lin, M. Fardad, and M. R. Jovanovic, "Design of optimal sparse feedback gains via the alternating direction method of multipliers," *Automatic Control, IEEE Transactions on*, vol. 58, no. 9, pp. 2426–2431, 2013.

[72] V. Blondel and J. N. Tsitsiklis, "NP-hardness of some linear control design problems," *SIAM Journal on Control and Optimization*, vol. 35, no. 6, pp. 2118–2127, 1997.

[73] M. C. Rotkowitz and S. Lall, "A characterisation of convex problems in decentralised control," *IEEE transactions on Automatic Control*, vol. 51, pp. 274–286, 2006.

[74] P. Shah and P. A. Parrilo, "Optimal decentralized control over posets: A state-space solution for state-feedback," *Automatic Control, IEEE Transactions on*, vol. 58, no. 12, pp. 3084–3096, 2013.

[75] G. Fazelnia, R. Madani, and J. Lavaei, "Convex relaxation for optimal distributed control problem," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 896–903.

[76] K. Dvijotham, E. Todorov, and M. Fazel, "Convex structured controller design in finite horizon," *Control of Network Systems, IEEE Transactions on*, vol. 2, no. 1, pp. 1–10, 2015.

[77] C. Scherer, P. Gahinet, and M. Chilali, "Multiobjective output-feedback control via lmi optimization," *Automatic Control, IEEE Transactions on*, vol. 42, no. 7, pp. 896–911, 1997.

[78] D. Šiljak and D. Stipanović, "Robust stabilization of nonlinear systems: the lmi approach," *Mathematical problems in Engineering*, vol. 6, no. 5, pp. 461–493, 2000.

[79] L. Markenzon, O. Vernet, and L. H. Araujo, "Two methods for the generation of chordal graphs," *Annals of Operation Research*, vol. 157, pp. 47–60, 2008.

[80] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Mathematical programming*, vol. 39, no. 2, pp. 117–129, 1987.

[81] P. A. Parrilo, "Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization," Ph.D. dissertation, Caltech, Pasadena, CA, 2000.

[82] G. Blekherman, P. A. Parrilo, and R. R. Thomas, *Semidefinite optimization and convex algebraic geometry.* Society for industrial and applied mathematics, 2013.

[83] J. Lofberg and P. A. Parrilo, "From coefficients to samples: a new approach to sos optimization," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 3. IEEE, 2004, pp. 3154–3159.

[84] B. Sturmfels, "Polynomial equations and convex polytopes," *American Mathematical Monthly*, pp. 907–922, 1998.

[85] M. Kojima, S. Kim, and H. Waki, "Sparsity in sums of squares of polynomials," *Research Roprts on Mathematical Computing Sciences Series B*, vol. 15, pp. 697–719, 2003.

[86] S. Kim, M. Kojima, and H. Waki, "Generalized lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems," *SIAM Journal of Optimization*, vol. 15, pp. 697–719, 2005.

[87] N. Shor, "Class of global minimum bounds of polynomial functions," *Cybernetics and Systems Analysis*, vol. 23, no. 6, pp. 731–734, 1987.

[88] A. Papachristodoulou and S. Prajna, "On the construction of Lyapunov functions using the sum of squares decomposition," in *Decision and Control,*

2002, Proceedings of the 41st IEEE Conference on, vol. 3. IEEE, 2002, pp. 3482–3487.

[89] H. Zhu and G. Giannakis, "Multi-area state estimation using distributed SDP for nonlinear power systems," in *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*. IEEE, 2012, pp. 623–628.

[90] S. Khoshfetrat Pakazad, M. S. Andersen, and A. Hansson, "Distributed solutions for loosely coupled feasibility problems using proximal splitting methods," *Optimization Methods and Software*, vol. 30, no. 1, pp. 128–161, 2015.

[91] S. K. Pakazad, A. Hansson, M. S. Andersen, and A. Rantzer, "Distributed semidefinite programming with application to large-scale system analysis," *arXiv preprint arXiv:1504.07755*, 2015.

[92] ——, "Distributed robustness analysis of interconnected uncertain systems using chordal decomposition," *arXiv preprint arXiv:1402.2066*, 2014.

[93] M. J. Wainwright and M. I. Jordan, "Variational inference in graphical models: The view from the marginal polytope," *proceedings of the annual Allerton conference on communication control and computing*, vol. 41, no. 2, pp. 961–971, 2003.

[94] ——, "Semidefinite relaxations for approximate inference on graphs with cycles," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-03-1226, Jan 2003.

[95] R. J. McEliece, D. J. C. MacKay, and J.-F. Cheng, "Turbo decoding as an instance of pearl's belief propagation algorithm," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 2, pp. 140–152, 1998.

[96] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.

[97] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," *Information Theory, IEEE Transactions on*, vol. 51, no. 7, pp. 2282–2312, 2005.

[98] D. Sontag and T. S. Jaakkola, "New outer bounds on the marginal polytope," *Proceedings of Neural Information Processing Systems*, pp. 1393–1400, 2007.

[99] L. Vandenberghe, S. Boyd, and S.-P. Wu, "Determinant maximization with linear matrix inequality constraints," *SIAM journal on matrix analysis and applications*, vol. 19, no. 2, pp. 499–533, 1998.

[100] M. J. Wainwright and M. I. Jordan, "Log-determinant relaxation for approximate inference in discrete markov random fields," *Signal Processing, IEEE Transactions on*, vol. 54, no. 6, pp. 2099–2109, 2006.

[101] L. Vandenberghe and M. S. Andersen, "Chordal graphs and semidefinite optimization," *Foundations and Trends in Optimization*, vol. 1, no. 4, pp. 241–433, 2015.