

Polynomial systems: Graphical structure, Geometry, and Applications

by

Diego Cifuentes

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2018

© Massachusetts Institute of Technology 2018. All rights reserved.

Author
Department of Electrical Engineering and Computer Science
January 26, 2018

Certified by
Pablo A. Parrilo
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Polynomial systems: Graphical structure, Geometry, and Applications

by

Diego Cifuentes

Submitted to the Department of Electrical Engineering and Computer Science
on January 26, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Solving systems of polynomial equations is a foundational problem in computational mathematics, that has several applications in the sciences and engineering. A closely related problem, also prevalent in applications, is that of optimizing polynomial functions subject to polynomial constraints. In this thesis we propose novel methods for both of these tasks. By taking advantage of the graphical and geometrical structure of the problem, our methods can achieve higher efficiency, and we can also prove better guarantees.

Various problems in areas such as robotics, power systems, computer vision, cryptography, and chemical reaction networks, can be modeled by systems of polynomial equations, and in many cases the resulting systems have a simple sparsity structure. In the first part of this thesis we represent this sparsity structure with a graph, and study the algorithmic and complexity consequences of this graphical abstraction. Our main contribution is the introduction of a novel data structure, chordal networks, that always preserves the underlying graphical structure of the system. Remarkably, many interesting families of polynomial systems admit compact chordal network representations (of size linear in the number of variables), even though the number of components is exponentially large. Our methods outperform existing techniques by orders of magnitude in applications from algebraic statistics and vector addition systems.

We then turn our attention to the study of graphical structure in the computation of matrix permanents, a classical problem from computer science. We provide a novel algorithm that requires $\tilde{O}(n 2^\omega)$ arithmetic operations, where ω is the treewidth of its bipartite adjacency graph. We also investigate the complexity of some related problems, including mixed discriminants, hyperdeterminants, and mixed volumes. Although seemingly unrelated to polynomial systems, our results have natural implications on the complexity of solving sparse systems.

The second part of this thesis focuses on the problem of minimizing a polynomial function subject to polynomial equality constraints. This problem captures many important applications, including Max-Cut, tensor low rank approximation, the triangulation problem, and rotation synchronization. Although these problems are nonconvex, tractable semidefinite programming (SDP) relaxations have been proposed. We introduce a methodology to derive more efficient (smaller) relaxations, by leveraging the geometrical structure of the underlying variety. The main idea behind our method is to describe the variety with a generic set of samples, instead of relying on an algebraic description. Our methods are particularly appealing for varieties that are easy to sample from, such as $SO(n)$, Grassmannians, or rank k tensors. For arbitrary varieties we can take advantage of the tools from numerical algebraic geometry.

Optimization problems from applications usually involve parameters (e.g., the data), and

there is often a natural value of the parameters for which SDP relaxations solve the (polynomial) problem exactly. The final contribution of this thesis is to establish sufficient conditions (and quantitative bounds) under which SDP relaxations will continue to be exact as the parameter moves in a neighborhood of the original one. Our results can be used to show that several statistical estimation problems are solved exactly by SDP relaxations in the low noise regime. In particular, we prove this for the triangulation problem, rotation synchronization, rank one tensor approximation, and weighted orthogonal Procrustes.

Thesis Supervisor: Pablo A. Parrilo

Title: Professor of Electrical Engineering and Computer Science

To my wife

Acknowledgments

My greatest acknowledgements go to my advisor, Pablo Parrilo. It has been a pleasure to work with him for four years. Pablo is an outstanding researcher, a passionate speaker, a highly motivating teacher, and an extraordinary mentor. His extensive knowledge always impressed me, and I thank him for the hundreds of references he gave me. His keen instinct was also very helpful, pointing me in the right path several times.

I thank my collaborator and thesis reader, Rekha Thomas. I am grateful for the detailed feedback and constant support she always provided. I thank Caroline Uhler, who kindly accepted to be my second thesis reader. I also thank Sameer Agarwal, with whom I had a very exciting collaboration.

I thank my friends from LIDS: James, Hamza, Frank, Shreya, Dalton, Jason, Chenyang, Dogyoon, Christos. They have been a great support during all these years. I thank my other friends at MIT, and most specially my previous roommates: Ardavan, Julian, Karthik. I also want to thank my Colombian friends in the Boston area: Aleja, Andrés, Stephy, Leonor, Juan José, Felipe, Silke, Marco, Darío, Pedro, Gigi, Germán, Diana, Catalina, Paula. They have been very close friends, and we enjoyed several football matches together in these PhD years.

Finally, I thank my caring family for always being there for me, despite the distance. I most specially thank my mother, Martha Rocío, for her utmost dedication to me and my brother. Last but not least, I thank my wife María Baez. She has been with me during all these years at MIT, has seen the different stages of my dissertation, has comforted me during the struggling times, and has rejoiced with me in the happy moments. María is the best thing MIT has given me.

The research in this thesis was funded in part by grant AFOSR FA9550-11-1-0305 .

Contents

List of Figures	15
List of Tables	17
1 Introduction	19
1.1 Outline and contributions	22
1.2 Notation	24
2 Background	25
2.1 Algebraic preliminaries	25
2.1.1 Ideals and varieties	25
2.1.2 Solving polynomial systems	27
2.1.3 Polynomial optimization	29
2.2 Treewidth and chordality	31
2.2.1 Chordal graphs and chordal completions	31
2.2.2 Tree decompositions	33
I Polynomial systems and graphical structure	36
3 Chordal networks of polynomial ideals	37
3.1 Introduction	37
3.2 Chordal networks	44
3.3 The chordally zero-dimensional case	47
3.3.1 Triangular sets	48

3.3.2	Chordal triangularization	49
3.3.3	Algorithm analysis	50
3.3.4	Radical and irreducible decompositions	53
3.3.5	Complexity	54
3.4	Computing with chordal networks	56
3.4.1	Elimination	57
3.4.2	Counting solutions	57
3.4.3	Sampling solutions	58
3.4.4	Radical membership	59
3.5	Monomial ideals	65
3.5.1	Chordal triangularization	65
3.5.2	Computing with chordal networks	67
3.6	The general case	69
3.6.1	Regular chains	69
3.6.2	Regular systems	71
3.6.3	Chordal triangularization	72
3.6.4	Computing with chordal networks	74
3.7	Examples	75
3.7.1	Commuting birth and death ideal	75
3.7.2	Lattice walks	77
3.7.3	Finite state diagram representation	78
3.8	Additional proofs	80
3.8.1	Proofs from Section 3.3	80
3.8.2	Proofs from Section 3.4	82
3.8.3	Proofs from Section 3.6	84
4	Graphical structure in permanents and related problems	87
4.1	Introduction	87
4.2	Graph representations of a sparse matrix	91
4.3	Column decompositions	95
4.3.1	Partial permanent	96

4.3.2	Decomposition algorithm for the permanent	97
4.3.3	Permanent of all submatrices	99
4.3.4	Recursion formula	100
4.3.5	Computing the determinant	103
4.4	Bipartite decompositions	108
4.4.1	Bipartite decomposition algorithm	109
4.4.2	Recursion formula	111
4.4.3	Complexity analysis	114
4.5	Mixed discriminant and higher dimensions	116
4.5.1	Mixed discriminant	116
4.5.2	Higher dimensions	120
4.6	Mixed volume of zonotopes	122
4.6.1	Mixed volumes and permanents	123
4.6.2	Graph representation	124
4.6.3	Hardness result	126
4.6.4	Application to sparse polynomial systems	128
II	Polynomial optimization	130
5	Sampling algebraic varieties for SOS optimization	131
5.1	Introduction	131
5.2	Preliminaries	135
5.2.1	Algebraic geometry	135
5.2.2	Sampling varieties	136
5.2.3	SOS certificates on varieties	137
5.3	Computing pre-certificates	138
5.3.1	Sampling SDP	138
5.3.2	Poisedness implies correctness	140
5.3.3	Reducing complexity	141
5.4	Verifying sampling pre-certificates	143
5.4.1	Genericity	143

5.4.2	Identity testing	144
5.5	Selecting the samples	144
5.5.1	Poisedness again	145
5.5.2	How many samples	146
5.5.3	The irreducible case	147
5.5.4	Verifying the number of samples	148
5.5.5	Reducible varieties	149
5.6	Computing sampling certificates	150
5.7	Examples	151
5.7.1	Nilpotent matrices	152
5.7.2	Weighted orthogonal Procrustes	153
5.7.3	Trace ratio problem	154
5.7.4	Low rank approximation	156
5.7.5	Certifying infeasibility	157
5.7.6	Amoeba membership	158
6	Local stability of SDP relaxations	161
6.1	Introduction	161
6.2	Formalizing the problem	165
6.2.1	Problem statement	166
6.2.2	Lagrange multipliers and zero-duality-gap	167
6.3	Continuity of Lagrange multipliers	168
6.3.1	A first stability result	169
6.3.2	Restricted Slater	170
6.3.3	Stability under Assumption RS	171
6.4	An easy first case	173
6.4.1	Preparing the problem	174
6.4.2	Small multipliers	175
6.4.3	Guaranteed region of zero-duality-gap	176
6.5	The general case	178
6.6	Proof of main theorem	183

6.6.1	The implicit function theorem	184
6.7	Applications	186
6.7.1	Estimation problems with strictly convex objective	186
6.7.2	Stability of unconstrained SOS	189
6.7.3	Noisy Euclidean distance matrix completion	192
6.8	Additional proofs	194
7	Summary	197
	Bibliography	199

List of Figures

2-1	Chordal completion and elimination tree	32
2-2	Tree decomposition of a graph	34
3-1	Chordal network for the 3-chromatic ideal of a cycle	39
3-2	Chordal network for the edge ideal of a tree.	40
3-3	Chordal network for the ideal of adjacent minors	41
3-4	G -chordal network, where G is the graph from Figure 2-1a	45
3-5	Chordal triangularization	50
3-6	Sketch of the radical ideal membership test	61
3-7	Top-dimensional part of a chordal network	66
3-8	Chordal triangularization (positive dimension)	73
3-9	Top-dimensional part of the chordal network from Figure 3-8.	74
3-10	Illustration of the card problem using 5 decks.	77
3-11	State diagrams for ideals of adjacent minors of a matrix.	79
3-12	State diagram for the ideal of cyclically adjacent minors	79
4-1	Complexity of permanents and related problems	89
4-2	Bipartite graph, symmetrized graph, and column graph of a matrix	92
4-3	Structured matrix and its bipartite graph	95
4-4	Tree decomposition of a column graph	95
4-5	Tree decomposition of a bipartite graph	108
5-1	Infeasibility certificate for the cyclic 9-roots problem	160

6-1	Duality gap in a nearest point problem	163
6-2	Lift of the plane curve $y_2^2 = y_1^3$	164
6-3	Conditions that imply zero-duality-gap nearby $\bar{\theta}$	170
6-4	Guaranteed region of zero-duality-gap	176

List of Tables

3.1	Gröbner bases vs. regular chains	71
3.2	Performance of chordal triangularization on the ideals I^{n_1, n_2}	76
3.3	Irreducible components of the ideals $I^{n_1, 1}$	76
3.4	High dimensional irreducible components of the ideals $I^{n_1, 1}$	77
3.5	Performance of radical ideal membership tests	78
5.1	SOS relaxations for the weighted orthogonal Procrustes problem	154
5.2	SOS relaxations for the trace ratio problem	155

Chapter 1

Introduction

This thesis is concerned with computational aspects of *multivariate polynomial equations*, and their applications in sciences and engineering. Specifically, we approach two basic problems from computational mathematics: *solving* systems of polynomial equations, and *optimizing* polynomial equations subject to polynomial constraints. Both of these problems arise across sciences and engineering, in areas as diverse as robotics, power systems, computer vision, cryptography, statistics, chemical reaction networks, dynamical systems, game theory, machine learning. Although there is a plethora of algorithms and tools to approach these problems, current techniques do not scale well as the number of variables grows. Consequently, solving large scale problems arising in practice demands methods that can take account of the *structure* of the problem. A central topic of this thesis is the study of special structure in systems of polynomial equations and polynomial optimization problems, with emphasis on effective methods. We explore how graphical and geometrical considerations may enable both faster algorithms and better guarantees.

The first part of this thesis concerns the problem of finding complex solutions to systems of polynomial equations. The solution set of such a system is known as an *algebraic variety*. The history of algebraic varieties dates back to centuries, being the fundamental object of study of algebraic geometry. Computational aspects of algebraic varieties have been explored since the 1960s, and belong to the area of computational algebraic geometry [46, 47]. The most basic problem of interest is the feasibility of a polynomial system, or equivalently, the problem of deciding whether the associated (complex) variety is nonempty. Among the dif-

ferent approaches for solving polynomial systems, *Gröbner bases* is the most widely used [46]. Although Gröbner bases methods are available in most computer algebra systems, their high computational cost limits its applicability to systems with only tens of variables.

This thesis addresses polynomial system solving from a new angle. The key insight is that systems coming from applications often have a simple sparsity structure. This holds, in particular, in applications such as power systems, sensor networks, and chemical reaction networks. This sparsity structure can be represented with a graph that describes the interactions among the variables. Although it is natural to expect that the complexity of solving the polynomial system should depend on the underlying graphical structure, existing techniques (e.g., Gröbner bases) completely ignore it. In this thesis we propose new methods for solving polynomial systems that exploit this graphical structure, as well as a novel data structure to represent the associated variety.

The idea of using graphical structure in polynomial systems follows a general paradigm that has been successful in several other areas, including numerical linear algebra [111], discrete and continuous optimization [136], graphical models [87] and constraint processing [50]. The notions of *chordality* and *treewidth* prevail in all these areas, and they also play a fundamental role in our treatment of polynomial systems. The relevance of chordality is already seen in the special case of degree one polynomials, namely linear equations. It is well-known that symmetric Gaussian elimination does not introduce nonzero entries in a sparse matrix precisely when the adjacency graph of the matrix is chordal [111]. The case of arbitrary polynomial systems is slightly more complex, and an interplay between graphical structure and geometric properties of the variety will govern the complexity of our methods.

The study of structured polynomial systems is tightly connected to lattice polytopes [127]. In particular, the number of solutions of a polynomial system is always upper bounded by a quantity known as the *mixed volume* of their associated Newton polytopes. Computing such a bound is a crucial step in *homotopy continuation* methods [124]. We point out that mixed volume encompasses the matrix *permanent* as a special instance, and consequently, it is computationally hard in general. Our study of graphical structure in polynomial system thus naturally leads to computing mixed volumes and permanents under appropriate graphical structure. For the case of permanents, we provide a novel algorithm that takes $\tilde{O}(n2^\omega)$ arithmetic operations, where ω is the treewidth of the graph. This algorithm naturally extends

to compute generalizations of the permanent to higher dimensional arrays. On the other hand, we show that mixed volumes remain hard to compute when the treewidth is bounded. Our results have natural complexity implications about solving sparse polynomial systems.

The second part of this thesis is devoted to polynomial optimization problems. Concretely, we consider the problem of minimizing polynomial functions over real algebraic varieties. There are two main kinds of methods used for such problems. The first class consists of general nonlinear optimization methods [19] (e.g., gradient descend, Newton’s method) which, though efficient, only converge to a local optima. The second class, tailored toward the polynomial case, relies on a hierarchy of *semidefinite programming* (SDP) relaxations coming from the *sum-of-squares* (SOS) method [105]. Our focus is in the latter class.

The SOS method, also known as Lasserre hierarchy, is a general approach to construct SDP relaxations for polynomial optimization problems [22, 85]. In its simplest form, the SOS method attempts to find a certificate of the (global) nonnegativity of a polynomial. A natural class of certificates to consider is based in rewriting the polynomial as a sum-of-squares (SOS). The close connection between SOS polynomials and positive semidefinite (PSD) matrices leads to a systematic procedure for certifying polynomial nonnegativity using SDP’s. The history of nonnegativity and SOS dates back to Hilbert, but it was only in the past two decades that their connections with optimization and SDP’s became apparent, and this is a subject undergoing intense study. Even though the SOS method can be applied to arbitrary polynomial optimization problems, the case in which only equality constraints are present (i.e., when the feasible set is a variety) is rather special. The second part of this thesis studies how the geometric structure of the underlying variety contributes both to more efficient methods and to better guarantees on the quality of the relaxation.

The SOS method relies on writing polynomial functions as sum-of-squares. Importantly, any algebraic variety has a natural equivalence relation that allows us to reduce the space of functions we need to consider. Namely, we consider two polynomials equivalent if they take the same values on the given variety. The space of equivalence classes is known as the *coordinate ring* of the variety. Working modulo the coordinate ring has two well-known advantages: it leads to a smaller SDP since the space of functions to consider is smaller, and the associated relaxation is stronger since it takes into account the geometry of the variety. Despite the desirability of coordinate ring relaxations, the only previously known method to derive them

relied on Gröbner bases [104], and thus its practical applicability was rather limited. We propose the first coordinate ring relaxation independent of Gröbner bases. Our main insight is that having sufficiently many samples of the variety allows us to understand the structure of its coordinate ring.

Finally, we consider the problem of finding the nearest point to an algebraic variety. This problem arises, in particular, in statistical estimation problems for which the model is defined by algebraic equations (e.g., the model consists of low rank matrices). Indeed, this class encompasses estimation problems such as low rank tensor approximation, the triangulation problem from computer vision, sensor network localization, camera resectioning, and approximate GCD. We study the behavior of SOS relaxations in the *low noise* regime, i.e., in the case where the point is sufficiently close to the variety. We establish sufficient conditions that guarantee that the relaxation solves the problem *exactly* under low noise, as well as quantitative bounds on the amount of noise tolerated by the relaxation. More generally, we analyze the *stability* properties of SDP relaxations for parametrized polynomial optimization problems. Our methods, in particular, generalize previously known results about exactness of SDP relaxations in camera triangulation [1] and rotation synchronization problems [112].

1.1 Outline and contributions

Chapter 2 introduces some tools from algebra and graph theory that are used throughout this document. In particular, we recall notions such as algebraic variety, polynomial ideal, SOS certificate, chordality, and treewidth. The remainder of the thesis is divided into two parts, each containing two chapters.

Part I: Graphical structure and polynomial systems

Chapter 3

The main contribution of this chapter is the introduction of a new data structure to represent structured polynomial systems: *chordal networks*. Unlike traditional methods (e.g., Gröbner bases), chordal networks always preserve the underlying graphical structure of the system. We provide an algorithm to compute chordal network representations for arbitrary polynomial systems. Remarkably, for several interesting families of polynomial systems (with

exponentially large Gröbner bases) the obtained chordal network has size proportional to the number of variables. Chordal networks give a computationally convenient description of the variety, that can be efficiently processed to compute properties such as dimension, cardinality, equidimensional components, and radical ideal membership. Preliminary implementation of our methods show *orders of magnitude reduction* against state-of-the-art algorithms in cases from algebraic statistics and vector addition systems. This chapter is based on [41].

Chapter 4

Computing the permanent of a $n \times n$ matrix is a classical problem in computer science, known to be #P-hard in general. Chapter 4 studies methods to efficiently compute the permanent of matrices with *structured sparsity*. We also study some related quantities from convex geometry: mixed discriminants, hyperdeterminants, and mixed volumes. Our results rely on describing the sparsity structure of the matrix (or array) in terms of a graph, and exploiting the chordal completions of this graph. Our method requires $\tilde{O}(n2^\omega)$ operations to compute permanents, where ω is the treewidth of the graph, and $\tilde{O}(n^2 + n3^\omega)$ operations for mixed discriminants and hyperdeterminants. We further show that mixed volume computation remains hard for bounded treewidth. This last result implies that solving polynomial systems of bounded treewidth is also #P-hard in the generic case. This chapter is based on [38].

Part II: Polynomial optimization

Chapter 5

Consider the problem of minimizing a polynomial function over an algebraic variety. SDP relaxations provide a tractable alternative for these nonconvex problems. This chapter introduces a new methodology for obtaining such relaxations. Unlike previous techniques, which rely on an algebraic description, we represent the variety with a generic set of complex *samples*. This approach depends only on the geometry of the variety, avoiding representation issues such as multiplicity and choice of generators. It also takes advantage of the coordinate ring structure to reduce the size of the corresponding SDP, and it is the first relaxation independent of Gröbner bases that uses this structure. Our methods are particularly appealing for varieties that are easy to sample from but for which the defining equations are complicated, such as

$SO(n)$, Grassmannians, or rank k tensors. For arbitrary varieties we can obtain the samples by using the tools of numerical algebraic geometry. In this way we connect the areas of SOS optimization and numerical algebraic geometry. This chapter is based on [42].

Chapter 6

Consider semidefinite programming (SDP) relaxations of *parametrized* polynomial optimization problems. Often times in applications there is a natural value of the parameters for which the relaxation will solve the problem exactly. Chapter 6 establishes conditions (and quantitative bounds) under which the relaxation will continue to be exact as the parameter moves in a neighborhood of the original one. Our framework captures several statistical estimation problems, such as low rank approximation, camera triangulation, rotation synchronization, approximate matrix completion, and approximate GCD. In these applications, a solution is easy under noiseless observations, and our results guarantee that the SDP relaxation will continue to solve the problem in the low noise regime. This chapter is based on [37].

1.2 Notation

The following table summarizes some of the common notations used throughout this thesis.

polynomials	$\mathbb{K}[x_1, \dots, x_n]$	polynomials in variables x_1, \dots, x_n with coefficients in \mathbb{K}
	$\mathbf{V}(h)$	zero set (variety) of a polynomial set h
	$\langle h \rangle$	ideal generated by a polynomial set h
	\sqrt{I}	radical ideal of I
matrices	\mathcal{S}^n	space of $n \times n$ real symmetric matrices
	id_n	$n \times n$ identity matrix
	$A \succ B$ ($A \succeq B$)	$A - B$ is positive (semi)definite
	$A \bullet B$	trace inner product
	$\ A\ , \ A\ _F$	spectral norm, and Frobenius norm
other	$ S $	cardinality of a set S
	$S = S_1 \sqcup S_2$	S is the union of the disjoint sets S_1 and S_2
	$\tilde{O}(f(n) 2^{g(\omega)})$	same as $O(f(n) 2^{g(\omega)})$, but ignores polynomial factors in ω

Chapter 2

Background

2.1 Algebraic preliminaries

2.1.1 Ideals and varieties

Let \mathbb{K} be a field, and let $\mathbb{K}[x] = \mathbb{K}[x_0, \dots, x_{n-1}]$ be the ring of polynomials in n variables with coefficients in \mathbb{K} . In applications we often focus in the case $\mathbb{K} = \mathbb{Q}$ (rational numbers) and the case $\mathbb{K} = \mathbb{R}$ (reals numbers). Let $\overline{\mathbb{K}}$ be the algebraic closure of \mathbb{K} . For instance, $\overline{\mathbb{K}} = \mathbb{C}$ (complex numbers) if either $\mathbb{K} = \mathbb{Q}$ or \mathbb{R} .

Consider a *polynomial system* $F = \{f_1, f_2, \dots, f_m\} \subseteq \mathbb{K}[x]$. We may associate two important objects to such a system: one algebraic (ideal) and one geometric (variety). The *generated ideal* of F is

$$I = \langle F \rangle := \{r_1 f_1 + r_2 f_2 + \dots + r_m f_m : r_i \in \mathbb{K}[x]\},$$

and the *variety* or *zero set* of F is

$$\mathbf{V} = \mathbf{V}_{\overline{\mathbb{K}}}(F) := \{x \in \overline{\mathbb{K}}^n : f(x) = 0 \text{ for all } f \in F\}.$$

Observe that $\mathbf{V}_{\overline{\mathbb{K}}}(F) = \mathbf{V}_{\overline{\mathbb{K}}}(\langle F \rangle)$.

Definition 2.1. A *polynomial ideal* is a set $I \subseteq \mathbb{K}[x]$ that is generated by some polynomial system. Similarly, an *algebraic variety* $\mathbf{V} \subseteq \overline{\mathbb{K}}^n$ is the zero set of some polynomial system.

There is a nice interplay between ideals and varieties. In particular, the *weak Hilbert's Nullstellensatz* establishes that $\mathbf{V}_{\overline{\mathbb{K}}}(F) = \emptyset$ (the system is infeasible) if and only if $1 \in \langle F \rangle$. The complete correspondence between ideals and varieties is given by the *strong Hilbert's Nullstellensatz*, as stated below.

Definition 2.2. The *vanishing ideal* of a set $S \subseteq \overline{\mathbb{K}}^n$ is

$$\mathbf{I}_{\overline{\mathbb{K}}}(S) := \{f \in \mathbb{K}[x] : f(s) = 0 \text{ for all } s \in S\}.$$

The *radical* of an ideal $I \subseteq \mathbb{K}[x]$ is the following ideal

$$\sqrt{I} := \{p \in \mathbb{K}[x] : p^k \in I \text{ for some integer } k \geq 1\}.$$

Theorem 2.1. (*Nullstellensatz*) Let $I \subseteq \mathbb{K}[x]$ be an ideal. Then

$$I \subseteq \mathbf{I}_{\overline{\mathbb{K}}}(\mathbf{V}_{\overline{\mathbb{K}}}(I)) = \sqrt{I}.$$

In other words, \sqrt{I} is the largest ideal that has the same zero set as I .

Remark 2.1. We say that I is *radical* if $\sqrt{I} = I$. The Nullstellensatz implies a one-to-one correspondence between radical ideals and varieties.

Finally, we recall that there is a canonical way to decompose an algebraic variety into simpler sets. We say that a variety $\mathbf{V} \subseteq \overline{\mathbb{K}}^n$ is *irreducible* if it is not the union of two proper varieties. Any variety can be decomposed in a unique way in the form

$$\mathbf{V} = \mathbf{V}_1 \cup \dots \cup \mathbf{V}_r, \quad \text{where } \mathbf{V}_i \text{ is irreducible and } \mathbf{V}_i \not\subseteq \mathbf{V}_j \text{ for } i \neq j. \quad (2.1)$$

The varieties \mathbf{V}_i are called the *irreducible components* of \mathbf{V} .

Remark 2.2. The algebraic analog of irreducible varieties are *prime* ideals. A prime ideal I is one such that $ab \in I \iff a \in I \text{ or } b \in I$.

2.1.2 Solving polynomial systems

Many different problems in sciences and engineering can be stated as *solving* systems of polynomial equations, in areas such as graph theory, robotics, computer vision, power networks, cryptography, game theory, and chemistry. As an illustration, we now show how to pose the q -coloring problem in terms of polynomial systems.

Example 2.1 (Coloring ideal). Let $\mathcal{G} = (V, E)$ be a graph. The proper q -colorings of G are in bijection with the solutions of the following system of equations (see e.g., [49]):

$$x_i^q - 1 = 0 \quad i \in V \quad (2.2a)$$

$$x_i^{q-1} + x_i^{q-2}x_j + \cdots + x_ix_j^{q-2} + x_j^{q-1} = 0 \quad ij \in E \quad (2.2b)$$

Although “solving a system” might mean different things depending on the application, it generally entails understanding the structure of the associated variety. Given a polynomial system $F = \{f_1, \dots, f_m\} \subseteq \mathbb{K}[x]$, the most basic problem of interest is the following:

Feasibility. Determine if the variety of F is nonempty, and if so compute a solution.

In some applications one requires further information about the underlying variety. Some other problems of concern are:

Counting. Determine the number of solutions (it might be infinite).

Dimension. Determine the dimension of the variety.

Components. Describe the irreducible decomposition of the variety.

The field of *computational algebraic geometry* studies effective methods to answer all of these questions. More generally, computational algebraic geometry concerns the problems of *representing* and *manipulating* algebraic varieties effectively on a computer.

There are a few different approaches for solving polynomial systems. We now briefly describe three of the most commonly used methods.

Gröbner bases. This is a symbolic method introduced by Buchberger [30] in 1965. A Gröbner bases of an ideal is a very special generating set, which generalizes the concept of a

matrix in “reduced echelon form”. These special bases can be computed for any system using sparse linear algebra. Gröbner bases were the first method proposed for solving polynomial systems. It continues to be the most commonly used technique, being implemented in most computer algebra programs. The book [46] provides a very nice introduction to this theory.

Triangular sets. This is also a symbolic method, introduced by Wu [149] in 1978. The starting point for this theory is that any algebraic variety can be decomposed into “simpler” varieties, that can be described by polynomial sets with “triangular form”. These methods are often used in conjunction with Gröbner bases for the computation of irreducible decompositions. We will elaborate more on these methods in Chapter 3. Triangular sets are implemented in Maple and Axiom, and also in Singular and Magma for the zero-dimensional case.

Numerical algebraic geometry. This is a relatively recent numerical approach for solving polynomial equations [124]. The basic idea is to form a homotopy between the original system and an auxiliary system, for which the solutions are known, and then track the solutions along this path. These methods are gaining popularity, thanks to features such as being trivially parallelizable, and offering better numerical stability than symbolic methods. There are different software implementations, including Bertini [13] and PHCpack [137].

We point out that solving polynomial systems is NP-hard, and all methods from above scale exponentially in n , where n is the number of variables. In the first part of this thesis we propose a novel technique for solving polynomial systems that attempts to reduce this complexity by taking into account the sparsity structure of the system. In particular, we show that our method takes time $O(n)$ for several (well structured) families of polynomial systems.

2.1.3 Polynomial optimization

Let us now fix the field $\mathbb{K} = \mathbb{R}$ of real numbers. Given polynomials $p, h_1, \dots, h_m \in \mathbb{R}[x]$, consider the *polynomial optimization problem*

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & p(x) \\ & h_i(x) = 0, \quad \text{for } i = 1, \dots, m. \end{aligned}$$

Equivalently, we are minimizing p over the *real trace* of an algebraic variety:

$$\min_{x \in \mathbf{V} \cap \mathbb{R}^n} p(x), \quad \text{where} \quad \mathbf{V} := \{x \in \mathbb{C}^n : h_i(x) = 0 \text{ for all } h_i\}. \quad (2.3)$$

Problem (2.3) captures many problems of interest, such as Max-Cut, maximum power flow, the triangulation problem (computer vision), and tensor low rank approximation. Although this optimization problem is computationally hard, semidefinite programming (SDP) relaxations based on the sum-of-square (SOS) method provide a tractable alternative, that has been successful in many different applications. In particular, it provides the best known polynomial-time approximation algorithm for the Max-Cut problem [65]. We now proceed to describe it.

In its simplest form, the SOS method attempts to certify that a polynomial $F \in \mathbb{R}[x]$ is globally nonnegative, i.e., whether $F(x) \geq 0$ for all x . We say that $F(x)$ is a sum-of-squares (SOS) if it can be written in the form $F(x) = \sum_i f_i(x)^2$ for some $f_i \in \mathbb{R}[x]$. The SOS method relies on the trivial fact that SOS polynomials are nonnegative. Thanks to the following proposition, determining whether a polynomial is SOS can be done efficiently with SDP.

Proposition 2.2 ([105]). *Let $F \in \mathbb{R}[x]$ be a polynomial of degree $2d$. Let $u(x) \in \mathbb{R}[x]^N$ be the vector containing all $N = \binom{n+d}{d}$ monomials of degree at most d . Then $F(x)$ is SOS if and only there is a matrix $Q \in \mathcal{S}^N$ such that*

$$F(x) = u(x)^T Q u(x), \quad Q \succeq 0.$$

Coming back to optimization, consider the unconstrained minimization of a polynomial $p(x)$. Note that this is equivalent to finding the largest γ such that $p(x) - \gamma$ is nonnegative.

Relaxing the nonnegativity constraint with an SOS condition we get

$$\begin{aligned} \max_{\gamma \in \mathbb{R}^n} \quad & \gamma \\ & p(x) - \gamma \text{ is SOS} \end{aligned}$$

By Proposition 2.2, the above problem is an SDP, and its solution γ^* is a valid lower bound on $p(x)$. SOS lower bounds tend to be very good in practice, being the true minimum in certain applications [22, 85]. Moreover, if the minimizer x^* is unique and the dual solution of the SDP is rank-one, then x^* might be recovered from the SDP.

The constrained case of (2.3) is very similar. First observe that (2.3) is equivalent to the following problem:

$$\begin{aligned} \max_{\gamma \in \mathbb{R}^n} \quad & \gamma \\ & p(x) - \gamma \geq 0 \text{ for all } x \in \mathbf{V} \cap \mathbb{R}^n \end{aligned}$$

Although the nonnegativity constraint from above is complicated, it can be relaxed with a suitable SOS condition. Assume that there exist polynomials $F, g_1, \dots, g_m \in \mathbb{R}[x]$ such that

$$p(x) - \gamma = F(x) + \sum_i g_i(x)h_i(x), \quad F(x) \text{ is SOS.} \quad (2.4)$$

Such a tuple (F, g_1, \dots, g_m) certifies the nonnegativity of $p(x) - \gamma$ on the variety. The SOS relaxation of (2.3) is

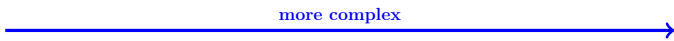
$$\begin{aligned} \max_{\gamma \in \mathbb{R}^n, F, g_1, \dots, g_m \in \mathbb{R}[x]} \quad & \gamma \\ & p(x) - \gamma = F(x) + \sum_i g_i(x)h_i(x); \\ & F(x) \text{ is SOS} \end{aligned}$$

By restricting the degrees of F, g_1h_1, \dots, g_mh_m to be at most $2d$, the above problem becomes an SDP. As before, the solution γ^* of the SDP gives a lower bound on the actual value of (2.3).

The second part of this thesis focuses on the SOS method. We will show how to derive more efficient relaxations by exploiting the geometry of the variety. We will also analyze the quality of these relaxations for the case of nearest point problems (i.e., $p(x) = \|x - \theta\|^2$).

2.2 Treewidth and chordality

The notion of *treewidth* is fundamental in many areas of computer science and applied mathematics [26,50]. Intuitively, the treewidth is a measure of complexity of a graph, which quantifies how close it is from being a tree. A graph has treewidth one if and only if it is a forest, i.e., a disjoint union of trees. The smaller the treewidth, the closer the graph is to a tree, and the easier it is to solve certain problems on it. The following diagram summarizes the treewidth of some simple graphs.

					
graph	tree	cycle graph C_n	grid graph $P_{\sqrt{n}} \times P_{\sqrt{n}}$	compl. bipartite $K_{n/2, n/2}$	compl. graph K_n
treewidth	1	2	\sqrt{n}	$n/2$	$n - 1$

Remarkably, several hard combinatorial problems (e.g., independent set, clique number, chromatic number) can be solved efficiently on graphs of *bounded treewidth* [26]. More precisely, many such problems can be solved via dynamic programming in time $O(n 2^{O(\omega)})$, where ω is the treewidth. We will see in the first part of this thesis that similar results can be derived for certain polynomial systems, as well as for permanents of matrices. Unfortunately, computing the treewidth of a graph is NP-hard [3]. Nevertheless, there are good heuristics and approximation algorithms. See [27] for a comparison of some of these heuristics.

There are a few equivalent ways to define the treewidth ω of a graph \mathcal{G} . Two of them are:

- $\omega + 1$ is the smallest clique number of a *chordal completion* of \mathcal{G} .
- ω is the smallest width of a *tree decomposition* of \mathcal{G} .

We now proceed to explain the concepts of chordal completions and tree decompositions.

2.2.1 Chordal graphs and chordal completions

Chordal graphs have many equivalent characterizations. A good presentation is found in [21]. For our purposes, we use the following definition.

Definition 2.3. Let G be a graph with vertices x_0, \dots, x_{n-1} . An ordering of its vertices

$x_0 > x_1 > \cdots > x_{n-1}$ is a *perfect elimination ordering* if for each x_l the set

$$X_l := \{x_l\} \cup \{x_m : x_m \text{ is adjacent to } x_l, x_m < x_l\} \quad (2.5)$$

is such that the restriction $G|_{X_l}$ is a clique. A graph G is *chordal* if it has a perfect elimination ordering.

Remark 2.3. Observe that lower indices correspond to larger vertices.

Chordal graphs have many interesting properties. For instance, they have at most n maximal cliques, given that any clique is contained in some X_l . Note that trees are chordal graphs, since by successively pruning a leaf from the tree we get a perfect elimination ordering. We can always find a perfect elimination ordering of a chordal graph in linear time [111].

Definition 2.4. Let \mathcal{G} be an arbitrary graph. We say that G is a *chordal completion* of \mathcal{G} if it is chordal and \mathcal{G} is a subgraph of G . The *clique number* of G is the size of its largest clique. The *treewidth* of \mathcal{G} is the minimum clique number of G (minus one) among all possible chordal completions.

Observe that given any ordering $x_0 > \cdots > x_{n-1}$ of the vertices of \mathcal{G} , there is a natural chordal completion G , i.e. we add edges to \mathcal{G} in such a way that each $G|_{X_l}$ is a clique. In general, we want to find a chordal completion with a small clique number. However, there are $n!$ possible orderings of the vertices and thus finding the best chordal completion is not simple. Indeed, computing the treewidth is NP-hard [3].

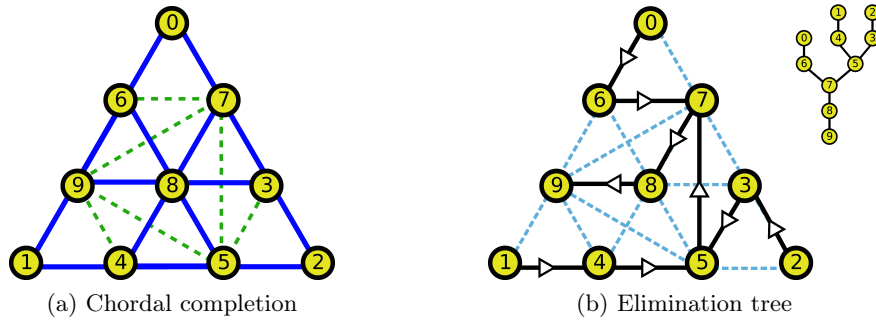


Figure 2-1: Left: 10-vertex graph (blue/solid) and a chordal completion (green/dashed). Right: Elimination tree of the chordal completion.

Example 2.2. Let \mathcal{G} be the blue/solid graph in Figure 2-1a. This graph is not chordal but if we add the six green/dashed edges shown in the figure we obtain a chordal completion G . In fact, the ordering $x_0 > \dots > x_9$ is a perfect elimination ordering of the chordal completion. The clique number of G is four and the treewidth of \mathcal{G} is three.

Given a chordal graph G with some perfect elimination ordering, there is an associated tree that will be very helpful in our discussion.

Definition 2.5. Let G be an ordered graph with vertices $x_0 > \dots > x_{n-1}$. The *elimination tree* of G is the following directed spanning tree: for each l there is an arc from x_l towards the largest x_p that is adjacent to x_l and $x_p < x_l$. We will say that x_p is *the parent* of x_l and x_l is *a child* of x_p . Note that the elimination tree is rooted at x_{n-1} .

Figure 2-1b shows an example of the elimination tree. We now present a simple property of the elimination tree of a chordal graph.

Lemma 2.3. *Let G be a chordal graph, let x_l be some vertex and let x_p be its parent in the elimination tree T . Then $X_l \setminus \{x_l\} \subseteq X_p$, where X_i is as in eq. (2.5).*

Proof. Let $Y := X_l \setminus \{x_l\}$. Note that Y is a clique, whose largest variable is x_p . Since X_p is the unique largest clique satisfying such property, we must have $Y \subseteq X_p$. \square

2.2.2 Tree decompositions

Definition 2.6. Let \mathcal{G} be a graph with vertex set X . A *tree decomposition* of \mathcal{G} is a pair (T, χ) , where T is a rooted tree and $\chi : T \rightarrow \{0, 1\}^X$ assigns some $\chi(t) \subseteq X$ to each node t of T , that satisfies the following conditions.

- (i) The union of $\{\chi(t)\}_{t \in T}$ is the whole vertex set X .
- (ii) For every edge (x_i, x_j) of \mathcal{G} , there exists some node t of T with $x_i, x_j \in \chi(t)$.
- (iii) For every $x_i \in X$ the set $\{t : x_i \in \chi(t)\}$ forms a subtree of T .

The sets $\chi(t)$ are usually referred to as *bags*. The *width* of the decomposition is the size of the largest bag (minus one). The *treewidth* of \mathcal{G} can also be defined as the minimum width among all possible tree decompositions.

Example 2.3. Let \mathcal{G} be the blue/solid graph from Figure 2-1a. Figure 2-2 shows a tree decomposition of \mathcal{G} (the conditions from above are easy to check). Note that the width of the decomposition is $4 - 1 = 3$. We will see that this tree decomposition arises from the (green/dashed) chordal completion shown in Figure 2-1a.

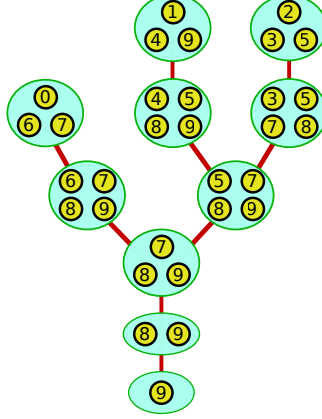


Figure 2-2: Tree decomposition of the blue/solid graph from Figure 2-1a.

A simple property of tree decompositions we will use later is that condition (ii) also holds for any clique of the graph, as stated next.

Lemma 2.4. *Let (T, χ) be a tree decomposition of \mathcal{G} . Then for any clique Y of \mathcal{G} there is some node t of T with $Y \subseteq \chi(t)$.*

Proof. For each $y \in Y$, let T_y denote the subtree of all bags containing y . Let $t_y \in T_y$ be the closest node to the root and let $d(t_y)$ denote the distance from t_y to the root. Observe that if $d(t_y) \leq d(t_{y'})$ then $t_{y'} \in T_y$ (otherwise, $T_y \cap T_{y'} = \emptyset$ and the edge (y, y') would not belong to any bag). Let $t \in \{t_y\}_{y \in Y}$ be the farthest away from the root, i.e., $d(t_y) \leq d(t)$ for all $y \in Y$. It follows that $Y \subseteq \chi(t)$. \square

We proceed to explain the relationship between chordal completions and tree decompositions. Given a chordal graph G , we can always construct a tree decomposition where the bags $\chi(t)$ correspond to cliques X_t from (2.5).

Lemma 2.5 (Clique tree of a chordal graph). *Let G be a chordal graph with vertex set $X = \{x_0, \dots, x_{n-1}\}$, and let T be its elimination tree. Let $\chi : T \rightarrow \{0, 1\}^X$ be such that $\chi(x_t) := X_t$, where X_t is the clique from (2.5). Then (T, χ) is a tree decomposition of G .*

Proof. Condition (i) follows from the fact that vertex x_i belongs to clique X_i . Condition (ii) follows by noticing that for each edge (x_i, x_j) we have $x_i, x_j \in X_i$ (assuming $x_i > x_j$). It remains to check condition (iii).

Consider a path $x_{i_1}—x_{i_2}—\dots—x_{i_k}$ of the elimination tree. We need to show that if a vertex x_l belongs to both cliques X_{i_1} and X_{i_k} , then x_l belongs to all the cliques in the path. It suffices to show that x_l lies in one of the cliques in the interior of the path, since we can then repeat the same argument on a smaller subpath. It is easy to see that either $x_{i_1} > x_{i_2}$ or $x_{i_k} > x_{i_{k-1}}$. Assume WLOG that $x_{i_1} > x_{i_2}$, which means that x_{i_2} is the parent of x_{i_1} . If we show that $x_l \neq x_{i_1}$ we would be done, since by Lemma 2.3 we would have $x_l \in X_{i_1} \setminus \{x_{i_1}\} \subseteq X_{i_2}$. Assume by contradiction that $x_l = x_{i_1}$. Since $x_{i_1} \in X_{i_k}$ then $x_{i_1} < x_{i_k}$, and moreover, x_{i_1} is an ancestor of x_{i_k} . This means that x_{i_1} is the parent of x_{i_2} , which is a contradiction. \square

Remark 2.4. (Chordal completions \iff tree decompositions) Let \mathcal{G} be an arbitrary graph, and let G be a chordal completion. Let (T, χ) be the tree decomposition of G from above, and observe that (T, χ) is also a tree decomposition of \mathcal{G} . Therefore, any chordal completion of \mathcal{G} gives rise to a tree decomposition. Conversely, any tree decomposition (T, χ) of \mathcal{G} can be used to construct a chordal completion, simply by connecting the vertices in each of the bags $\chi(t)$.

Part I

Polynomial systems and graphical structure

Chapter 3

Chordal networks of polynomial ideals

The sparsity structure of a polynomial system is often described by a graph that captures the interactions among the variables. This chapter proposes a novel representation of polynomial systems, chordal networks, which takes advantage of this graphical structure. The content of this chapter is based on [41].

3.1 Introduction

Systems of polynomial equations can be used to model a large variety of applications, and in most cases the resulting systems have a particular sparsity structure. We can describe this sparsity structure using a graph. A natural question that arises is whether this graphical structure can be effectively used to solve the system. Unfortunately, existing techniques such as Gröbner bases destroy this graphical structure. Consequently, polynomial systems with simple structure may have overly complicated Gröbner bases (see Example 3.1). In this chapter we introduce a new method for solving polynomial systems and a new data structure for representing the associated varieties, that aim to exploit the underlying graphical structure. We call this data structure *chordal networks*.

Chordal networks describe a decomposition of the polynomial system into simpler (triangular) polynomial sets. This decomposition gives quite a rich description of the underlying variety. In particular, chordal networks can be efficiently used to compute dimension, cardinality, equidimensional components and also to test radical ideal membership. Remarkably, several families of polynomial systems (with exponentially large Gröbner bases) admit a com-

pact chordal network representation, of size proportional to the number of variables. We will shortly present some motivational examples after setting up the main terminology.

Throughout this chapter we work in the polynomial ring $\mathbb{K}[X] = \mathbb{K}[x_0, x_1, \dots, x_{n-1}]$ over some field \mathbb{K} . We fix once and for all the ordering of the variables $x_0 > x_1 > \dots > x_{n-1}$ ¹. We consider a system of polynomials $F = \{f_1, f_2, \dots, f_m\}$. There is a natural graph $\mathcal{G}(F)$, with vertex set $X = \{x_0, \dots, x_{n-1}\}$, that abstracts the *sparsity structure* of F . The graph is given by cliques: for each f_i we add a clique in all its variables. Equivalently, there is an edge between x_i and x_j if and only if there is some polynomial in F that contains both variables. We will consider throughout this chapter a chordal completion G of the graph $\mathcal{G}(F)$, and we will assume that $x_0 > \dots > x_{n-1}$ is a perfect elimination ordering of G (see Definition 2.3).

Definition 3.1. Let $F \subseteq \mathbb{K}[X]$ be a polynomial system and let G be a chordal graph. We say that F is *supported* on G if G is a chordal completion of $\mathcal{G}(F)$, i.e., $G \supseteq \mathcal{G}(F)$.

Some motivating examples

The notions of chordality and treewidth are ubiquitous in applied mathematics and computer science. In particular, several hard combinatorial problems can be solved efficiently on graphs of small treewidth by using some type of recursion (or dynamic program) [26]. We will see that this recursive nature is also present in several polynomial systems of small treewidth. We now illustrate this with three simple examples.

Example 3.1 (Coloring a cycle graph). Graph coloring is a classical NP-complete problem that can be solved efficiently on graphs of small treewidth. We consider the cycle graph C_n with vertices $0, 1, \dots, n-1$, whose treewidth is two. Coloring C_n is particularly simple by proceeding in a recursive manner: color vertex $n-1$ arbitrarily and then subsequently color vertex i avoiding the color of $i+1$ and possibly $n-1$.

We saw in Example 2.1 how to pose the q -coloring problem as a system of polynomial equations. Let $F_{n,q}$ denote the polynomial system (2.2) for the case of the cycle graph C_n . Given that coloring the cycle graph is so easy, it should be possible to solve these equations efficiently. However, if we compute a Gröbner basis the result is not so simple. In particular, for the case of $F_{9,3}$ one of these polynomials has 81 terms (with both lex and grevlex order).

¹Observe that smaller indices correspond to larger variables.

This is a consequence of the fact that Gröbner bases destroy the graphical structure of the equations.

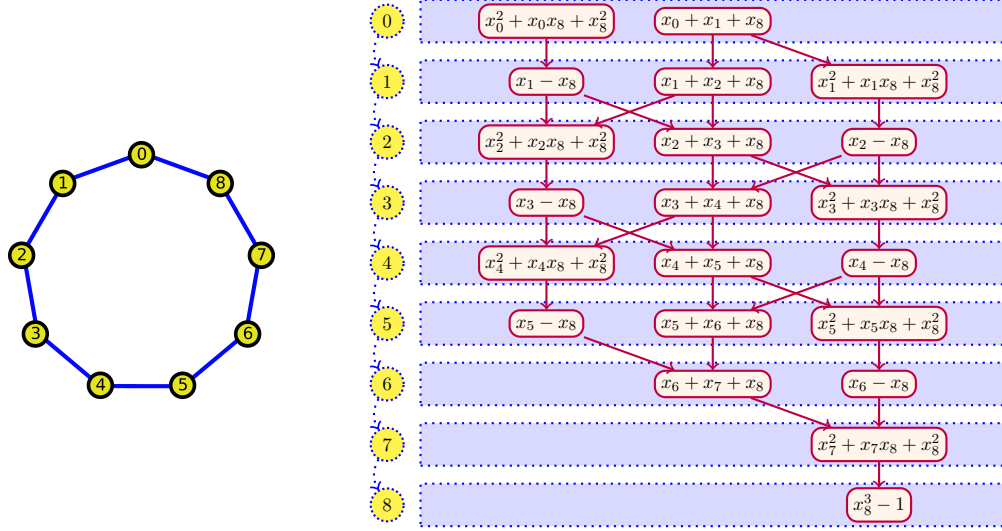


Figure 3-1: Chordal network for the 3-chromatic ideal of a cycle

Nonetheless, one may hope to give a simple representation of the above polynomials that takes into account their recursive nature. Indeed, a triangular decomposition of these equations is presented in Figure 3-1 for the case of $F_{9,3}$, and the pattern is very similar for arbitrary values of n, q . The decomposition represented is:

$$\mathbf{V}(F_{9,3}) = \bigcup_T \mathbf{V}(T)$$

where the union is over all maximal directed paths in the diagram of Figure 3-1. One path is

$$T = \{x_0 + x_1 + x_8, x_1^2 + x_1x_8 + x_8^2, x_2 - x_8, x_3^2 + x_3x_8 + x_8^2, x_4 - x_8, \\ x_5^2 + x_5x_8 + x_8^2, x_6 - x_8, x_7^2 + x_7x_8 + x_8^2, x_8^3 - 1\}.$$

Recall that a set of polynomials is triangular if the largest variables of these polynomials are all distinct, and observe that all maximal paths T are triangular. Note that the total number of triangular sets is 21, and in general we get the $(n-1)$ -th Fibonacci number. Even though the size of the triangular decomposition grows rapidly, it admits a very compact representation (linear in n) and the reason is precisely the recursive nature of the equations. Indeed, the

diagram of Figure 3-1 is constructed in a very similar way as we construct colorings: choose x_8 arbitrarily, then for each x_i choose it based on the values of x_{i+1} and x_8 .

Example 3.2 (Vertex covers of a tree). We now consider the problem of finding minimum vertex coverings of a graph. Recall that a subset S of vertices is a cover if any edge is incident to at least one element in S . Since the complement of a vertex cover is an independent set, computing a minimum vertex cover is NP-complete. Nevertheless, when the graph is a tree the minimal vertex covers have a very special structure. Indeed, we can construct such a cover recursively, starting from the root, as follows. For the root node, we can decide whether to include it in the cover or not. If we include it, we can delete the root and then recurse on each of its children. Otherwise, we need to include in the cover all of its children, so we can delete them all, and then recurse.

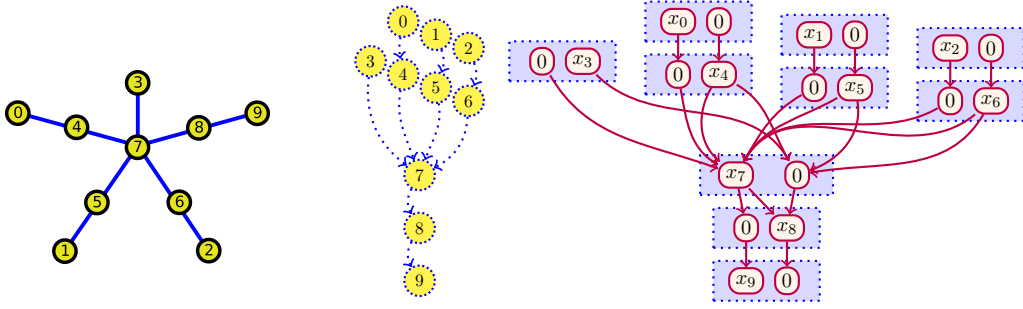


Figure 3-2: Chordal network for the edge ideal of a tree.

The minimal vertex covers of a graph $\mathcal{G} = (V, E)$ are in correspondence with the irreducible components of its *edge ideal* $I(\mathcal{G}) := \langle x_i x_j : ij \in E \rangle$ (see e.g., [139, Prop 7.2.3]). Therefore, the irreducible components of the edge ideal of a tree always have a very simple structure (although there might be exponentially many). For instance, the diagram in Figure 3-2 represents the components for the case of a simple 10-vertex tree. Here the components are given by the possible choices of one node from each of the (purple) boxes so that these nodes are connected (e.g., $T = \{0, 0, 0, x_3, x_4, x_5, x_6, 0, x_8, 0\}$). Note that there are $2^4 + 1 = 17$ components.

Example 3.3 (Adjacent minors). Let X be a $2 \times n$ matrix of indeterminates, and consider the polynomial set F_n given by its adjacent minors, i.e.,

$$X := \begin{pmatrix} x_0 & x_2 & \cdots & x_{2n-2} \\ x_1 & x_3 & \cdots & x_{2n-1} \end{pmatrix}, \quad F_n := \{x_{2i}x_{2i+3} - x_{2i+1}x_{2i+2} : 0 \leq i < n-1\}.$$

The corresponding ideal has been studied in e.g., [53,70]. Figure 3-3 shows the graph associated to this system. We are interested in describing the irreducible components of $\mathbf{V}(F_n)$.

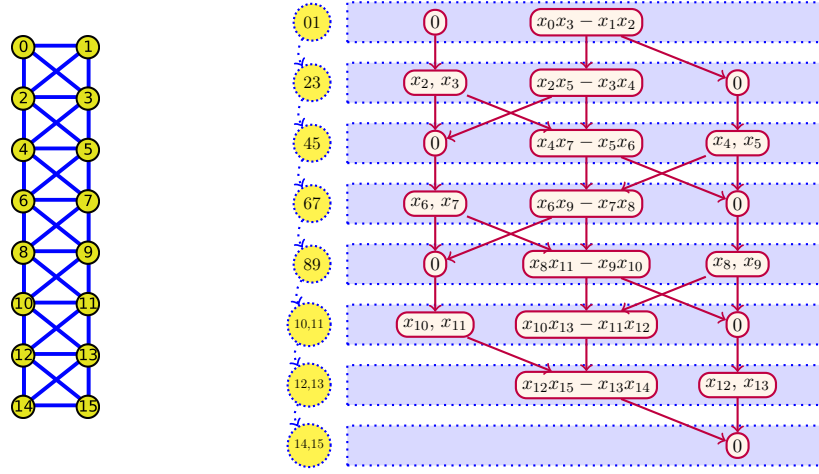


Figure 3-3: Chordal network for the ideal of adjacent minors

As in Example 3.1, there is a simple recursive procedure to produce points on $\mathbf{V}(F_n)$: we choose the values of the last column of the matrix arbitrarily, and then for column i we either choose it arbitrarily, in case that column $i + 1$ is zero, or we scale column $i + 1$ if it is nonzero. This procedure is actually describing the irreducible components of the variety. In this way, the irreducible components admit a compact description, which is shown in Figure 3-3. Again, the components are given by the maximal directed paths (e.g., $T = \{0, x_2, x_3, 0, x_6, x_7, x_8x_{11} - x_9x_{10}, 0, x_{12}, x_{13}, 0\}$) and its cardinality is the n -th Fibonacci number.

Contributions

The examples from above show how certain polynomial systems with tree-like structure admit a compact chordal network representation. The aim of this chapter is to develop a general framework to systematically understand and compute chordal networks. We also study how to effectively use chordal networks to solve different problems from computational algebraic geometry. A major difficulty is that exponentially many triangular sets may appear (e.g., the Fibonacci number in Example 3.1).

This chapter presents the following contributions:

- We introduce the notion of chordal networks, a novel representation of polynomial

ideals aimed toward exploiting structured sparsity.

- We develop the *chordal triangularization* method (Algorithm 1) to compute such chordal network representation. Its correctness is established in Theorems 3.1 and 3.19.
- We show that several families of polynomial systems admit a “small” chordal network representation, of size $O(n)$. This is true for certain zero-dimensional ideals (Remark 3.6), all monomial ideals (Theorem 3.18) and certain binomial/determinantal ideals (Section 3.7.3), although in general this cannot be guaranteed (Remark 3.7).
- We show how to effectively use chordal networks to compute several properties of the underlying variety. In particular, the cardinality (Section 3.4.2), dimension and top-dimensional component (Section 3.5.2) can be computed in linear time. In some interesting cases we can also describe the irreducible components.
- We present a Monte Carlo algorithm to test radical ideal membership (Algorithm 3). We show in Theorem 3.12 that the complexity is linear when the given polynomial preserves some of the graphical structure of the system.

We point out that all the methods presented in this chapter are available in the Macaulay2 package `Chordal` [40].

Structure of this chapter

The organization of this chapter is as follows. In Section 3.2 we formalize the notion of chordal network. We then proceed to explain our methods, initially only for a restricted class of zero-dimensional problems (Sections 3.3 and 3.4), then for the case of monomial ideals (Section 3.5), and finally considering the fully general case (Section 3.6). We conclude the chapter in Section 3.7 with numerical examples of our methods.

The reason for presenting our results in this stepwise manner, is that the general case requires highly technical concepts from the theory of triangular sets. Indeed, we encourage the reader unfamiliar with triangular sets to omit Section 3.6 in the first read. On the other hand, by first specializing our methods to the zero-dimensional and monomial cases we can introduce them all in a transparent manner. Importantly, the basic structure of the chordal triangularization algorithm, presented in Section 3.3, remains the same for the general case.

Similarly, our algorithms that use chordal networks to compute properties of the variety (e.g., cardinality, dimension), introduced in Sections 3.4 and 3.5, also extend in a natural way.

Related work

The development of chordal networks can be seen as a continuation of our earlier work [36,39], and we refer the reader to these documents for a detailed survey of the relevant literature on graphical structure in computational algebraic geometry. For this reason, below we only discuss related work in the context of triangular sets, and point out the main differences between this chapter and [36,39].

This chapter improves upon [36,39] in two main areas. Firstly, chordal networks provide a much richer description of the variety than the elimination ideals obtained by chordal elimination. For instance, the elimination ideals of the equations from Example 3.3 are trivial, but its chordal network representation reveals its irreducible components. In addition, neither the dimension, cardinality nor radical ideal membership can be directly computed from the elimination ideals (we need a Gröbner basis). Secondly, we show how to compute chordal network representations for arbitrary polynomial systems (in characteristic zero). In contrast, chordal elimination only computes the elimination ideals under certain assumptions.

There is a broad literature studying triangular decompositions of ideals [5, 78, 88, 95, 143]. However, past work has not considered the case of sparse polynomial systems. Among the many existing triangular decomposition algorithms, Wang’s elimination methods are particularly relevant to us [143, 144]. Although seemingly unnoticed by Wang, most of his algorithms preserve the chordal structure of the system. As a consequence, we have experimentally seen that his methods are more efficient than those based on regular chains [89, 95] for the examples considered in this chapter.

As opposed to previous work, we emphasize chordal networks as our central object of study, rather than the explicit triangular decomposition obtained. This is a key distinction since for several families of ideals the size of the chordal network is linear even though the corresponding triangular decomposition has exponential size (see the examples from above). In addition, our methods deliberately treat triangular decompositions as a black box algorithm, allowing us to use either Lazard’s LexTriangular algorithm [88] for the zero-dimensional case or Wang’s

RegSer algorithm [142] for the positive-dimensional case.

3.2 Chordal networks

We proceed to formalize the concept of chordal networks. We will use the concepts of chordal graph and elimination tree introduced in Section 2.2.1.

Definition 3.2. Let G be a chordal graph with vertex set $X = \{x_0, \dots, x_{n-1}\}$, and let X_l be as in eq. (2.5). A G -chordal network is a directed graph \mathcal{N} , whose nodes are polynomial sets in $\mathbb{K}[X]$, such that:

- (*nodes supported on cliques*) each node F_l of \mathcal{N} is given a rank $l = \text{rk}(F_l)$, with $0 \leq l < n$, such that $F_l \subseteq \mathbb{K}[X_l]$.
- (*arcs follow elimination tree*) if (F_l, F_p) is an arc of \mathcal{N} then (l, p) is an arc of the elimination tree of G , where $l = \text{rk}(F_l), p = \text{rk}(F_p)$.

A chordal network is *triangular* if each node consists of a single polynomial f , and either $f = 0$ or its largest variable is $x_{\text{rk}(f)}$.

There is one parameter of a chordal network that will determine the complexity of some of our methods. The *width* of a chordal network, denoted as W , is the largest number of nodes of any given rank. Note that the number of nodes in the network is at most nW , and the number of arcs is at most $(n-1)W^2$.

We can represent chordal networks using the diagrams we have shown throughout the chapter. Since the structure of a chordal network resembles the elimination tree (second item in the definition), we usually show the elimination tree to the left of the network.

Example 3.4. Let \mathcal{G} be the blue/solid graph from Figure 2-1a, and let G be the green/dashed chordal completion. Figure 3-4 shows a G -chordal network of width 5, that represents the 4-colorings of graph \mathcal{G} (Equation (2.2)). The elimination tree of G is shown to the left of the diagram. Note that this network is triangular, and thus all its nodes consist of a single polynomial. For instance, two of its nodes are $f_5 = x_5 + x_7 + x_8 + x_9$ and $f_6 = x_6 - x_7$. Nodes are grouped in blue rectangular boxes according to their rank. In particular, f_5 has rank 5 and f_6 rank 6, and indeed $f_5 \in \mathbb{K}[X_5] = \mathbb{K}[x_5, x_7, x_8, x_9]$ and $f_6 \in \mathbb{K}[X_6] = \mathbb{K}[x_6, x_7, x_8, x_9]$.

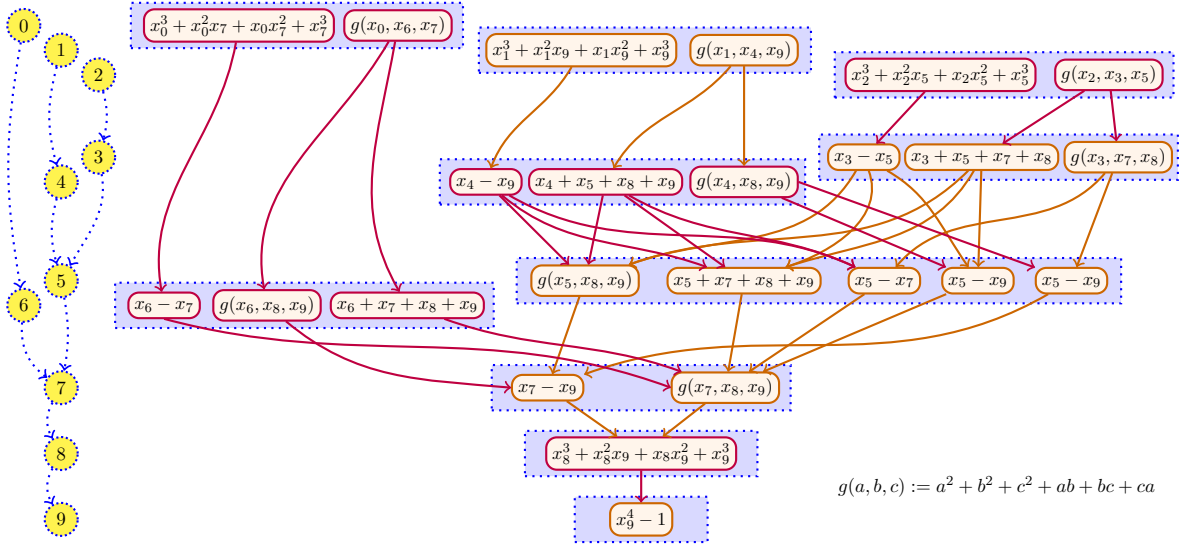


Figure 3-4: G -chordal network, where G is the chordal graph from Figure 2-1a. The elimination tree of G is shown on the left.

Example 3.5. Let \mathcal{G} be the 9-cycle with vertices x_0, \dots, x_8 . Let G be the chordal completion obtained by connecting vertex x_8 to all the others. Figure 3-1 shows a triangular G -chordal network. The elimination tree, shown to the left of the network, is the path $x_0 \rightarrow \dots \rightarrow x_8$.

Remark 3.1. Sometimes we collapse certain ranks to make the diagram visually simpler. In particular, in Figure 3-3 we collapse the ranks $2i, 2i + 1$ into a single group.

As suggested by the examples in the introduction, a triangular chordal network gives a decomposition of the polynomial ideal into triangular sets. Each such triangular set corresponds to a *chain* of the network, as defined next.

Definition 3.3. Let \mathcal{N} be a G -chordal network. A *chain* of \mathcal{N} is a tuple of nodes $C = (F_0, F_1, \dots, F_{n-1})$ such that:

- $\text{rk}(F_l) = l$ for each l .
- if x_p is the parent of x_l , then (F_l, F_p) is an arc of \mathcal{N} .

Example 3.6. The chordal network from Figure 3-4 has 21 chains, one of which is:

$$C = (x_0^2 + x_0x_6 + x_0x_7 + x_6^2 + x_6x_7 + x_7^2, x_1^3 + x_1^2x_9 + x_1x_9^2 + x_9^3, x_2^3 + x_2^2x_5 + x_2x_5^2 + x_5^3, \\ x_3 - x_5, x_4 - x_9, x_5^2 + x_5x_8 + x_5x_9 + x_8^2 + x_8x_9 + x_9^2, \\ x_6^2 + x_6x_8 + x_6x_9 + x_8^2 + x_8x_9 + x_9^2, x_7 - x_9, x_8^3 + x_8^2x_9 + x_8x_9^2 + x_9^3, x_9^4 - 1).$$

Binary Decision Diagrams

Although motivated from a different perspective and with quite distinct goals, throughout the development of this work we realized the intriguing similarities between chordal networks and a class of data structures used in computer science known as ordered binary decision diagrams (OBDD) [2, 29, 82, 148].

A binary decision diagram (BDD) is a data structure that can be used to represent Boolean (binary) functions in terms of a directed acyclic graph. They can be interpreted as a binary analogue of a straight-line program, where the nodes are associated with variables and the outgoing edges of a node correspond to the possible values of that variable. A particularly important subclass are the *ordered* BDDs (or OBDDs), where the branching occurs according to a specific fixed variable ordering. Under a mild condition (reducibility) this representation can be made unique, and thus every Boolean function has a canonical OBDD representation. OBDDs can be effectively used for further manipulation (e.g., decide satisfiability, count satisfying assignments, compute logical operations). Interestingly, several important functions have a compact OBDD representation. A further variation, *zero-suppressed* BDDs (ZBDDs), can be used to efficiently represent subsets of the hypercube $\{0, 1\}^n$ and to manipulate them (e.g., intersections, sampling, linear optimization).

Chordal networks can be thought of as a wide generalization of OBDDs/ZBDDs to arbitrary algebraic varieties over general fields (instead of finite sets in $(\mathbb{F}_2)^n$). Like chordal networks, an OBDD corresponds to a certain directed graph, but where the nodes are variables (x_0, x_1, \dots) instead of polynomial sets. We will see in Section 3.5 that for the specific case of monomial ideals, the associated chordal networks also have this form. Since one of our main goals is to preserve graphical structure for efficient computation, in this document we define chordal networks only for systems that are structured according to some chordal graph. In addition, for computational purposes we do not insist on uniqueness of the representation

(although it might be possible to make them canonical after further processing).

The practical impact of data structures like BDDs and OBDDs over the past three decades has been very significant, as they have enabled breakthrough results in many areas of computer science including model checking, formal verification and logic synthesis. We hope that chordal networks will make possible similar advances in computational algebraic geometry. The connections between BDDs and chordal networks run much deeper, and we plan to further explore them in the future.

3.3 The chordally zero-dimensional case

In this section we present our main methods to compute triangular chordal networks, although focused on a restricted type of zero-dimensional problems. This restriction is for simplicity only; we will see that our methods naturally extend to arbitrary ideals. Concretely, we consider the following family of polynomial sets.

Definition 3.4 (Chordally zero-dimensional). Let $F \subseteq \mathbb{K}[X]$ be supported on a chordal graph G . We say that F is *chordally* zero-dimensional, if for each maximal clique X_l of graph G the ideal $\langle F \cap \mathbb{K}[X_l] \rangle$ is zero-dimensional.

Note that the q -coloring equations in (2.2) are chordally zero-dimensional. As in Example 3.1, chordally zero-dimensional problems always have simple chordal network representations.

Remark 3.2 (The geometric picture). There is a nice geometric interpretation behind the chordally zero-dimensional condition. Denoting V_l the variety of $F \cap \mathbb{K}[X_l]$, the condition is that each V_l is finite. Note now that $\pi_{X_l}(\mathbf{V}(F)) \subseteq V_l$, where π_{X_l} denotes the projection onto the coordinates of X_l . Thus, independent of the size of $\mathbf{V}(F)$, the chordally zero-dimensional condition allows us to bound the size of its projections onto each X_l . More generally, although not elaborated in this document, our methods are expected to perform well on any F (possibly positive-dimensional) for which the projections $\pi_{X_l}(\mathbf{V}(F))$ are well-behaved.

3.3.1 Triangular sets

We now recall the basic concepts of triangular sets for the case of zero-dimensional ideals, following [88]. We delay the exposition of the positive-dimensional case to Section 3.6.

Definition 3.5. Let $f \in \mathbb{K}[X] \setminus \mathbb{K}$ be a non-constant polynomial. The *main variable* of f , denoted $\text{mvar}(f)$, is the greatest variable appearing in f . The *initial* of f , denoted $\text{init}(f)$, is the leading coefficient of f when viewed as a univariate polynomial in $\text{mvar}(f)$. A *zero-dimensional triangular set* is a collection of non-constant polynomials $T = \{t_0, \dots, t_{n-1}\}$ such that $\text{mvar}(t_i) = x_i$ and $\text{init}(t_i) = 1$ for each i .

Most of the analysis done in this chapter will work over an arbitrary field \mathbb{K} . For some results we require the field to contain sufficiently many elements, so we might need to consider a field extension. We denote by $\overline{\mathbb{K}}$ the algebraic closure of \mathbb{K} . For a polynomial set F , we let $\mathbf{V}(F) \subseteq \overline{\mathbb{K}}^n$ be its variety. Note that for a zero-dimensional triangular set T , we always have

$$|\mathbf{V}(T)| \leq \deg(T) := \prod_{t \in T} \text{mdeg}(t), \quad (3.1)$$

where $\text{mdeg}(t) := \deg(t, \text{mvar}(t))$ denotes the degree on the main variable. Furthermore, the above is an equality if we count multiplicities.

For a triangular set T , let $\langle T \rangle$ denote the generated ideal. It is easy to see that a zero-dimensional triangular set is a lexicographic Gröbner basis of $\langle T \rangle$. In particular, we can test ideal membership by taking normal form. We also denote as $\text{elim}_p(T) := T \cap \mathbb{K}[x_p, x_{p+1}, \dots]$ the subset of T restricted to variables less or equal than x_p . Note that $\text{elim}_p(T)$ generates the elimination ideal of $\langle T \rangle$ because of the elimination property of lexicographic Gröbner bases.

Notation. $S = S_1 \sqcup S_2$ denotes a disjoint union, i.e., $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$.

Definition 3.6. Let $I \subseteq \mathbb{K}[X]$ be a zero-dimensional ideal. A *triangular decomposition* of I is a collection \mathcal{T} of triangular sets, such that $\mathbf{V}(I) = \bigsqcup_{T \in \mathcal{T}} \mathbf{V}(T)$. We say that \mathcal{T} is *squarefree* if each $T \in \mathcal{T}$ generates a radical ideal. We say that \mathcal{T} is *irreducible* if each $T \in \mathcal{T}$ generates a prime ideal (or equivalently, a maximal ideal).

Lazard proposed algorithms to compute a triangular decomposition from a Gröbner basis [88]. He also showed how to post-process it to make it squarefree/irreducible.

Remark 3.3. As explained in [88], there might be several distinct triangular decompositions of an ideal, but there are simple ways to pass from one description to another.

3.3.2 Chordal triangularization

We proceed to explain how to compute a triangular chordal network representation of a polynomial set F . We will start with a particular (induced) chordal network that is modified step after step to make it triangular.

Definition 3.7. Let $F \subseteq \mathbb{K}[X]$ be supported on a chordal graph G . The *induced G -chordal network* has a unique node of rank k , namely $F_k := F \cap \mathbb{K}[X_k]$, and its arcs are the same as in the elimination tree, i.e., (F_l, F_p) is an arc if x_p is the parent of x_l .

We will sequentially perform two types of operations to the induced chordal network.

Triangulate(F_l) Let \mathcal{T} be a triangular decomposition of a node F_l of the network. Replace node F_l with one node for each triangular set in \mathcal{T} . Any node which was previously connected to F_l is then connected to each of the new nodes.

Eliminate(T) Let T be a rank l node and let x_p be the parent of x_l . Let $T_p := \text{elim}_p(T)$ and $T_l := T \setminus T_p$. For each arc (T, F_p) we create a new rank p node $F'_p := F_p \cup T_p$, and we substitute arc (T, F_p) with (T, F'_p) . Then, we copy all arcs coming out of F_p to F'_p (while keeping the old arcs). Next, we replace the content of node T with the polynomial set T_l .

The operations are performed in rounds: in the l -th round we triangulate/eliminate all rank l nodes. After each round, we may reduce the network with the following additional operations.

MergeIn(l) Merge any two rank l nodes F_l, F'_l if they define the same ideal, and they have the same sets of incoming arcs.

MergeOut(l) Merge any two rank l nodes F_l, F'_l if they define the same ideal, and they have the same sets of outgoing arcs.

Example 3.7. Consider the polynomial set $F = \{x_0^3 - x_0, x_0x_2 - x_2, x_1 - x_2, x_2^2 - x_2, x_2x_3^2 - x_3\}$, whose associated graph is the star graph (x_2 is connected to x_0, x_1, x_3). Figure 3-5 illustrates

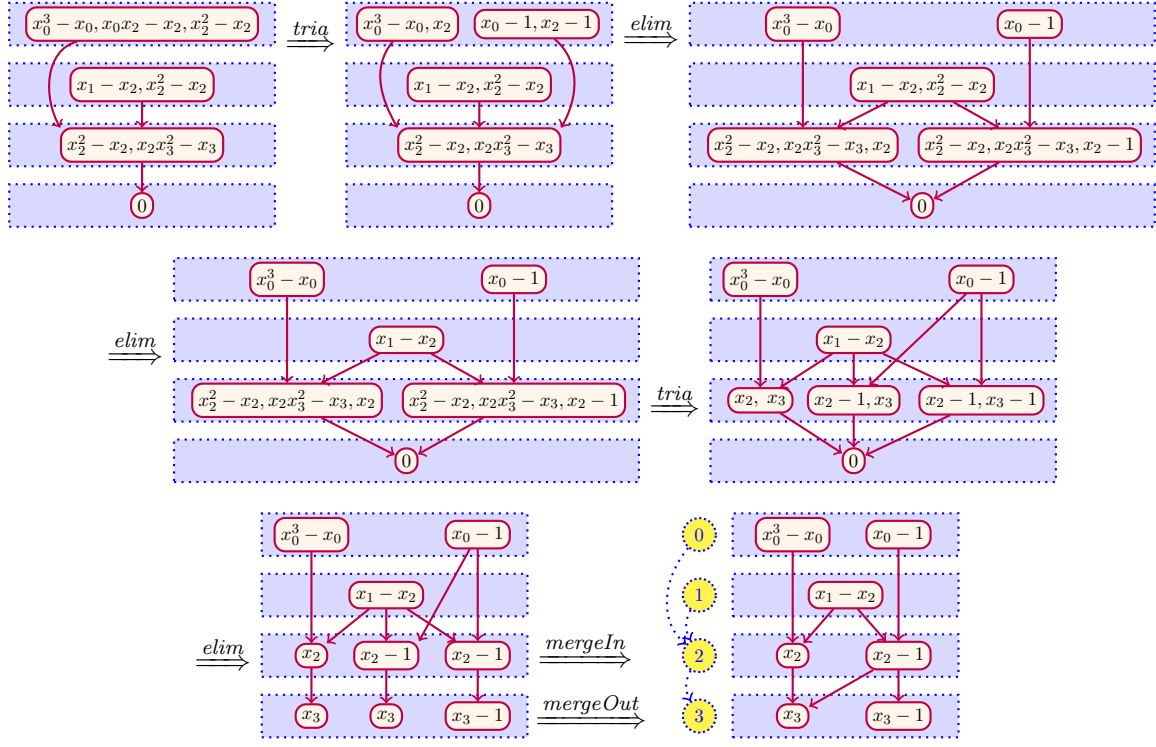


Figure 3-5: Chordal triangularization from Example 3.7.

a sequence of operations (triangulation, elimination and merge) performed on its induced chordal network. The chordal network obtained has three chains:

$$(x_0^3 - x_0, x_1 - x_2, x_2, x_3), (x_0 - 1, x_1 - x_2, x_2 - 1, x_3), (x_0 - 1, x_1 - x_2, x_2 - 1, x_3 - 1).$$

These chains give triangular decomposition of F .

Algorithm 1 presents the chordal triangularization method. The input consists of a polynomial set $F \subseteq \mathbb{K}[X]$ and a chordal graph G . As in the above example, the output of the algorithm is always a triangular chordal network, and it encodes a triangular decomposition of the given polynomial set F .

3.3.3 Algorithm analysis

The objective of this section is to prove that, when the input F is chordally zero-dimensional (Definition 3.4), Algorithm 1 produces a G -chordal network, whose chains give a triangular decomposition of F . As described below, the chordally zero-dimensional assumption is only

Algorithm 1 Chordal Triangularization

Input: Polynomial set $F \subseteq \mathbb{K}[X]$ supported on a chordal graph G

Output: Triangular G -chordal network \mathcal{N} such that $\mathbf{V}(\mathcal{N}) = \mathbf{V}(F)$

```
1: procedure CHORDALNET( $F, G$ )
2:    $\mathcal{N} :=$  induced  $G$ -chordal network of  $F$ 
3:   for  $l = 0 : n - 1$  do
4:     for  $F_l$  node of  $\mathcal{N}$  of rank  $l$  do
5:       TRIANGULATE( $F_l$ )
6:       MERGEOUT( $l$ )
7:       if  $l < n - 1$  then
8:          $x_p :=$  parent of  $x_l$ 
9:         for  $T$  node of  $\mathcal{N}$  of rank  $l$  do
10:          ELIMINATE( $T$ )
11:          MERGEOUT( $p$ )
12:       MERGEIN( $l$ )
13:   return  $\mathcal{N}$ 
```

needed in order for the algorithm to be *well-defined* (recall that up to this point we have only defined triangular decompositions of zero-dimensional systems). Later in the chapter we will see how to extend Algorithm 1 to arbitrary ideals.

Definition 3.8. Let \mathcal{N} be a chordal network, and let $C = (F_0, \dots, F_{n-1})$ be a chain. The *variety* of the chain is $\mathbf{V}(C) := \mathbf{V}(F_0 \cup \dots \cup F_{n-1})$. The *variety* $\mathbf{V}(\mathcal{N})$ of the chordal network is the union of $\mathbf{V}(C)$ over all chains C .

Theorem 3.1. *Let $F \subseteq \mathbb{K}[X]$, supported on chordal graph G , be chordally zero-dimensional. Then the output is a G -chordal network whose chains give a triangular decomposition of F .*

We will split the proof of Theorem 3.1 into several lemmas. We first show that the algorithm is well-defined, i.e., we only perform triangulation operations (line 5) on nodes F_l that define zero-dimensional ideals.

Lemma 3.2. *Let $F \subseteq \mathbb{K}[X]$ be chordally zero-dimensional. Then in Algorithm 1 every triangulation operation is performed on a zero-dimensional ideal.*

Proof. See Section 3.8.1. □

We now show that the chordal structure is preserved during the algorithm.

Lemma 3.3. *Let \mathcal{N} be a G -chordal network. Then the result of performing a triangulation or elimination operation is also a G -chordal network.*

Proof. Consider first a triangulation operation. Note that if $F_l \subseteq \mathbb{K}[X_l]$ then each T in a triangular decomposition is also in $\mathbb{K}[X_l]$. Consider now an elimination operation. Let $T \subseteq \mathbb{K}[X_l]$ and $F_p \subseteq \mathbb{K}[X_p]$ be two adjacent nodes. Using Lemma 2.3, $\text{elim}_p(T) \subseteq \mathbb{K}[X_l \setminus \{x_l\}] \subseteq \mathbb{K}[X_p]$. Thus, the new node $F'_p := F_p \cup \text{elim}_p(T) \subseteq \mathbb{K}[X_p]$. It is clear that for both operations the layered structure of \mathcal{N} is preserved (i.e., arcs follow the elimination tree). \square

We next show that the chains of the output network are triangular sets.

Lemma 3.4. *The output of Algorithm 1 is a triangular G -chordal network.*

Proof. Let $T \subseteq \mathbb{K}[X_l]$ be a rank l node for which we will perform an elimination operation. Note that T must be triangular as we previously performed a triangulation operation. Therefore, there is a unique polynomial $f \in T$ with $\text{mvar}(f) = x_l$. When we perform the elimination operation this is the only polynomial of T we keep, which concludes the proof. \square

Finally, we show that the variety is preserved during the algorithm.

Lemma 3.5. *Let \mathcal{N} be the output of Algorithm 1. Then $\mathbf{V}(\mathcal{N}) = \mathbf{V}(F)$, and moreover, any two chains of \mathcal{N} have disjoint varieties.*

Proof. Let us show that the variety is preserved when we perform triangulation, elimination and merge operations. Firstly, note that a merge operation does not change set of chains of the network, so the variety is preserved. Consider now the case of a triangulation operation. Let \mathcal{N} be a chordal network and let F be one of its nodes. Let \mathcal{T} be a triangular decomposition of F , and let \mathcal{N}' be the chordal network obtained after replacing F with \mathcal{T} . Let C be a chain of \mathcal{N} containing F , and let $C' = C \setminus \{F\}$. Then

$$\mathbf{V}(C) = \mathbf{V}(C') \cap \mathbf{V}(F) = \mathbf{V}(C') \cap \left(\bigcup_{T \in \mathcal{T}} \mathbf{V}(T) \right) = \bigcup_{T \in \mathcal{T}} \mathbf{V}(C') \cap \mathbf{V}(T) = \bigcup_{T \in \mathcal{T}} \mathbf{V}(C' \cup \{T\}).$$

Note that $C' \cup \{T\}$ is a chain of \mathcal{N}' . Moreover, all chains of \mathcal{N}' that contain one of the nodes of \mathcal{T} have this form. Thus, the triangulation step indeed preserves the variety.

Finally, consider the case of an elimination operation. Let $T \subseteq \mathbb{K}[X_l]$ be a node, let (T, F_p) be an arc and let $F'_p = F_p \cup \text{elim}_p(T)$, $T_l = T \setminus \text{elim}_p(T)$. Let \mathcal{N}' be the network obtained after an elimination step on T . It is clear that

$$\mathbf{V}(T \cup F_p) = \mathbf{V}(T_l \cup \text{elim}_p(T) \cup F_p) = \mathbf{V}(T_l \cup F'_p).$$

Since a chain in \mathcal{N} containing T, F_p turns into a chain in \mathcal{N}' containing T_l, F'_p , we conclude that the elimination step also preserves the variety. \square

Proof of Theorem 3.1. We already proved the theorem, since we showed that: the algorithm is well-defined (Lemma 3.2), chordal structure is preserved (Lemma 3.3) and the chains in the output are triangular sets (Lemma 3.4) that decompose the given variety (Lemma 3.5). \square

3.3.4 Radical and irreducible decompositions

We just showed that Algorithm 1 can compute chordal network representations of some zero-dimensional problems. However, we sometimes require additional properties of the chordal network. In particular, in Section 3.4 we will need squarefree representations, i.e., such that any chain generates a radical ideal. As shown next, we can obtain such representations by making one change in Algorithm 1: whenever we perform a triangulation operation, we should produce a squarefree decomposition.

Proposition 3.6. *Assume that all triangular decompositions computed in Algorithm 1 are squarefree. Then any chain of the output network generates a radical ideal.*

Proof. See Section 3.8.1 \square

Instead of radicality, we could further ask for an irreducible representation, i.e., such that any chain generates a prime ideal. The obvious modification to make is to require all triangulation operations to produce irreducible decompositions. Unfortunately, this does not always work. Indeed, we can find irreducible univariate polynomials $f \in \mathbb{K}[x_0]$, $g \in \mathbb{K}[x_1]$ such that $\langle f, g \rangle \subseteq \mathbb{K}[x_0, x_1]$ is not prime (e.g., $f = x_0^2 + 1, g = x_1^2 + 1$).

Nonetheless, there is an advantage of maintaining prime ideals through the algorithm: it gives a simple bound on the size of the triangular network computed, as shown next. This bound will be used when analyzing the complexity of the algorithm.

Lemma 3.7. *Assume that all triangular decompositions computed in Algorithm 1 are irreducible. Then the number of rank l nodes in the output is at most $|\mathbf{V}(F \cap \mathbb{K}[X_l])|$.*

Proof. Let us see that there are at most $|\mathbf{V}(F \cap \mathbb{K}[X_l])|$ rank l nodes after the merge operation from line 6. First note that when we perform this operation any rank l node has an outgoing arc to all rank p nodes (where x_p is the parent of x_l). Therefore, this operation merges any two rank l nodes that define the same ideal. Since these ideals are all maximal, then for any two distinct nodes T_l, T'_l we must have $\mathbf{V}(T_l) \cap \mathbf{V}(T'_l) = \emptyset$. Also note that both $\mathbf{V}(T_l), \mathbf{V}(T'_l)$ are subsets of $\mathbf{V}(F \cap \mathbb{K}[X_l])$. The lemma follows. \square

Remark 3.4. There are other ways to achieve the above bound that do not require computing irreducible decompositions. For instance, we can force the varieties $\mathbf{V}(T_l), \mathbf{V}(T'_l)$ to be disjoint by using ideal saturation.

3.3.5 Complexity

We proceed to estimate the cost of Algorithm 1 in the chordally zero-dimensional case. We will show that the complexity² is $O(nq^{O(\omega)})$, where ω is the treewidth (or clique number) of the graph, and q is a certain degree bound on the polynomials that we formalize below. In particular, when the treewidth ω is bounded the complexity is linear in n and polynomial in the degree bound q .

Definition 3.9 (q -domination). We say that a polynomial set $F_l \subseteq \mathbb{K}[X_l]$ is q -dominated if for each $x_i \in X_l$ there is some $f \in F_l$ such that $\text{mvar}(f) = x_i$, $\text{init}(f) = 1$ and $\deg(f, x_i) \leq q$. Let $F \subseteq \mathbb{K}[X]$ be supported on a chordal graph G . We say that F is *chordally* q -dominated if $F \cap \mathbb{K}[X_l]$ is q -dominated for each maximal clique X_l of graph G .

Example 3.8. The coloring equations in eq. (2.2) are chordally q -dominated since the equations $x_i^q - 1$ are present. Another important example is the case of finite fields \mathbb{F}_q , since if we include the equations $x_i^q - x_i$, as is often done, the problem becomes chordally q -dominated.

Remark 3.5. Observe that if F is chordally q -dominated then it is also chordally zero-dimensional. Conversely, if F is chordally zero-dimensional then we can apply a simple transformation to

² Here the complexity is measured in terms of the number of field operations.

make it chordally q -dominated (for some q). Concretely, for each maximal clique X_l we can enlarge F with a Gröbner basis of $F \cap \mathbb{K}[X_l]$.

We note that we also used the q -dominated condition in [36, 39] to analyze the complexity of chordal elimination. The importance of this condition is that it allows us to easily bound the complexity of computing Gröbner bases or triangular decompositions, as stated next.

Proposition 3.8. *For any q -dominated polynomial set on k variables, the complexity of computing Gröbner bases and (squarefree/irreducible) triangular decompositions is $q^{O(k)}$.*

Proof. See Section 3.8.1. □

The above proposition gives us the cost of the triangulation operations. However, we need to ensure that these operations are indeed performed on a q -dominated ideal, as shown next.

Lemma 3.9. *Let $F \subseteq \mathbb{K}[X]$ be chordally q -dominated. Then in Algorithm 1 any triangulation operation is performed on a q -dominated ideal.*

Proof. The proof is analogous to the one of Lemma 3.2. □

We are ready to estimate the complexity of chordal triangularization. For the analysis we assume that the merge operation from line 6 (resp. line 11) is performed simultaneously with the triangulation (resp. elimination) operations, i.e., as soon as we create a new node we compare it with the previous nodes of the same rank to check if it is repeated.

Lemma 3.10. *Let $F \subseteq \mathbb{K}[X]$ be chordally q -dominated. Assume that all triangular decompositions computed in Algorithm 1 are irreducible. Then throughout the algorithm the width of the network is always bounded by q^ω , independent of the number of variables.*

Proof. This is a consequence of Lemma 3.7. See Section 3.8.1. □

Remark 3.6 (Chordal network of linear size). It follows from the lemma that for fixed q, ω , any chordally q -dominated $F \subseteq \mathbb{K}[X]$ of treewidth ω has a chordal network representation with $O(n)$ nodes.

Theorem 3.11. *Let $F \subseteq \mathbb{K}[X]$ be chordally q -dominated. The complexity of chordal triangularization is $O(nWq^{O(\omega)})$, where W is a bound on the width of the network throughout the algorithm. If all triangulation operations are irreducible, the complexity is $O(nq^{O(\omega)})$.*

Proof. From Proposition 3.8 and Lemma 3.9 we know that each triangulation operation takes $q^{O(\omega)}$, and thus the cost of all triangulations is $O(nWq^{O(\omega)})$. The cost of the elimination operations is negligible. As for the merging operations, we can efficiently verify if a new node is repeated by using a hash table. Thus, the cost of the merging operation is also negligible. Finally, if all triangulation operations are irreducible, then $W \leq q^k$ because of Lemma 3.10. \square

Remark 3.7 (Beyond chordally zero-dimensional). We will later see that, after a suitable redefinition of the triangulation step, Algorithm 1 can also be applied to arbitrary ideals. Nonetheless, the complexity bounds from above do depend on the special structure of the chordally zero-dimensional case. Indeed, solving polynomial equations of treewidth one is NP-hard [39, Ex 1.1], and counting their number of solutions is #P-hard even in the *generic* case for treewidth two (see Corollary 4.25). As a consequence, chordal triangularization will not always run in polynomial time. When using Algorithm 1 in such hard instances we may end up with very high degree polynomials or with a very large number of nodes.

3.4 Computing with chordal networks

Triangular decompositions are one of the most common tools in computational algebraic geometry. The reason is that there are many good algorithms to compute them, and that they can be used to derive several properties of the underlying variety. However, as seen in Example 3.1, the size of the decomposition obtained might be extremely large (exponential) even for very simple cases. Chordal networks can provide a compact representation for these large decompositions. We will see how to effectively use the data structure of chordal networks to compute several properties of the variety.

Let $I = \langle F \rangle$ be a zero-dimensional ideal. We consider the following problems.

Elimination Describe the projection of $\mathbf{V}(I)$ onto the last $n - l$ coordinates.

Zero count Determine the number of solutions, i.e., the cardinality of $\mathbf{V}(I)$.

Sampling Sample random points from $\mathbf{V}(I)$ uniformly.

Radical membership Determine if a polynomial $h \in \mathbb{K}[X]$ vanishes on $\mathbf{V}(I)$, or equivalently, determine if $h \in \sqrt{I}$.

In this section we will develop efficient algorithms for the above problems, given a squarefree chordal network \mathcal{N} (with possibly exponentially many chains). Recall that such network can be obtained as explained in Proposition 3.6. We will see that the first three problems can be solved relatively easily. The radical membership problem is more complicated, and most of this section will be dedicated to it. We note that the algorithms for elimination and radical membership will naturally extend to the positive-dimensional case.

3.4.1 Elimination

The elimination problem is particularly simple, thanks to the elimination property of lexicographic Gröbner bases. For an arbitrary chordal network \mathcal{N} , let $\mathcal{N}_{\geq l}$ denote the subset of \mathcal{N} consisting of nodes of rank k with $k \geq l$. Then $\mathcal{N}_{\geq l}$ is a chordal network representation of the projection of $\mathbf{V}(I)$ onto the last $n - l$ coordinates.

3.4.2 Counting solutions

We want to determine $|\mathbf{V}(\mathcal{N})|$ for a squarefree chordal network \mathcal{N} . Recall from Equation (3.1) that $|\mathbf{V}(T)| = \deg(T)$ for a squarefree triangular set T . Therefore, we just need to compute the sum of $\deg(C)$ over all chains C of the network. We can do this efficiently via dynamic programming, as explained in the following example.

Example 3.9 (Zero count). Let us determine $|\mathbf{V}(\mathcal{N})|$ for the chordal network from Figure 3-4, which corresponds to counting 4-colorings for the blue/solid graph from Figure 2-1a. For a rank l node f_l of the network, let its weight $w(f_l)$ be its degree in x_l . Then we just need to compute $\sum_C \prod_{f_l \in C} w(f_l)$ where the sum is over all chains of the network. We can do this efficiently by successively eliminating the nodes of the network.

Let us first eliminate the nodes of rank 0. Let f_0^a, f_0^b be the two nodes of rank 0, with weights $w(f_0^a) = 3, w(f_0^b) = 2$. Let f_6^a, f_6^b, f_6^c be the nodes of rank 6, with weights $w(f_6^a) = w(f_6^c) = 1, w(f_6^b) = 2$. Note that any chain containing f_6^a must also contain f_0^a . Therefore, we can remove the arc (f_0^a, f_6^a) and update the weight $w(f_6^a) = 1 \times 3$. Similarly, any chain containing f_6^b (or f_6^c) must contain also f_0^b . So we may delete the arcs (f_0^b, f_6^b) and (f_0^b, f_6^c) and update the weights $w(f_6^b) = 2 \times 2, w(f_6^c) = 1 \times 2$. By doing this, we have disconnected, or eliminated, all nodes of rank 0. Continuing this procedure, the final weights obtained for

each rank are shown below. The number of solutions is the last number computed: 10968.

$$\begin{aligned} \text{rk}(0) &\rightarrow [3, 2], & \text{rk}(1) &\rightarrow [3, 2], & \text{rk}(2) &\rightarrow [3, 2], & \text{rk}(3) &\rightarrow [3, 2, 4], \\ \text{rk}(4) &\rightarrow [3, 2, 4], & \text{rk}(5) &\rightarrow [50, 25, 20, 20, 16], & \text{rk}(6) &\rightarrow [3, 4, 2], \\ \text{rk}(7) &\rightarrow [264, 650], & \text{rk}(8) &\rightarrow [2742], & \text{rk}(9) &\rightarrow [10968]. \end{aligned}$$

Algorithm 2 Count solutions

Input: Chordal network \mathcal{N} (triangular, squarefree)

Output: Cardinality of $\mathbf{V}(\mathcal{N})$

```

1: procedure ZEROCOUNT( $\mathcal{N}$ )
2:   for  $f$  node of  $\mathcal{N}$  do
3:      $w(f) := (\text{mdeg}(f) \text{ if } x_{\text{rk}(f)} \text{ is a leaf else } 0)$ 
4:   for  $l = 0 : n - 1$  do
5:     for  $(f_l, f_p)$  arc of  $\mathcal{N}$  with  $\text{rk}(f_l) = l$  do
6:        $w(f_p) := w(f_p) + w(f_l) \text{mdeg}(f_p)$ 
7:   return sum of  $w(f_{n-1})$  over all nodes of rank  $n - 1$ 
```

Algorithm 2 generalizes the above example to arbitrary chordal networks. The complexity is $O(nW^2)$, since we perform one operation for each arc of the network.

3.4.3 Sampling solutions

Uniformly sampling solutions can be done quite easily, by using the partial root counts computed in Algorithm 2. Instead of giving a formal description we simply illustrate the procedure with an example.

Example 3.10 (Sampling). Consider again the chordal network of Figure 3-4. We want to uniformly sample a point $(\hat{x}_0, \dots, \hat{x}_9)$ from its variety, and we follow a bottom up strategy. Let us first choose the value \hat{x}_9 . Since there is a unique rank 9 node $f_9 = x_9^4 - 1$, then \hat{x}_9 must be one of its four roots. Note that each of those roots extend to 2742 solutions (a fourth of the total number of solutions). Therefore, \hat{x}_9 should be equally likely to be any of these roots. Given the value of \hat{x}_9 , we can now set \hat{x}_8 to be any of the three roots of $f_8 = x_8^3 + x_8^2 \hat{x}_9 + x_8 \hat{x}_9^2 + \hat{x}_9^3$, each equally likely. Consider now the two rank 7 nodes f_7^a, f_7^b of degrees 1 and 2. Note that \hat{x}_7 should be either a root of f_7^a or a root of f_7^b (for the given values of \hat{x}_8, \hat{x}_9). In order to sample uniformly, we need to know the number of solutions that

each of those values extend to. From Example 3.9 we know that f_7^a leads to 264 points on the variety, and f_7^b leads to 650. Therefore, we can decide which of them to use based on those weights. Assuming we choose f_7^b , we can now set \hat{x}_7 to be any of its two roots, each equally likely. It is clear how to continue.

3.4.4 Radical membership

In the radical ideal membership problem we want to check whether $h \in \mathbb{K}[X]$ vanishes on $\mathbf{V}(\mathcal{N})$. This is equivalent to determining whether for each chain C of \mathcal{N} the normal form $h_C := h \bmod C$ is identically zero. We will propose a Monte Carlo algorithm to efficiently test this property (without iterating over all chains) under certain structural assumptions on the polynomial h . Our main result is the following.

Theorem 3.12 (Radical membership). *Let $F \subseteq \mathbb{K}[X]$ be chordally q -dominated. Let \mathcal{N} be a chordal network representation of F of width W . Let h be a polynomial that decomposes as $h = \sum_l h_l$ with $h_l \subseteq \mathbb{K}[X_l]$. There is a Monte Carlo algorithm that determines whether h vanishes on $\mathbf{V}(F)$ in $\tilde{O}(nWq^{2\omega} + nW^2q^\omega)$. Here the notation \tilde{O} ignores polynomial factors in the clique number ω .*

Remark 3.8. The theorem is restricted to polynomials h that preserve some of the structure of the graph G , although they may involve all the variables in the ring $\mathbb{K}[X]$ (as opposed to the polynomials of the chordal network). The above mentioned Monte Carlo algorithm also works for other types of polynomials h , but we do not prove complexity bounds for them.

We point out that the above complexity result is far from trivial. To justify this claim we can show that a simple variation of the radical membership problem is NP-hard under very mild assumptions.

Example 3.11 (Zero divisor problem). Consider the zero divisor problem: determine if a polynomial $h \in \mathbb{K}[X]$ vanishes on at least one point of $\mathbf{V}(I)$. Also consider the NP-complete subset sum problem: decide if a set of integers $A = \{a_0, \dots, a_{n-1}\}$ contains a subset whose sum is some given value S . We can reduce it to the zero divisor problem by considering the ideal $I := \langle x_i(x_i - a_i) : 0 \leq i < n \rangle$ and the polynomial $h := \sum_i x_i - S$. Note that the associated graph is the completely disconnected graph ($\omega = 1$) and thus its induced chordal network is already triangular ($W = 1, q = 2$).

We proceed to derive our radical ideal membership test. We will initially assume that the variables of h are all contained in a path of the elimination tree. Later, we will extend the algorithm to polynomials h that decompose into multiple paths of the elimination tree. Finally, we will prove the complexity bound from Theorem 3.12.

Membership on a path

Consider the case where the elimination tree of the graph G is a path (i.e., it has only one leaf). Alternatively, we can assume that all the variables of h are contained in a path of the elimination tree. As before, let $h_C := h \bmod C$ denote the normal form with respect to chain C . Our radical ideal membership test is based on two simple ideas. Firstly, we will check whether the polynomial $H(X) := \sum_C r_C h_C(X)$ is identically zero, for some random coefficients $r_C \in \mathbb{K}$. Clearly, for sufficiently generic values of r_C , the polynomial $H(X)$ will be zero if and only if each h_C is zero. The second idea is that we evaluate $H(X)$ in some random points $\hat{x}_i \in \mathbb{K}$. Thus, we just need to check whether the scalar $H(\hat{X}) \in \mathbb{K}$ is zero. We illustrate how the algorithm works through the following example.

Example 3.12 (Radical membership). Consider again the chordal network of Figure 3-4. Let us verify that the polynomial $h(x)$ from Figure 3-6 vanishes on its variety. We need to show that the reduction (normal form) of h by each chain of the network is zero. As in the case of counting solutions, we will achieve this by successively eliminating nodes. Note that the variables of h are $\{x_0, x_6, x_7, x_8, x_9\}$, which correspond to a path of the elimination tree. Thus, we restrict ourselves to the part of the network given by these variables, as shown in Figure 3-6.

Let us start by processing the two nodes of rank 0. We have to compute the reduction of $h(x)$ modulo each of these nodes. Afterwards, we will substitute x_0 in these reduced polynomials with a random value on \mathbb{K} ; in this case we choose $\hat{x}_0 = 1$. Let h_0^a, h_0^b be the polynomials obtained after the reduction and substitution, as shown in Figure 3-6. These two polynomials will be sent to the adjacent rank 6 nodes.

Consider now a rank p node f_p that receives certain polynomials from its adjacent rank l nodes. We now perform a random linear combination of these incoming polynomials, then we reduce this linear combination modulo f_p , and lastly we substitute x_p with a random value \hat{x}_p .

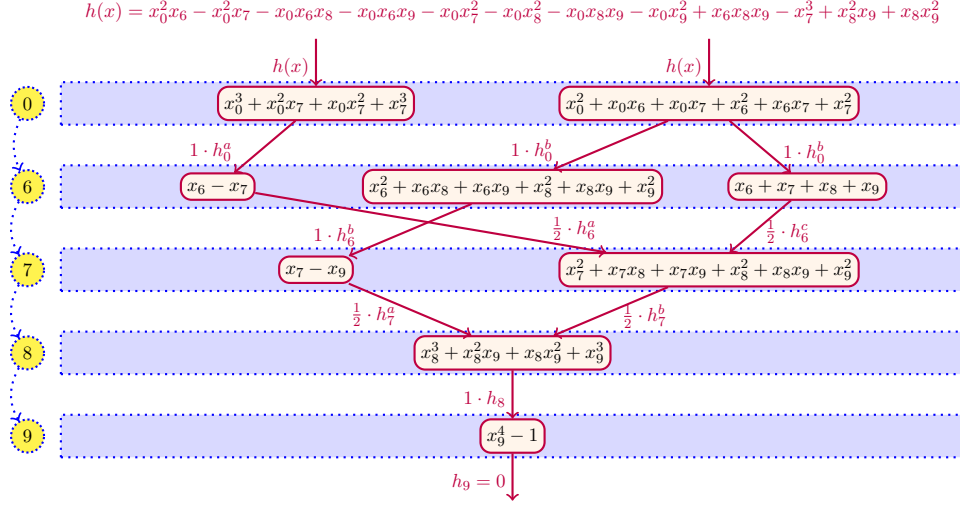


Figure 3-6: Sketch of the radical ideal membership test from Example 3.12.

For this example the linear combination will be an average, and the random points \hat{x}_p will be one. Figure 3-6 indicates the polynomials received and output by each node. For instance, h_8 is obtained by reducing $\frac{1}{2}(h_7^a + h_7^b)$ modulo $f_8 = x_8^3 + x_8^2x_9 + x_8x_9^2 + x_9^3$ and then plugging in $\hat{x}_8 = 1$. The polynomials obtained with this procedure are shown below. Note that the last polynomial computed is zero, agreeing with the fact that $h(x)$ vanishes on the variety.

$$\begin{aligned}
h_0^a &= x_6x_8x_9 - x_6x_8 - x_6x_9 + x_6 - x_7^3 - x_7^2 - x_7 + x_8^2x_9 - x_8^2 + x_8x_9^2 - x_8x_9 - x_9^2 \\
h_0^b &= -x_6^3 - x_6^2 + x_6x_8x_9 - x_6x_8 - x_6x_9 + x_8^2x_9 - x_8^2 + x_8x_9^2 - x_8x_9 - x_9^2 \\
h_6^a &= -x_7^3 - x_7^2 + x_7x_8x_9 - x_7x_8 - x_7x_9 + x_8^2x_9 - x_8^2 + x_8x_9^2 - x_8x_9 - x_9^2 \\
h_6^b &= -x_8^3 - x_8^2x_9 - x_8x_9^2 - x_9^3 \\
h_6^c &= x_7^3 + x_7^2(3x_8 + 3x_9 - 1) + x_7(3x_8^2 + 5x_8x_9 - x_8 + 3x_9^2 - x_9) + x_8^3 + 3x_8^2x_9 - x_8^2 + 3x_8x_9^2 - x_8x_9 + x_9^3 - x_9^2 \\
h_7^a &= h_7^b = -x_8^3 - x_8^2x_9 - x_8x_9^2 - x_9^3 \\
h_8 &= h_9 = 0
\end{aligned}$$

Algorithm 3 generalizes the procedure from the above example. Observe that each node f_l of the network has an associated polynomial $H(f_l)$, which is first reduced modulo f_l , then we substitute the value \hat{x}_l and finally we pass this polynomial to the adjacent nodes. Also note that that we choose one random scalar \hat{x}_l for each variable, and one random scalar r_{lp} for each arc of the network.

Algorithm 3 Radical ideal membership

Input: Chordal network \mathcal{N} (triangular, squarefree) and polynomial $h(x)$ such that all its variables are contained in a path of the elimination tree.

Output: True, if h vanishes on $\mathbf{V}(\mathcal{N})$. False, otherwise.

```
1: procedure RIDEALMEMBERSHIP( $\mathcal{N}, h$ )
2:    $x_m := \text{mvar}(h)$ 
3:   for  $f$  node of  $\mathcal{N}$  do
4:      $H(f) := (h \text{ if } \text{rk}(f) = m \text{ else } 0)$ 
5:   for  $l = 0 : n - 1$  do
6:      $\hat{x}_l := \text{random scalar}$ 
7:     for  $f_l$  node of  $\mathcal{N}$  of rank  $l$  do
8:        $H(f_l) := H(f_l) \bmod f_l$ 
9:       plug in  $\hat{x}_l$  in  $H(f_l)$ 
10:      for  $(f_l, f_p)$  arc of  $\mathcal{N}$  do
11:         $r_{lp} := \text{random scalar}$ 
12:         $H(f_p) := H(f_p) + r_{lp} H(f_l)$ 
13:   return (True if  $H(f_{n-1}) = 0$  for all rank  $n - 1$  nodes else False)
```

Correctness

We proceed to show the correctness of Algorithm 3. We will need a preliminary lemma and some new notation. For any l , let X^l denote the subtree of the elimination tree consisting of x_l and all its descendants (e.g., X^{n-1} consists of all variables). For a rank l node f_l of the network, we will say that an f_l -subchain C_l is the subset of a chain C , with $f_l \in C$, restricted to nodes of rank i for some $x_i \in X^l$.

Lemma 3.13. *Let \mathcal{N} be a chordal network whose elimination tree is a path, and let $h \in \mathbb{K}[X]$. Let f_l be a rank l node of \mathcal{N} . In Algorithm 3, the final value of $H(f_l)$ is given by plugging in the values $\hat{x}_1 \hat{x}_2, \dots, \hat{x}_l$ in the polynomial*

$$\sum_{C_l} r_{C_l} h \bmod C_l,$$

where the sum is over all f_l -subchains C_l , and where r_{C_l} denotes the product of the random scalars r_{ij} along the subchain C_l .

Proof. See Section 3.8.2. □

Theorem 3.14. *Let \mathcal{N} be a chordal network, triangular and squarefree, and let q be a bound*

on the main degrees of its nodes. Let $h \in \mathbb{K}[X]$ be such that all its variables are contained in a path of the elimination tree. Algorithm 3 behaves as follows:

- if h vanishes on $\mathbf{V}(\mathcal{N})$, it always returns “True”.
- if not, it returns “False” with probability at least $1/2$, assuming that the random scalars r_{lp}, x_l are chosen (i.i.d. uniform) from some set $S \subseteq \mathbb{K}$ with $|S| \geq 2nq$.

Proof. Denoting $h_C := h \bmod C$, Lemma 3.13 tells us that Algorithm 3 checks whether $\sum_C r_C h_C(\hat{X}) = 0$, where r_C is the product of all scalars r_{lp} along the chain C . If h vanishes on $\mathbf{V}(\mathcal{N})$, then each h_C is zero and thus the algorithm returns “True”. Assume now that h does not vanish on $\mathbf{V}(\mathcal{N})$, and thus at least one h_C is nonzero. Let R be the set of all random scalars r_{lp} used in the algorithm, which we now see as variables. Consider the polynomial

$$H(X, R) := \sum_C r_C(R) h_C(X),$$

and note that it is nonzero. Observe that the degree of $H(X, R)$ is at most nq , since $\deg(r_C) \leq n$ and $\deg(h_C) \leq n(q-1)$. Using the Schwartz-Zippel lemma (see e.g., [140, §6.9]), the probability that H evaluates to zero for random values $r_{lp}, \hat{x}_l \in S$ is at most $nq/|S| \leq 1/2$. \square

Remark 3.9. The above theorem requires that \mathbb{K} contains sufficiently many elements. If necessary, we may consider a field extension $\mathbb{L} \supseteq \mathbb{K}$ and perform all computations over $\mathbb{L}[X]$.

Combining multiple paths

We now extend Algorithm 3 to work for other polynomials h . Specifically, we assume that the polynomial can be written as $h = \sum_i h_i$ where the variables of each h_i belong to a path of the elimination tree. We let $x_{m_i} := \text{mvar}(h_i)$ denote the main variables, and we can assume that they are all distinct. We only need to make two simple modifications to Algorithm 3.

- Previously, we initialized the algorithm with nonzero values in a single rank (see line 4). We now initialize the algorithm in multiple ranks: $H(f_{m_i}) = h_i$ if $\text{rk}(f_{m_i}) = m_i$.
- When combining the incoming polynomials to a node f_p , we now take a random *affine* combination (i.e., $\sum_l r_{lp} H(f_l)$ for some scalars r_{lp} such that $\sum_l r_{lp} = 1$). Note that

in the example from Figure 3-6 we took the average of the incoming nodes, so this condition is satisfied.

The first modification is quite natural given the decomposition of the polynomial h . The second item is less intuitive, but it is simply a normalization to ensure that all polynomials h_i are scaled in the same manner. The correctness of this modified algorithm follows from the fact that Lemma 3.13 remains valid, as shown next.

Lemma 3.15. *Let \mathcal{N} be a chordal network and let $h = \sum_i h_i \in \mathbb{K}[X]$ be such that the variables of each h_i are contained in a path of the elimination tree. With the above modifications to Algorithm 3, the final value of $H(f_l)$ is as stated in Lemma 3.13.*

Proof. See Section 3.8.2. □

Remark 3.10. Note that any h can be written as $h = \sum_{i=0}^{n-1} h_i$, where h_i corresponds to the terms with main variable x_i . Even when the elimination tree is a path, it is usually more efficient to decompose it in this manner and use Algorithm 3 with the above modifications.

Complexity

We finally proceed to prove the complexity bound from Theorem 3.12. We restrict ourselves to polynomials h that preserve the sparsity structure given by the chordal graph G . More precisely, we assume that the variables of each of the terms of h correspond to a clique of G , or equivalently, that $h = \sum_l h_l$ for some $h_l \in \mathbb{K}[X_l]$. Naturally, we will use Algorithm 3 with the two modifications from above. The key idea to notice is that Algorithm 3 preserves chordality, as stated next.

Lemma 3.16. *Assume that in Algorithm 3 the initial values of $H(f_l)$ are such that $H(f_l) \subseteq \mathbb{K}[X_l]$ (where $\text{rk}(f_l) = l$). Then the same condition is satisfied throughout the algorithm.*

Proof. The update rule used in Algorithm 3 is of the form $H(f_p) := H(f_p) + r_{lp}\phi_l(\tilde{h}_l)$ for some $\tilde{h}_l \in \mathbb{K}[X_l]$, where ϕ_l denotes the functional that plugs in \hat{x}_l . Using Lemma 2.3, we have $\phi_l(\tilde{h}_l) \subseteq \mathbb{K}[X_l \setminus \{x_l\}] \subseteq \mathbb{K}[X_p]$. The result follows. □

Proof of Theorem 3.12. We consider Algorithm 3 with the modifications (i) and (ii) from above. Note that the q -dominated condition allows us to bound the degrees of all polynomials

computed in Algorithm 3. Furthermore, since chordality is preserved (Lemma 3.16), then all polynomials will have at most q^ω terms. The complexity of the algorithm is determined by the cost of polynomial divisions and polynomial additions. Polynomial addition takes linear time in the number of terms, and it is performed once for each arc of the network. Thus, their total cost is $O(nW^2q^\omega)$. As for polynomial division, $h \bmod f$ can be obtained in $O(|h| |f| \log |f|)$, where $|\cdot|$ denotes the number of terms [99]. Their total cost is $\tilde{O}(nWq^{2\omega})$, since there is one operation per node of the network. \square

3.5 Monomial ideals

We already showed how to compute chordal network representations of some zero-dimensional ideals. Before proceeding to the general case, we will consider the special class of monomial ideals. Recall that an ideal is monomial if it is generated by monomials. Monomial ideals might have positive-dimension, but their special structure makes their analysis particularly simple. As in Example 3.2, we will see that any monomial ideal admits a compact chordal network representation. We will also show how such chordal network can be effectively used to compute its dimension, its equidimensional components, and its irreducible components. These methods will be later generalized to arbitrary polynomial ideals.

3.5.1 Chordal triangularization

Algorithm 1 will be exactly the same for monomial ideals as in the zero-dimensional case. The only difference is that for the triangulation operations we need to specify the type of decomposition used, as explained now.

We will say that a set of monomials T is *triangular* if it consists of variables, i.e., $T = \{x_{i_1}, \dots, x_{i_m}\}$. It is well known that a monomial ideal is prime if and only if it is generated by variables. It is also known that the minimal primes of a monomial ideal are also monomial. It follows that any monomial ideal I *decomposes* as $\mathbf{V}(I) = \bigcup_T \mathbf{V}(T)$, where the union is over some triangular monomial sets T .

By using the above decomposition in each triangulation operation, chordal triangularization can now be applied to monomial ideals, as established in the proposition below. We point out that even though this decomposition seems quite different from the one of Section 3.3.1,

both are special instances of more general theory that will be discussed in Section 3.6.1.

Proposition 3.17. *Let F be a set of monomials supported on a chordal graph G . Algorithm 1 computes a G -chordal network, whose chains give a triangular decomposition of F .*

Proof. Proving that the variety is preserved in the algorithm is essentially the same as for the chordally zero-dimensional case (Lemma 3.5). It is straightforward to see that the chains of the output are triangular (i.e., they consist of variables). \square

Example 3.13. Consider the ideal $I = \langle x_0x_1, x_0x_2, x_0x_3, x_1x_2, x_1x_4, x_2x_5, x_3x_4, x_3x_5, x_4x_5 \rangle$. The result of chordal triangularization is shown to the left of Figure 3-7.

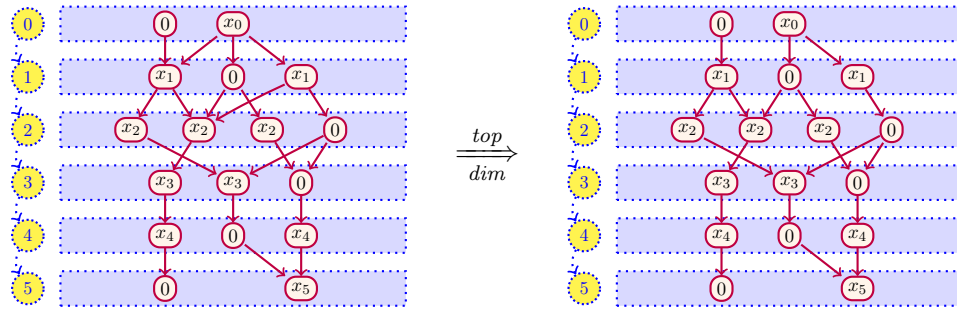


Figure 3-7: Chordal network from Example 3.13, and its top-dimensional part.

As in the chordally zero-dimensional case, we can also prove that the complexity is linear in n when the treewidth is bounded.

Theorem 3.18. *Let F be a set of monomials supported on a chordal graph G of clique number ω . Then F can be represented by a triangular chordal network with at most $n2^\omega$ nodes, which can be computed in time $O(n2^{O(\omega)})$.*

Proof. Note that after the l -th triangulation round we will have at most 2^ω rank l nodes, since the triangular monomial sets in $\mathbb{K}[X_l]$ are in bijection with the subsets of X_l . A similar argument proves that the width of the network is bounded by 2^ω after an elimination round, and thus throughout the algorithm. The cost of computing a triangular decomposition in $\mathbb{K}[X_l]$ is polynomial in $2^{|X_l|}$, since we can simply enumerate over all possible triangular monomial sets. Thus, the cost of all triangulation operations is $O(nW2^{O(\omega)}) = O(n2^{O(\omega)})$. The cost of the elimination and merging operations is negligible. \square

3.5.2 Computing with chordal networks

Let \mathcal{N} be a chordal network representation of a monomial ideal I . We will show how to effectively use \mathcal{N} to solve the following problems:

Dimension Determine the dimension of I .

Top-dimensional part Describe the top-dimensional part of $\mathbf{V}(I)$.

Irreducible components Determine the minimal primes of I .

The above problems can be shown to be hard in general by using the correspondence between minimal vertex covers of a graph and the irreducible components of its edge ideal (see Example 3.2). We will see that, given the chordal network, the first two problems can be solved in linear time with a dynamic program. The third one is much more complicated, since we need to enumerate over all chains of the network to verify if they are minimal. In order to do this efficiently, we will need to address the following problems.

Dimension count Classify the number of chains C of \mathcal{N} according to its dimension.

Isolate dimension d Enumerate all chains C of \mathcal{N} such that $\dim(\mathbf{V}(C)) = d$.

We proceed to solve each of the problems from above. To simplify the exposition, we will assume for this section that the elimination tree is a path, but it is not difficult to see that all these methods will work for arbitrary chordal networks.

Dimension

Let us see that it is quite easy to compute the dimension of $\mathbf{V}(\mathcal{N})$. Since the variety $\mathbf{V}(T)$ of a triangular monomial set is a linear space, its dimension is $\dim(\mathbf{V}(T)) = n - |T|$. Therefore, $\dim(\mathbf{V}(\mathcal{N})) = n - \min_C |C|$, where the minimum is taken over all chains of the network. Note that we ignore the zero entries of C . In particular, for the network in Figure 3-7 we have $\dim(\mathbf{V}(\mathcal{N})) = 6 - 4 = 2$.

We reduced the problem to computing the smallest cardinality of a chain of \mathcal{N} . This can be done using a simple dynamic program, which is quite similar to the one in Algorithm 2. For each node f_i we save the value $\ell(f_i)$ corresponding to the length of the shortest chain up to

level l . For an arc (f_l, f_p) with $f_p \neq 0$, the update rule is simply $\ell(f_p) := \min(\ell(f_p), 1 + \ell(f_l))$. It follows that we can compute in linear time the dimension of $\mathbf{V}(\mathcal{N})$.

Top-dimensional part

We can get a chordal network \mathcal{N}_{top} describing its top-dimensional part by modifying the procedure that computes the dimension. Indeed, assume that for some arc (f_l, f_p) we have $\ell(f_p) < 1 + \ell(f_l)$ and thus the update $\ell(f_p) := \min(\ell(f_p), 1 + \ell(f_l))$ is not needed. This means that the arc (f_l, f_p) is unnecessary for the top-dimensional component. By pruning the arcs of \mathcal{N} in such manner we obtain the wanted network \mathcal{N}_{top} .

Example 3.14. Let \mathcal{N} be the network on the left of Figure 3-7. Note that \mathcal{N} has 9 chains, two of them are $C_1 = (x_1, x_2, x_3, x_5)$, $C_2 = (x_0, x_1, x_2, x_3, x_5)$, of dimensions 2 and 1. By pruning some arcs, we obtain its highest dimensional part \mathcal{N}_{top} , shown to the right of Figure 3-7. This network \mathcal{N}_{top} only has 6 chains; note that C_2 is not one of them. In this case neither of the chains removed was minimal (e.g., $C_2 \supsetneq C_1$), so that $\mathbf{V}(\mathcal{N}) = \mathbf{V}(\mathcal{N}_{top})$. Thus, both \mathcal{N} and \mathcal{N}_{top} are valid chordal network representations of the ideal from Example 3.13, although the latter is preferred since all its chains are minimal. Similarly, the network from Figure 3-2 was obtained by using chordal triangularization and then computing its highest dimensional part.

Irreducible components

Chordal triangularization can also aid in computing the minimal primes of an ideal (geometrically, the irreducible components). In the monomial case, any chain of \mathcal{N} defines a prime ideal, and thus we only need to determine which chains are minimal with respect to containment. In some cases it is enough to prune certain arcs of the network (e.g., Figure 3-7), but this is not always possible.

Unfortunately, we do not know a better procedure than simply iterating over all chains of the network checking for minimality. Nonetheless, we can make this method much more effective by proceeding in order of decreasing dimension. This simple procedure is particularly efficient when we are only interested in the minimal primes of high dimension, as will be seen in Section 3.7.1. In the remaining of the section we will explain how to enumerate the chains by decreasing dimension (this is precisely the dimension isolation problem).

Dimension count

Classifying the number of chains according to its dimension can be done with a very similar dynamic program as for computing the dimension. As discussed above, the dimension of a chain is simply given by its cardinality. For a rank l node f_l of the network and for any $0 \leq k \leq l + 1$, let $c_k(f_l)$ denote the number of chains of the network (up to level l) with cardinality exactly k . Then for an arc (f_l, f_p) with $f_p \neq 0$ the update rule is simply $c_k(f_p) := c_k(f_p) + c_{k-1}(f_l)$.

Dimension isolation

For simplicity of exposition we only describe how to produce one chain C of dimension d , but it is straightforward to then generate all of them. As in Example 3.10, we follow a bottom up strategy, successively adding nodes to the chain. We first need to choose a rank $n - 1$ node f_{n-1} that belongs to at least one chain of dimension d . Using the values $c_k(f_l)$ from above, we can choose any f_{n-1} for which $c_{n-d}(f_{n-1}) \geq 1$. Assuming that we chose some $f_{n-1} \neq 0$, we now need to find an adjacent rank $n - 2$ node f_{n-2} such that $c_{n-d-1}(f_{n-2}) \geq 1$. It is clear how to continue.

3.6 The general case

We finally proceed to compute chordal network representations of arbitrary polynomial ideals. We will also see how the different chordal network algorithms developed earlier (e.g., radical ideal membership, isolating the top-dimensional component) have a natural extension to this general setting.

3.6.1 Regular chains

The theory of triangular sets for positive-dimensional varieties is more involved; we refer to [73, 143] for an introduction. We now present the concept of regular chains, which is at the center of this theory.

A set of polynomials $T \subseteq \mathbb{K}[X] \setminus \mathbb{K}$ is a *triangular set* if its elements have distinct main variables. Let h be the product of the initials (Definition 3.5) of the polynomials in T . The

geometric object associated to T is the *quasi-component*

$$\mathbf{W}(T) := \mathbf{V}(T) \setminus \mathbf{V}(h) \subseteq \overline{\mathbb{K}^n}.$$

The attached algebraic object is the *saturated ideal*

$$\text{sat}(T) := \langle T \rangle : h^\infty = \{f \in \mathbb{K}[X] : h^N f \in \langle T \rangle \text{ for some } N \in \mathbb{N}\}.$$

Note that $\mathbf{V}(\text{sat}(T)) = \overline{\mathbf{W}(T)}$, where the closure is in the Zariski topology.

Polynomial pseudo-division is a basic operation in triangular sets. Let f, g be polynomials of degrees d, e in $x := \text{mvar}(g)$. The basic idea is to see f, g as univariate polynomials in x (with coefficients in $\mathbb{K}[X \setminus \{x\}]$), and in order that we can always divide f by g , we first multiply by some power of $\text{init}(g)$. Formally, the *pseudo-remainder* of f by g is $\text{prem}(f, g) := f$ if $d < e$, and otherwise $\text{prem}(f, g) := \text{init}(g)^{d-e+1} f \bmod (g)$. Pseudo-division can be extended to triangular sets in the natural way. The *pseudo-remainder* of f by $T = \{t_1, \dots, t_k\}$, where $\text{mvar}(t_1) > \dots > \text{mvar}(t_k)$, is

$$\text{prem}(f, T) = \text{prem}(\dots (\text{prem}(f, t_1) \dots, t_k).$$

Definition 3.10. A *regular chain* is a triangular set T such that for any polynomial f

$$f \in \text{sat}(T) \iff \text{prem}(f, T) = 0.$$

Remark 3.11. Note that a zero-dimensional triangular set (Definition 3.5) is a regular chain, since pseudo-reduction coincides with Gröbner bases reduction.

Regular chains have very nice algorithmic properties. In particular, they are always consistent (i.e., $\mathbf{W}(T) \neq 0$), and furthermore $\dim(\mathbf{W}(T)) = n - |T|$. Table 3.1 summarizes some of these properties, comparing them with Gröbner bases.

Definition 3.11. A *triangular decomposition* of a polynomial set F is a collection \mathcal{T} of regular chains, such that $\mathbf{V}(F) = \bigcup_{T \in \mathcal{T}} \mathbf{W}(T)$.

Remark 3.12. There is a weaker notion of decomposition that is commonly used: \mathcal{T} is a *Kalkbrener* triangular decomposition if $\mathbf{V}(F) = \bigcup_{T \in \mathcal{T}} \overline{\mathbf{W}(T)}$.

Table 3.1: Gröbner bases vs. regular chains

	Gröbner basis (\mathcal{G})	Regular chain (T)
Geometric object	$\mathbf{V}(\mathcal{G})$	$\mathbf{W}(T)$
Algebraic object	$\langle \mathcal{G} \rangle$	$\text{sat}(T)$
Feasible	if $1 \notin \mathcal{G}$	always
Ideal membership	Remainder = 0	PseudoRemainder = 0
Dimension	from Hilbert series	$n - T $
Elimination ideal	$\mathcal{G}_{\text{lex}} \cap \mathbb{K}[x_l, \dots, x_{n-1}]$	$T \cap \mathbb{K}[x_l, \dots, x_{n-1}]$

Example 3.15. Let $F = \{x_0x_3 - x_1x_2, x_2x_5 - x_3x_4, x_4x_7 - x_5x_6\}$ consist of the adjacent minors of a 2×4 matrix. It can be decomposed into 8 regular chains:

$$(x_0x_3 - x_1x_2, x_2x_5 - x_3x_4, x_4x_7 - x_5x_6), (x_0x_3 - x_1x_2, x_4, x_5), (x_2, x_3, x_4x_7 - x_5x_6), \\ (x_1, x_3, x_4, x_5), (x_1, x_3, x_5, x_7), (x_2, x_3, x_5, x_7), (x_2, x_3, x_6, x_7), (x_0x_3 - x_1x_2, x_2x_5 - x_3x_4, x_6, x_7).$$

Note that the first three triangular sets (first line) have dimension $8 - 3 = 5$. Observe that the quasi-components $\mathbf{W}(T)$ of these three sets do not cover the points for which $x_3 = x_7 = 0$, which is why we need the remaining five sets. However, these three triangular sets alone give a Kalkbrener decomposition of the variety.

3.6.2 Regular systems

In the study of triangular sets, it is useful to consider systems of polynomials containing both equations $\{f_i(x) = 0\}_i$ and *inequations* $\{h_j(x) \neq 0\}_j$. Following the notation of [143], we say that a *polynomial system* $\mathfrak{F} = (F, H)$ is a pair of polynomial sets $F, H \subseteq \mathbb{K}[X]$, and its associated geometric object is the quasi-variety

$$\mathbf{Z}(\mathfrak{F}) := \{x \in \overline{\mathbb{K}}^n : f(x) = 0 \text{ for } f \in F, h(x) \neq 0 \text{ for } h \in H\}.$$

For instance, the quasi-component $\mathbf{W}(T)$ of a triangular set is the quasi-variety of the polynomial system $(T, \text{init}(T))$, where $\text{init}(T)$ is the set of initials of T .

For a polynomial system $\mathfrak{F} = (F, H)$ we denote by $\text{elim}_p(\mathfrak{F})$ the polynomial system $(\text{elim}_p(F), \text{elim}_p(H))$. We also denote by $\mathfrak{F}_1 + \mathfrak{F}_2$ the concatenation of two polynomial systems, i.e., $(F_1, H_1) + (F_2, H_2) := (F_1 \cup F_2, H_1 \cup H_2)$.

Definition 3.12. A *regular system* is a pair $\mathfrak{T} = (T, U)$ such that T is triangular and for any

$0 \leq k < n$:

(i) either $T^{(k)} = \emptyset$ or $U^{(k)} = \emptyset$, where the superscript $\langle k \rangle$ denotes the polynomials with main variable x_k .

(ii) $\text{init}(f)(\hat{x}^{k+1}) \neq 0$ for any $f \in T^{(k)} \cup U^{(k)}$ and $\hat{x}^{k+1} \in \mathbf{Z}(\text{elim}_{k+1}(\mathfrak{T})) \subseteq \overline{\mathbb{K}}^{n-k-1}$.

A regular system is *squarefree* if the polynomials $f(x_k, \hat{x}^{k+1}) \in \mathbb{K}[x_k]$ are squarefree for any $f \in T^{(k)} \cup U^{(k)}$ and any $\hat{x}^{k+1} \in \mathbf{Z}(\text{elim}_{k+1}(\mathfrak{T})) \subseteq \overline{\mathbb{K}}^{n-k-1}$.

For a regular system (T, U) the set T is a regular chain, and conversely, for a regular chain T there is some U such that (T, U) is a regular system [142]. Wang showed how to decompose any polynomial system in characteristic zero into (squarefree) regular systems [142, 143].

Definition 3.13. A *triangular decomposition* of a polynomial system \mathfrak{F} is a collection \mathcal{T} of regular systems, such that $\mathbf{Z}(\mathfrak{F}) = \bigcup_{\mathfrak{T} \in \mathcal{T}} \mathbf{Z}(\mathfrak{T})$.

Remark 3.13 (Binomial ideals). Consider a polynomial system $\mathfrak{F} = (F, U)$ such that F consists of binomials (two terms) and $H = \{x_{i_1}, \dots, x_{i_m}\}$ consists of variables. We can decompose \mathfrak{F} into regular systems $\mathfrak{T} = (T, U)$ that preserve the binomial structure. Assume first that $H = \{x_0, \dots, x_{n-1}\}$ contains all variables. Equivalently, we are looking for the zero set of F on the torus $(\overline{\mathbb{K}} \setminus \{0\})^n$. It is well known that F can be converted to (binomial) triangular form T by computing the Hermite normal form of the matrix of exponents [128, §3.2], and the inequations U correspond to the non-pivot variables. For an arbitrary H , we can enumerate over the choices of nonzero variables.

3.6.3 Chordal triangularization

Algorithm 1 extends to the positive-dimensional case in the natural way, although with one important difference: the nodes of the chordal network will be polynomial systems, i.e., pairs of polynomial sets. We now describe the modifications of the main steps of the algorithm:

Initialization The nodes of the induced G -chordal network are now of the form $\mathfrak{F}_l = (F_l, H_l)$, where $F_l = F \cap \mathbb{K}[X_l]$ and $H_l = \emptyset$.

Triangulation For a node \mathfrak{F}_l we decompose it into regular systems \mathfrak{T} and we replace \mathfrak{F}_l with a node for each of them.

Elimination Let \mathfrak{F}_l be the rank l node we will eliminate, and let \mathfrak{F}_p be an adjacent rank p node. Then we create a rank p node $\mathfrak{F}'_p := \mathfrak{F}_p + \text{elim}_p(\mathfrak{F}_l)$.

Termination After all triangulation/elimination operations, we may remove the inequations from the nodes of the network, i.e., replace $\mathfrak{F} = (F, H)$ with F .

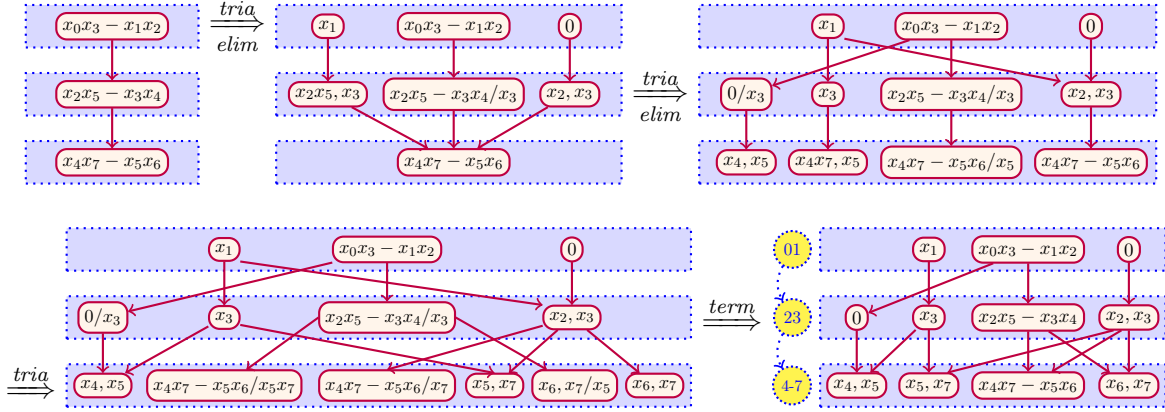


Figure 3-8: Chordal triangularization from Example 3.16.

Example 3.16. Figure 3-8 illustrates the chordal triangularization algorithm for the polynomial set F from Example 3.15. The nodes of the chordal network are polynomial systems $\mathfrak{F} = (F, H)$, which we represent in the figure as (F/H) . Note that in the termination step, after all triangulation/elimination operations, we remove the inequations to simplify the network. The final network has 8 chains, which coincide with the triangular decomposition from Example 3.15.

We can now compute chordal network representations of arbitrary systems.

Theorem 3.19. *Let $F \subseteq \mathbb{K}[X]$ be supported on a chordal graph G . With the above modifications, Algorithm 1 computes a G -chordal network \mathcal{N} , whose chains give a triangular decomposition of F . Furthermore, this decomposition is squarefree if all triangulation operations are squarefree.*

Proof. See Section 3.8.3. □

Remark 3.14. We have noticed that chordal triangularization is quite efficient for binomial ideals. Remark 3.13 partly explains this observation. However, we do not yet know whether it will always run in polynomial time when the treewidth is bounded.

3.6.4 Computing with chordal networks

We just showed how to compute chordal network representations of arbitrary polynomial systems. We now explain how to extend the chordal network algorithms from Section 3.4 and Section 3.5.2 to the general case.

Elimination

Since regular chains possess the same elimination property as lexicographic Gröbner bases, the approach from Section 3.4.1 works in the same way.

Radical ideal membership

Algorithm 3 extends to the positive-dimensional case simply by replacing polynomial division with pseudo-division. Note that we require a squarefree chordal network, which can be computed as explained in Theorem 3.19.

Dimension and equidimensional components

The dimension of a regular chain T is $n - |T|$, which is the same as for the monomial case. Thus, we can compute the dimension as in Section 3.5.2. Similarly, we can compute a chordal network describing the highest dimensional component, and also isolate any given dimension of the network.

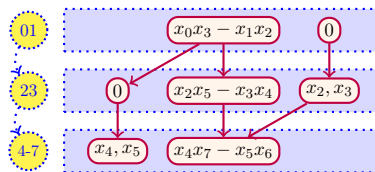


Figure 3-9: Top-dimensional part of the chordal network from Figure 3-8.

Example 3.17. Figure 3-9 shows the highest dimensional part of the chordal network from Figure 3-8. This network only has 3 chains, which give a Kalkbrener decomposition of the variety (see Example 3.15). Likewise, the chordal network from Figure 3-3 gives a Kalkbrener triangular decomposition of the ideal of adjacent minors of a 2×7 matrix.

Irreducible components

Unlike the monomial case, the chains of an arbitrary chordal network \mathcal{N} will not necessarily define prime ideals (see Section 3.3.4). However, in some interesting cases it will be the case, thanks to the following well known property.

Theorem 3.20. *Let $T = \{t_1, \dots, t_k\}$ be a regular chain. Assume that $\text{mdeg}(t_i) = 1$ for $1 \leq i < k$ and that t_k is an irreducible polynomial. Then $\text{sat}(T)$ is a prime ideal.*

Proof. This follows from [143, Thm 6.2.14]. □

In particular, note that all chains of the chordal network from Figure 3-3 are of this form. We will see in Section 3.7.3 that the same holds for other families of ideals. Assume now that all chains of the network define a prime ideal. A plausible strategy to compute all minimal primes (or only the high dimensional ones) is as follows:

- (i) Iterate over all chains T of the network in order of decreasing dimension.
- (ii) For a chain C , and a minimal prime I' previously found, determine whether $I' \subseteq \text{sat}(C)$ by checking whether $\text{prem}(f, C) = 0$ for each generator f of I' .
- (iii) If $I := \text{sat}(C)$ does not contain any previously found prime, compute generators for I by using Gröbner bases. We have a new minimal prime.

3.7 Examples

We conclude this chapter by exhibiting some examples of our methods. We implemented our algorithms in **Sage** [125] using Maple's library **Epsilon** [144] for triangular decompositions, and **Singular** [51] for Gröbner bases. The experiments are performed on an i7 PC with 3.40GHz, 15.6 GB RAM, running Ubuntu 14.04.

3.7.1 Commuting birth and death ideal

We consider the binomial ideal I^{n_1, \dots, n_k} from [58]. This ideal models a generalization of the one-dimensional birth-death Markov process to higher dimensional grids. In [58] it is given a parametrization of its top-dimensional component, as well as the primary decomposition

of some small cases. In [77] Kahle uses his Macaulay2 package **Binomials**, specialized in binomial ideals, to compute primary decompositions of larger examples. We now show how our methods can greatly surpass Kahle’s methods in computing the irreducible decomposition when the treewidth is small.

We focus on the case of a two dimensional grid:

$$I^{n_1, n_2} = \langle U_{i,j}R_{i,j+1} - R_{i,j}U_{i+1,j}, D_{i,j+1}R_{i,j} - R_{i,j+1}D_{i+1,j+1}, \\ D_{i+1,j+1}L_{i+1,j} - L_{i+1,j+1}D_{i,j+1}, U_{i+1,j}L_{i+1,j+1} - L_{i+1,j}U_{i,j} \rangle_{0 \leq i < n_1, 0 \leq j < n_2}.$$

We let the parameter n_1 take values between 1 to 100, while n_2 is either 1 or 2. Table 3.2 shows the time used by Algorithm 1 for different values of n_1, n_2 . Observe that, for small values of n_2 , our methods can handle very high values of n_1 thanks to our use of chordality. For comparison, we note that even for the case $n_1 = 10, n_2 = 1$ **Singular**’s Gröbner basis algorithm (grevlex order) did not terminate within 20 hours of computation. Similarly, neither **Epsilon** [144] nor **RegularChains** [89] were able to compute a triangular decomposition of $I^{10,1}$ within 20 hours.

Table 3.2: Time required by chordal triangularization on ideals I^{n_1, n_2} . No other software we tried [51, 77, 89, 144] can solve these problems.

n_1	20	40	60	80	100
$n_2 = 1$	0:00:45	0:02:16	0:04:03	0:06:28	0:09:13
$n_2 = 2$	0:36:07	1:59:24	3:30:33	6:15:25	9:00:52

We now consider the computation of the irreducible components of the ideal $I^{n_1, 1}$. We follow the strategy described after Theorem 3.20, using **Sage**’s default algorithm to compute saturations. Table 3.3 compares this strategy (including the time of Algorithm 1) with the algorithm from **Binomials** [77]. It can be seen that our method is more efficient. In particular, for the ideal $I^{7,1}$ Kahle’s algorithm did not finish within 60 hours of computation.

Table 3.3: Irreducible components of the ideals $I^{n_1, 1}$.

n_1	1	2	3	4	5	6	7
#components	3	11	40	139	466	1528	4953
time ChordalNet	0:00:00	0:00:01	0:00:04	0:00:13	0:02:01	0:37:35	12:22:19
time Binomials	0:00:00	0:00:00	0:00:01	0:00:12	0:03:00	4:15:36	-

Comparing Table 3.2 with Table 3.3 it is apparent that computing a triangular chordal

network representation is considerably simpler than computing the irreducible components. Nonetheless, if we are only interested in the high dimensional components the complexity can be significantly improved. Indeed, in Table 3.4 we can see how we can very efficiently compute all components of the seven highest dimensions.

Table 3.4: High dimensional irreducible components of the ideals $I^{n_1,1}$.

	Highest 5 dimensions					Highest 7 dimensions			
n_1	20	40	60	80	100	10	20	30	40
#comps	404	684	964	1244	1524	2442	5372	8702	12432
time	0:01:07	0:04:54	0:15:12	0:41:52	1:34:05	0:05:02	0:41:41	3:03:29	9:53:09

3.7.2 Lattice walks

We now show a simple application of our radical membership test. We consider the lattice reachability problem from [53] (see also [83]). Given a set of vectors $\mathcal{B} \subseteq \mathbb{Z}^n$, construct a graph with vertex set \mathbb{N}^n in which $u, v \in \mathbb{N}^n$ are adjacent if $u - v \in \pm\mathcal{B}$. The problem is to decide whether two vectors $s, t \in \mathbb{N}^n$ are in the same connected component of the graph. This problem is equivalent to an ideal membership problem for certain binomial ideal $I_{\mathcal{B}}$ [53]. Therefore, our radical membership test can be used to prove that s, t are not in the same connected component (but it may fail to prove the converse). We consider the following sample problem.

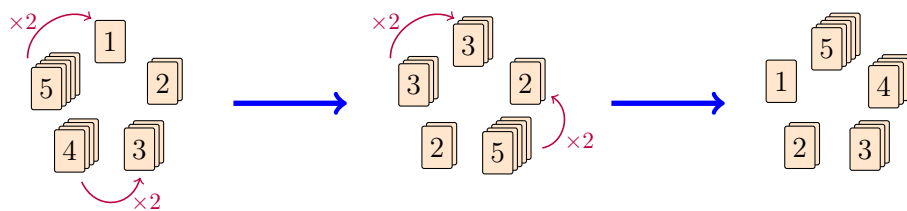


Figure 3-10: Illustration of the card problem using 5 decks.

Problem. *There are n card decks organized on a circle. Given any four consecutive decks we are allowed to move the cards as follows: we may take one card from each of the inner decks and place them in the outer decks (one in each), or we may take one card from the outer decks and place them on the inner decks. Initially the number of cards in the decks are $1, 2, \dots, n$. Is it possible to reach a state where the number of cards in the decks is reversed³ (i.e., the i -th deck has $n - i + 1$ cards)?*

³A combinatorial argument proves that this is only possible if all prime divisors of n are at least 5. However,

The above problem is equivalent to determining whether $f_n \in I_n$, where

$$f_n := x_0 x_1^2 x_2^3 \cdots x_{n-1}^n - x_0^n x_1^{n-1} \cdots x_{n-1}, \quad I_n := \{x_i x_{i+3} - x_{i+1} x_{i+2} : 0 \leq i < n\},$$

and where the indices are taken modulo n . Table 3.5 compares our method against **Singular**'s Gröbner basis (grevlex order) and **Epsilon**'s triangular decomposition. Even though the ideal I_n is not radical, in all experiments performed we obtained the right answer. Note that the complexity of our method is almost linear. This contrasts with the exponential growth of both **Singular** and **Epsilon**, which did not terminate within 20 hours for the cases $n = 30$ and $n = 45$. We do not include timings for **Binomials** and **RegularChains** since they are both slower than **Singular** and **Epsilon**.

Table 3.5: Time (seconds) to test (radical) ideal membership on the ideals I_n .

n	5	10	15	20	25	30	35	40	45	50	55
ChordalNet	0.7	3.0	8.5	14.3	21.8	29.8	37.7	48.2	62.3	70.6	84.8
Singular	0.0	0.0	0.2	17.9	1036.2	-	-	-	-	-	-
Epsilon	0.1	0.2	0.4	2.0	54.4	160.1	5141.9	17510.1	-	-	-
Test result	true	false	false	false	true	false	true	false	false	false	true

3.7.3 Finite state diagram representation

One of the first motivations in this chapter was the very nice chordal network representation of the irreducible components of the ideal of adjacent minors of a $2 \times n$ matrix. We will see now that similar chordal network representations exist for other determinantal ideals.

First, notice that the chordal network in Figure 3-3 has a simple pattern. Indeed, there are three types of nodes $A_i = \{x_{2i}x_{2i+3} - x_{2i+1}x_{2i+2}\}$, $B_i = \{0\}$, $C_i = \{x_{2i}, x_{2i+1}\}$, and we have some valid transitions: $A_i \rightarrow \{A_{i+1}, B_{i+1}\}$, $B_i \rightarrow \{C_{i+1}\}$, $C_i \rightarrow \{A_{i+1}, B_{i+1}\}$. This transition pattern is represented in the state diagram shown in Figure 3-11a. Following the convention from automata theory, we mark the initial states with an incoming arrow and the terminal states with a double line.

We can also consider the ideal of 3×3 adjacent minors of a $3 \times n$ matrix. As seen in Figure 3-11b, a very similar pattern arises. In order to make sense of such diagram let us think of how

this argument does not generalize to other choices of the final state (e.g., we cannot reach a state where the number of cards is 2, 1, 3, 4, 5, \dots , n for any n).

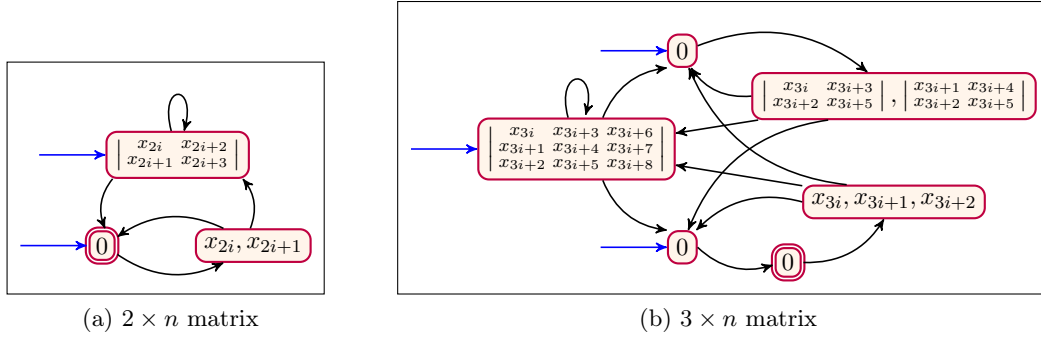


Figure 3-11: State diagrams for ideals of adjacent minors of a matrix.

to generate a $3 \times n$ matrix satisfying all these minor constraints. Let $v_1, \dots, v_n \in \overline{\mathbb{K}}^3$ denote the column vectors. Given v_{i+1}, v_{i+2} we can generate v_i as follows: it can be the zero vector, or it can be a multiple of v_{i+1} , or it can be a linear combination of v_{i+1}, v_{i+2} . These three choices correspond to the three main states shown in the diagram. Note now that if v_i is the zero vector then we can ignore it when we generate v_{i-1} and v_{i-2} . This is why in order to reach the state $(x_{3i}, x_{3i+1}, x_{3i+2})$ we have to pass two trivial states. Similarly, if v_i is parallel to v_{i+1} then we can ignore v_{i+1} when we generate v_{i-1} .

It is easy to see that the above reasoning generalizes if we consider the adjacent minors of a $k \times n$ matrix. Therefore, for any fixed k , the ideal of adjacent minors of a $k \times n$ matrix has a finite state diagram representation (and thus it has a chordal network representation of linear size). Since the nodes of the network are given by minors, then all chains of the network are of the form of Theorem 3.20. Thus, the decomposition obtained is into irreducible components.

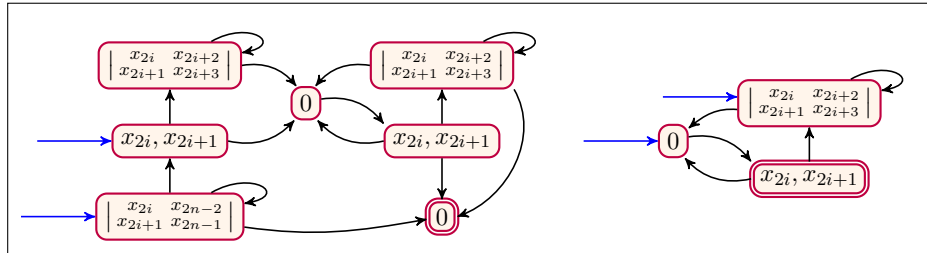


Figure 3-12: State diagram for the ideal of cyclically adjacent minors of a $2 \times n$ matrix.

Many other families of ideals admit a simple state diagram representation. For instance, the ideal generated by the n cyclically adjacent minors of a $2 \times n$ matrix (see Figure 3-12). Interestingly, this chordal network has two equidimensional components. Similarly, the ideal

of (cyclically) adjacent permanent minors has a finite state diagram representation. We can also easily provide families of zero-dimensional problems with such property (e.g., Figure 3-1), since they often admit a chordal network of linear size (Remark 3.6). A similar reasoning applies for monomial ideals. It is natural to ask for further examples of this behaviour.

Question. *Characterize interesting families of ideals (parametrized by n) whose triangular decomposition admits a finite state diagram representation (and thus have a chordal network representation of size $O(n)$).*

Remark 3.15. The class of binomial edge ideals [70] is a natural starting point for this question, given that it generalizes both the ideal of adjacent minors (Figure 3-11a) and cyclically adjacent minors (Figure 3-12) of a $2 \times n$ matrix.

3.8 Additional proofs

3.8.1 Proofs from Section 3.3

Proof of Lemma 3.2. Let $m < n$ be such that X_m is a maximal clique, and consider a rank m node $F_m \subseteq \mathbb{K}[X_m]$ to which we will apply a triangulation operation. Also let $F'_m := F \cap \mathbb{K}[X_m]$ be the unique initial node of rank m . By assumption, F'_m is zero-dimensional. Note that when we create a new node of rank m in an elimination operation, we copy the equations from a previous rank m node. In particular, we must have that $F'_m \subseteq F_m$, and therefore F_m is also zero-dimensional. This proves the lemma for this case.

Consider now some $p < n$ such that X_p is not maximal, which means that x_p is not a leaf of the elimination tree. Since X_p is not maximal, there is a child x_l of x_p such that $X_l = X_p \cup \{x_l\}$. By induction, we may assume that the lemma holds for all nodes of rank l . Consider a rank p node $F_p \subseteq \mathbb{K}[X_p]$ that we want to triangulate, and let F_l of rank l be adjacent to F_p . Let F'_l be the same rank l node, but before the l -th elimination round. By induction, $F'_l \subseteq \mathbb{K}[X_l]$ is zero-dimensional. Therefore, $\text{elim}_p(F'_l) \subseteq \mathbb{K}[X_l \setminus \{x_l\}] = \mathbb{K}[X_p]$ is also zero-dimensional, and as $\text{elim}_p(F'_l) \subseteq F_p$, we conclude that F_p is zero-dimensional. \square

Lemma 3.21. *Let $X_1, X_2 \subseteq X$ and let $I_1 \subseteq \mathbb{K}[X_1]$, $I_2 \subseteq \mathbb{K}[X_2]$ be radical zero-dimensional ideals. Then $I_1 + I_2 \subseteq \mathbb{K}[X_1 \cup X_2]$ is also radical and zero-dimensional.*

Proof. This is a direct consequence of the following known fact (see e.g., [128, Thm 2.2]): an ideal $I \subseteq \mathbb{K}[X]$ is radical and zero-dimensional if and only if for any $x_i \in X$ there is a nonzero squarefree polynomial $f \in I \cap \mathbb{K}[x_i]$. \square

Proof of Proposition 3.6. For any l , let X^l denote the subtree of the elimination tree consisting of x_l and all its descendants. For a chordal network \mathcal{N} , we will say that an l -subchain C_l is the subset of a chain C restricted to nodes with rank i for some $x_i \in X^l$. Note that any chain is also a $(n-1)$ -chain. Thus, it suffices to show that every l -subchain is radical after the l -th triangulation round in Algorithm 1, and we proceed to show it by induction.

If x_l is a leaf in the elimination tree, then any l -subchain is just the output of a triangulation operation and thus it is radical. Assume that the result holds for all $l < p$. Let T_p be a rank p node obtained after the p -th triangulation round. Let C be a p -subchain containing T_p ; we want to show that $\langle C \rangle$ is radical. Let x_{l_1}, \dots, x_{l_k} be the children of x_p . For each l_j , let C_{l_j} be the l_j -subchain obtained by restricting C to ranks in X^{l_j} . Also let C'_{l_j} be the same l_j -subchain, but before the l_j -th elimination round. Observe that

$$\langle C \rangle = \langle T_p \rangle + \sum_j \langle C'_{l_j} \rangle.$$

Note that $\langle T_p \rangle$ is zero-dimensional and radical, and by induction the same holds for each $\langle C'_{l_j} \rangle$. It follows from Lemma 3.21 that $\langle C \rangle$ is radical. \square

Proof of Proposition 3.8. It was shown in [64] that the complexity of Buchberger's algorithm is $q^{O(k)}$ if the equations $x_i^q - x_i$ are present, and the same analysis works for any q -dominated ideal. Given a Gröbner basis, the LexTriangular algorithm [88] computes a triangular decomposition in time $D^{O(1)}$, where $D \leq q^k$ is the number of standard monomials. For irreducible (or squarefree) decompositions, we can reduce the problem to the univariate case by using a rational univariate representation [113] (here we need that \mathbb{K} contains sufficiently many elements). This representation can also be obtained in $D^{O(1)}$. Since the complexity of univariate (squarefree) factorization [79] is polynomial in the degree (D), the result follows. \square

Proof of Lemma 3.10. Let us see that the result holds after each triangulation and elimination round. We showed in Lemma 3.7 that after the l -th triangulation round all rank l nodes have

disjoint varieties, and thus there are at most $|\mathbf{V}(F \cap \mathbb{K}[X_l])| \leq q^\omega$ of them. Consider now the l -th elimination round, and let us see that all the resulting rank p nodes (x_p parent of x_l) also have disjoint varieties, and thus the same bound holds.

Assume by induction that all rank p nodes have disjoint varieties before the l -th elimination round. Let F_p be a rank p node (before the elimination) and let T_1, \dots, T_k be its adjacent rank l nodes. We just need to show that the new rank p nodes $F_p \cup \text{elim}_p(T_1), \dots, F_p \cup \text{elim}_p(T_k)$ have disjoint varieties (or are the same). By assumption, each $T_i \subseteq \mathbb{K}[X_l]$ defines a maximal (or prime) ideal, and thus $\text{elim}_p(T_i) \subseteq \mathbb{K}[X_l \setminus \{x_l\}]$ also defines a maximal ideal. Therefore, $\mathbf{V}(\text{elim}_p(T_i)), \mathbf{V}(\text{elim}_p(T_j))$ are either equal or disjoint, and it follows that the same holds for $\mathbf{V}(F_p \cup \text{elim}_p(T_i)), \mathbf{V}(F_p \cup \text{elim}_p(T_j))$. \square

3.8.2 Proofs from Section 3.4

Lemma 3.22. *Let \mathbb{L} be a ring and let $f \in \mathbb{L}[y]$ be a monic univariate polynomial. Let $\phi : \mathbb{L}[y] \rightarrow \mathbb{L}[y]$ be an endomorphism such that $\phi(f) = f$ and $\deg(\phi(h)) \leq \deg(h)$ for any $h \in \mathbb{L}[y]$. Then $\phi(h \bmod f) = \phi(h) \bmod f$, for any $h \in \mathbb{L}[y]$.*

Proof. Consider the Euclidean division $h = qf + r$, where $q, r \in \mathbb{L}[y]$ and $\deg(r) < \deg(f)$. Then $\phi(h) = \phi(q)f + \phi(r)$ and $\deg(\phi(r)) \leq \deg(r) < \deg(f)$, so this is the Euclidean division of $\phi(h)$. It follows that $\phi(h \bmod f) = \phi(r) = \phi(h) \bmod f$. \square

Proof of Lemma 3.13. We proceed by induction on l . The base case, $l = 0$, is clear. Assume now that the lemma holds for some l , and let us prove it for $p := l + 1$. Let f_p be a rank p node and let $f_{l,1}, f_{l,2}, \dots, f_{l,k}$ be its adjacent rank l nodes. Let us denote as ϕ_l the functional that plugs in the values $\hat{x}_0, \dots, \hat{x}_l$. By induction, we know that

$$H(f_{l,i}) = \phi_l\left(\sum_{C_{l,i}} r_{C_{l,i}} h \bmod C_{l,i}\right)$$

where the sum is over all $f_{l,i}$ -subchains $C_{l,i}$. Note that the algorithm sets

$$H(f_p) = \phi_p\left(\sum_i r_{l,i} H(f_{l,i}) \bmod f_p\right),$$

where ϕ_p is the functional that plugs in the value \hat{x}_p . Therefore,

$$H(f_p) = \phi_p\left(\sum_i r_{l,i} \phi_l\left(\sum_{C_{l,i}} r_{C_{l,i}} h \bmod C_{l,i}\right) \bmod f_p\right) = \phi_p\left(\phi_l\left(\sum_i \sum_{C_{l,i}} r_{l,i} r_{C_{l,i}} h \bmod C_{l,i}\right) \bmod f_p\right).$$

Since any f_p -subchain is of the form $C_p = C_{l,i} \cup \{f_p\}$ for some i , we can rewrite

$$H(f_p) = \phi_p\left(\phi_l\left(\sum_{C_p} r_{C_p} h \bmod C'_p\right) \bmod f_p\right),$$

where the sum is over all f_p -subchains C_p , and where $C'_p := C_p \setminus \{f_p\}$. To complete the proof we just need to see that ϕ_l commutes with $\bmod f_p$. This follows from Lemma 3.22 by setting $y = x_p$ and $\mathbb{L} = \mathbb{K}[X \setminus \{x_p\}]$. \square

Proof of Lemma 3.15. Let x_{m_i} denote the main variable of h_i , which is one of the ranks where the algorithm is initialized. It is enough to prove the lemma for ranks l where the paths (in the elimination tree) starting from different x_{m_i} first meet. Thus, we restrict ourselves to some m_1, \dots, m_k such that their respective paths all meet at rank l . More precisely, we assume that $X_{m_i}^l \cap X_{m_j}^l = \{x_l\}$ for $i \neq j$, where $X_{m_i}^l$ denotes the path in the elimination tree connecting x_{m_i} to x_l .

By applying Lemma 3.13 to each h_i , it follows that the final value of $H(f_l)$ is given by plugging in the values of $\hat{x}_1, \hat{x}_2, \dots, \hat{x}_l$ in the polynomial

$$\sum_i \sum_{C_i} r_{C_i} h_i \bmod C_i,$$

where C_i is an f_l -subchain restricted to the path $X_{m_i}^l$. Let $C = \bigcup_i C_i$ be the f_l -subchain obtained by combining them. We want to show that the above expression is equal to

$$\sum_C r_C (h_1 + \dots + h_k) \bmod C.$$

Note now that h_i does not involve any variable in $X_{m_j}^l$ for $j \neq i$. Thus, $h_i \bmod C = h_i \bmod C_i$. Observe that $X_{m_i}^l, X_{m_j}^l$ have no common arcs since they only meet at level l , and thus $r_C =$

$\prod_i r_{C_i}$. Denoting $h_{C_i} := h_i \bmod C_i$, the problem reduces to proving the following equality:

$$\sum_i \sum_{C_i} r_{C_i} h_{C_i} = \sum_C r_{C_1} r_{C_2} \cdots r_{C_k} (h_{C_1} + h_{C_2} + \cdots + h_{C_k}). \quad (3.2)$$

In order to prove eq. (3.2), let us look at the right hand side as a polynomial in variables h_{C_1}, \dots, h_{C_k} . Note that the coefficient of h_{C_1} in such polynomial is

$$\sum_{C \supseteq C_1} r_{C_1} r_{C_2} \cdots r_{C_k} = r_{C_1} \prod_{i=2}^k \left(\sum_{C_i} r_{C_i} \right)$$

and we want to show that this expression reduces to r_{C_1} . Recall that the scalar coefficients r_{C_i} are normalized (this was the second modification made to Algorithm 3). It follows that $\sum_{C_i} r_{C_i} = 1$ for all i , and thus eq. (3.2) holds. \square

3.8.3 Proofs from Section 3.6

Proof of Theorem 3.19. We have to show that: chordality is preserved, the variety is preserved, and the chains in the output are regular systems. The proofs of first two statements are essentially the same as for the chordally zero-dimensional case (Lemma 3.3 and Lemma 3.5). It only remains to show that the chains of the output are regular systems. Proving that the chains are squarefree is very similar, so we skip it.

Let X^l denote the subtree of the elimination tree consisting of x_l and its descendants. We say that an l -subchain is the subset of a chain given by nodes of rank i for some $x_i \in X^l$. We will show by induction on l that after the l -th triangulation round every l -subchain is a regular system. The base case is clear. Assume that the result holds for all $l < p$. Let \mathfrak{T}_p be a rank p node obtained after the p -th triangulation round. Let \mathfrak{C} be a p -subchain containing \mathfrak{T}_p ; we want to show that it is a regular system. It is easy to see that \mathfrak{C} is triangular and that condition (i) from Definition 3.12 is satisfied. We just need to check condition (ii).

Let $f \in \mathfrak{C}$ be a rank k polynomial; we want to show that $\text{init}(f)(\hat{x}^{k+1}) \neq 0$ for any $\hat{x}^{k+1} \in \mathbf{Z}(\text{elim}_{k+1}(\mathfrak{C}))$. First consider the case that $k \geq p$, which means that $f \in \mathfrak{T}_p$. The result follows from the fact that \mathfrak{T}_p is a regular system. Assume now that $k < p$, in which case there must be a child x_l of x_p such that $x_k \in X^l$. This means that f belongs to an l -subchain \mathfrak{C}_l , which is a subset of \mathfrak{C} . Let \mathfrak{C}'_l be the same l -subchain, but before the l -th elimination round.

By induction, we know that \mathfrak{C}'_l is a regular system, and thus $\text{init}(f)(\hat{x}^{k+1}) \neq 0$ for any $\hat{x}^{k+1} \in \mathbf{Z}(\text{elim}_{k+1}(\mathfrak{C}'_l))$. The result follows by noticing that $\mathbf{Z}(\text{elim}_{k+1}(\mathfrak{C})) \subseteq \mathbf{Z}(\text{elim}_{k+1}(\mathfrak{C}'_l))$. \square

Chapter 4

Graphical structure in permanents and related problems

This chapter presents an efficient algorithm to compute permanents of matrices with structured sparsity. We also study some higher dimensional generalizations, such as mixed discriminants, hyperdeterminants, and mixed volumes. The content of this chapter is based on [38].

4.1 Introduction

The *permanent* of a $n \times n$ matrix M is defined as

$$\text{Perm}(M) := \sum_{\pi} \prod_{i=1}^n M_{i,\pi(i)}$$

where the sum is over all permutations π of the numbers $1, \dots, n$. Computing the permanent is #P-hard [134], which means that it is unlikely that it can be done efficiently for arbitrary matrices. As a consequence, research on this problem tends to fall into two categories: algorithms to approximate the permanent, and exact algorithms that assume some structure of the matrix. The contributions of this chapter lie in the second category. We further study related problems in structured higher dimensional arrays, such as mixed discriminants, hyperdeterminants and mixed volumes.

The sparsity pattern of a matrix M can be seen as the bipartite adjacency matrix of some bipartite graph G . This bipartite graph fully encodes the structure of the matrix. We assume

here that the treewidth ω of G is small (constant or logarithmic in n). We show an algorithm to compute $\text{Perm}(M)$ in $\tilde{O}(n 2^\omega)$ arithmetic operations. In this thesis, the notation \tilde{O} ignores polynomial factors in ω . We note that the algorithm can be used over any commutative ring.

The permanent of a matrix can be generalized in several ways. In particular, given a list of n matrices of size $n \times n$, its *mixed discriminant* generalizes both the permanent and the determinant [8, 68]. Our algorithm for the permanent extends in a natural way to compute the mixed discriminant. The natural structure to represent the sparsity pattern in this case is a tripartite (i.e., 3-colorable) graph. The running time of the resulting algorithm is $\tilde{O}(n^2 + n 3^\omega)$, where ω is the treewidth of such graph. More generally, our methods extend to generalized determinants/permanents on tensors. A special case of interest is the *multidimensional permanent* [6, 55, 132]. Another interesting case is the first Cayley *hyperdeterminant*, also known as Pascal determinant, which is the simplest generalization of the determinant to higher dimensions [9, 32, 94]. Note that unlike the determinant, the hyperdeterminant is $\#P$ -hard, in particular because it contains mixed discriminants as a special case [68, 72].

Given a set of n polytopes in \mathbb{R}^n , its mixed volume provides a geometric generalization of the permanent and the determinant [120]. We focus on the special case of the mixed volume of n zonotopes. Although there is no “natural” graph to represent the structure of a set of zonotopes, we associate to it a bipartite graph that, when the mixed volume restricts to a permanent, corresponds to the matrix graph described above. This allows us to give a simple application for mixed volumes of zonotopes with few nonparallel edges. Nevertheless, we show that mixed volumes remain hard to compute when the treewidth is bounded. The close connection between sparse polynomial systems and lattice polytopes allows us to conclude that solving generic systems of bounded treewidth is also computationally hard.

The diagram of Figure 4-1 summarizes the scope of the chapter. It presents the main problems we consider, illustrating the relationships among them. Concretely, an arrow from A to B indicates that B is a special instance of A . It also divides the problems according to its difficulty, with and without bounded treewidth assumptions. In this chapter we start from the simplest problems, i.e., permanents and determinants of matrices, moving upwards in the diagram.

The document is structured as follows. In Section 4.2 we present three graph abstractions of a sparse matrix. Among these graphs is the bipartite graph G described above, and a

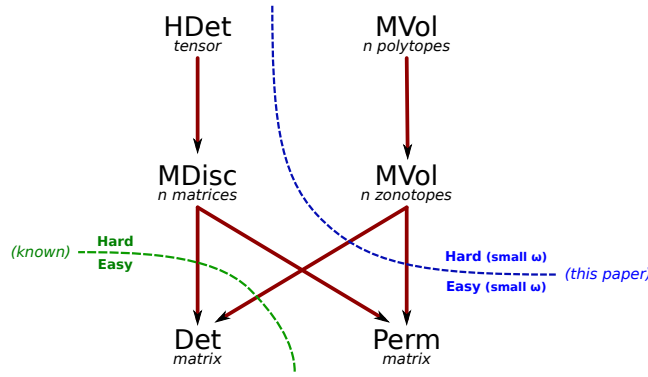


Figure 4-1: Diagram describing the complexity relations of computing: determinants, permanents, mixed discriminants, hyperdeterminants and mixed volumes.

projection G^X onto the column set. In Section 4.3 we present a decomposition method, Algorithm 4, that computes the permanent based on the graph G^X . We first use graph G^X because the decomposition algorithm is easier to explain in this case. In Section 4.4 we extend this method to work with the bipartite graph G , as shown in Algorithm 5. We provide a Matlab implementation of this algorithm. In Section 4.5 we discuss the case of mixed discriminants, presenting a decomposition method for it. We also treat the case of generalized determinants/permanents on tensors. Finally, in Section 4.6 we discuss the case of mixed volumes of zonotopes.

Related work

Permanents

The best known to date method for exactly computing the permanent of a general matrix was given by Ryser and its complexity is $O(n 2^n)$ [114]. There are two main research trends on permanent computation: approximation algorithms and exact algorithms for structured matrices. We briefly discuss related work in both of them. Tree decomposition algorithms, which belong to the second group, will be presented afterwards.

We first mention some work on approximation schemes. For arbitrary matrices, Gurvits gave a randomized polynomial time approximation, with error proportional to the operator norm raised to the power n [68]. For nonnegative matrices there is a vast literature, see e.g., [141] and the references therein. Most remarkably, Jerrum et al. gave a fully polynomial randomized approximation scheme (FPRAS) [75]. Recent work studies approximation schemes

based on belief propagation, also for nonnegative matrices [141, 147].

As for exact algorithms for structured matrices, different types of structure have been explored in the literature. Fisher, Kasteleyn and Temperley gave a polynomial time algorithm for matrices whose associated bipartite graph is planar [80, 130]. Barvinok showed that the permanent is tractable when the rank is bounded [10]. The case of 0/1 circulant matrices has also been considered [98], as well as sparse 0/1 Toeplitz matrices [44]. Schwartz showed a $O(\log(n)2^{6w})$ algorithm for certain band, Toeplitz matrices, where w is the bandwidth [121]. Temme and Wocjan showed a $O(n2^{3w^2})$ algorithm for a special type of band matrices [129]. Note that for arbitrary band matrices our algorithm is $\tilde{O}(n2^{2w})$.

Permanents and treewidth

Tree decomposition methods for permanent computation have been considered. Courcelle et al. first showed that the permanent can be computed efficiently if the treewidth is bounded [45], although their methods, based on the Feferman-Vaught-Shelah Theorem, do not lead to an implementable algorithm. Later work of Flarup et al. gives a $O(n2^{O(\omega^2)})$ algorithm [62]. This algorithm is extended in [96] to a wider class of matrices. Note the strong dependency on the treewidth. Furthermore, the graph abstraction used in the above methods, which is not the bipartite graph G , has two inconvenient features: its treewidth can be significantly larger than the one of G (see Example 4.1) and it is dependent on the specific order of the columns of the matrix (see Remark 4.2).

Closer to this chapter is the work of van Rooij et al. [135]. They gave a $\tilde{O}(n2^\omega)$ decomposition algorithm for counting perfect matchings in a graph. Counting perfect matchings is closely related to the permanent, and one could derive from their proof an analogous, but different, method for calculating the permanent. Our algorithm could be seen as a variant of such method that is easier to extend to the higher dimensional problems we consider.

Mixed discriminants, mixed volumes, tensors

The higher dimensional problems we study generalize the permanent of a matrix, and thus are #P-hard in general. As for the permanent, there are two natural relaxations: approximation algorithms and exact algorithms under special structure. Approximation algorithms have

been considered for mixed discriminants [11, 68], mixed volumes [11, 56] and multidimensional permanents [12]. As for exact algorithms under special structure, we are only aware of Gurvits' tractability result for mixed discriminants and the 4-hyperdeterminant under some bounded rank assumptions [68].

To our knowledge this is the first work that studies tree decomposition methods for mixed discriminants, mixed volumes and generalized determinants/permanents on tensors. Related to this is a recent log-space algorithm for computing determinants under bounded treewidth assumptions [7]. Also related is the problem of partitioning a low treewidth graph into k -cliques, which is considered in [135].

4.2 Graph representations of a sparse matrix

The sparsity structure of a matrix, i.e., its pattern of nonzero entries, can be described in terms of a graph. We consider here three possible graph abstractions of such sparsity structure, and we compare their treewidths. We will use the characterization of treewidth in terms of tree decompositions (see Definition 2.6).

Let M be a $n \times n$ matrix. We will index the rows with a set $A = \{a_1, \dots, a_n\}$ and the columns with a set $X = \{x_1, \dots, x_n\}$. We use subindices to index the coordinates of M , i.e., $M_{a,x}$ denotes the entry in the a -th row and x -th column. Similarly, let M_a be the a -th row of M . We now present two (undirected) graphs that are usually associated to a sparse matrix.

Definition 4.1 (Bipartite graph). Let M be a $n \times n$ matrix, let A denote its set of rows, and let X denote its set of columns. The *bipartite graph* of M , denoted as $G(M)$, has vertices $A \cup X$, and there is an edge (a, x) if $M_{a,x}$ is nonzero.

Definition 4.2. (Symmetrized graph) Let M be a $n \times n$ matrix, let A denote its set of rows, and let X denote its set of columns. The *symmetrized graph* of M , denoted as $G^s(M)$, has vertices $1, \dots, n$ and has an edge (i, j) if $M_{a_i, x_j} \neq 0$ or $M_{a_j, x_i} \neq 0$.

Remark 4.1. Note that $G^s(M)$ is the adjacency graph of the symmetric matrix $M + M^T$, assuming no terms cancel out.

Remark 4.2 (Permutation invariance). Note that the permanent of a matrix is invariant under independent row and column permutations. The bipartite graph G preserves this invariance.

On the other hand, the symmetrized graph G^s is only invariant under simultaneous row and column permutations.

The bipartite graph G is our main object of study in this chapter. Let $\omega := \text{tw}(G)$ be the treewidth of G and $\omega_s := \text{tw}(G^s)$ the one of G^s . Tree decomposition methods based on graph G^s have been studied before [7, 45, 62, 96]. We claim that graph G is a better abstraction for the purpose of permanent computation. In particular, G preserves the permutation invariance of the permanent as stated above. Furthermore, ω_s can be much larger than ω as shown in the following example.

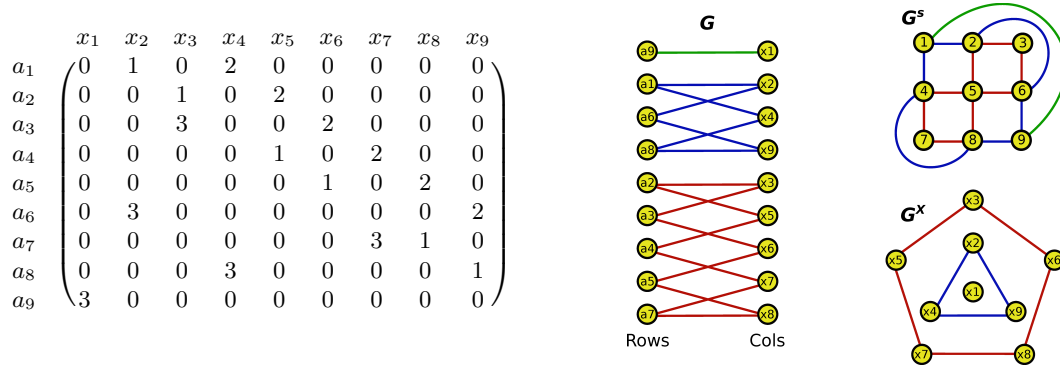


Figure 4-2: Graph abstractions of a matrix: bipartite graph G , symmetrized graph G^s and column graph G^X .

Example 4.1 (Two nonzero entries per row). Let M be a matrix with at most two nonzero entries per row. We claim that for all nontrivial cases the bipartite graph G has treewidth $\omega \leq 2$. Let G_0 be a connected component, and let n_0 be its number of row vertices. In order for G_0 to have a perfect matching, it must have as many row vertices as column vertices. Note also that G_0 has at most $2n_0$ edges because the row degrees are at most 2. Then G_0 is a connected graph with $2n_0$ vertices and at most $2n_0$ edges, so it has at most one cycle. It follows that $\omega \leq 2$.

On the other hand, we will see that ω_s is unbounded. Let $n = m^2$ and consider the matrix M whose nonzero entries are

$$\begin{aligned}
 M_{a_i, x_{i+1}} &= 1, \text{ if } i/m \notin \mathbb{Z}, & M_{a_{(m-i+1)m}, x_i} &= 3, \text{ if } i \leq m, \\
 M_{a_i, x_{i+m}} &= 2, \text{ if } i \leq n - m, & M_{a_{n-i}, x_{im+1}} &= 3, \text{ if } i < m.
 \end{aligned}$$

The graph G^s contains the grid graph, and thus $\omega_s \geq \sqrt{n}$. The case $n = 9$ is shown in Figure 4-2.

The following example shows that the treewidth of G is always better than the treewidth of G^s .

Example 4.2 (G is “better” than G^s). Let’s see that given a tree decomposition of G^s of width ω_s we can form a tree decomposition of G of width $2\omega_s$. Let (T, ι) be a tree decomposition of G^s , where $\iota(t) \subseteq \{1, \dots, n\}$. Let $\mu : T \rightarrow \{0, 1\}^{A \cup X}$, be such that $\mu(t) = \{a_i : i \in \iota(t)\} \cup \{x_i : i \in \iota(t)\}$. Then (T, μ) is a decomposition of G of width $2\omega_s$. On the contrary, for a fixed ω the treewidth of G^s is unbounded as seen in Example 4.1.

We now introduce a third graph G^X that we can associate to matrix M .

Definition 4.3. (Column graph) Let M be a $n \times n$ matrix, let A denote its set of rows and let X denote its set of columns. For any $a \in A$ let $X(a)$ denote the set of nonzero components of row M_a . The *column graph* $G^X(M)$ has X as its vertex set, and for each $a \in A$ we put a clique in $X(a)$. Equivalently, there is an edge (x_i, x_j) if there is some $a \in A$ such that $x_i, x_j \in X(a)$.

Graph G^X can be seen as a projection of G onto the column set X . The reason why we consider this graph is that we can give a very simple algorithm for the permanent based on it. We present this algorithm in Section 4.3, and then extend it to G in Section 4.4. We now show that $\omega \leq \omega_X + 1$, where $\omega_X := \text{tw}(G^X)$.

Example 4.3 (G is “better” than G^X). Let (T, χ) be a tree decomposition of G^X of width ω_X . For each row $a \in A$ we associate to it a unique node $t \in T$ such that $X(a) \subseteq \chi(t)$. This assignment can be made because of Lemma 2.4. For some $t \in T$, let $a_1^t, a_2^t, \dots, a_k^t$ be all rows that are assigned to t . Let’s replace node t of T with a path t_1, t_2, \dots, t_k , and let $\mu(t_j) = \chi(t) \cup \{a_j^t\}$ for $j = 1, \dots, k$. The nodes previously connected to t can be linked to any of the new nodes. By doing this for every $t \in T$, we obtain a tree decomposition (T, μ) of G of width $\omega_X + 1$.

On the other hand, for a fixed ω the treewidth of G^X is unbounded. For instance, consider

the matrix M whose only nonzero entries are: $M_{a_i, x_i} = 1$ (diagonal) and $M_{a_1, x_i} = 1$ (first row) for all i . In this case G^X is the complete graph ($\omega_X = n - 1$) and G is a tree ($\omega = 1$).

Given a matrix M (or a family of matrices) we can use any of the three graph abstractions presented to represent its sparsity structure. If any of these graphs has small treewidth then the methods of this chapter allow to compute its permanent efficiently. In particular, this is the case for matrices with band (or cyclic band) structure, as explained now.

Example 4.4 (Band structure). A matrix M is banded if $M_{a_i, x_j} = 0$ whenever $i - j > w_1$ or $j - i > w_2$, for some parameters $w_1, w_2 \in \mathbb{Z}_{>0}$. In the associated column graph G^X each vertex x_i is connected to the following $k := w_1 + w_2$ vertices x_{i+1}, \dots, x_{i+k} . This is a chordal graph known as the k -path and its treewidth is $\omega_X = k$ (the size of the largest cliques minus one). It follows that for the corresponding bipartite graph $\omega \leq w_1 + w_2 + 1$.

Similarly, we say that a matrix M is cyclically banded if $M_{a_i, x_j} = 0$ whenever $n - w_2 > i - j > w_1$ or $n - w_1 > j - i > w_2$. In this case, the column graph G^X is a circulant graph in which every vertex is connected to the following/previous $k := w_1 + w_2$ vertices in a cyclic way. It can be seen that the treewidth of such graph is $\omega_X = 2k$. Therefore, $\omega \leq 2w_1 + 2w_2 + 1$.

To conclude, we point out that treewidth is a natural measure of complexity to consider, which is well understood both in theory and practice. In particular, for an arbitrary graph G (corresponding to some matrix) and for a fixed k we can test in linear time whether $\omega \leq k$, and if so find a tree decomposition [24]. In addition, many interesting families of graphs have small treewidth [25], such as: series-parallel graphs, outerplanar graphs, Halin graphs, Apollonian networks, and certain recursive families of graphs. By interpreting any of such graphs as a matrix, using any of the three graph abstractions, we obtain a family of matrices with low treewidth.

Example 4.5 (Halin structure). Consider a planar graph that consists of a tree together with a cycle around some of its leaves. This is a Halin-like graph, and its treewidth can be shown to be 3 in the same way as in [25, Theorem 85]. Let's interpret such graph as the bipartite graph of some matrix. As seen in Figure 4-3, this matrix has a simple structure: by connecting the nonzero entries that share a row or a column, the matrix decomposes into a tree and a cycle, such that any entry in the cycle is in the same row/column of exactly one entry of the tree.

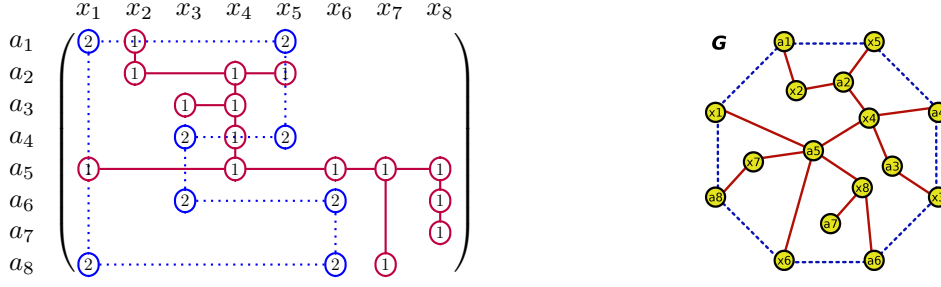


Figure 4-3: Structured matrix (zeros are not shown) and its bipartite graph.

4.3 Column decompositions

Notation. $Y = Y_1 \sqcup Y_2$ denotes a set partition, i.e., $Y = Y_1 \cup Y_2$ and $Y_1 \cap Y_2 = \emptyset$.

In this section, we develop an algorithm to compute the permanent based on a tree decomposition of the column graph G^X (see Definition 4.3). For this section only, we denote the treewidth of G^X by ω . We will show that we can compute $\text{Perm}(M)$ in $\tilde{O}(n 4^\omega)$. Recall that the notation \tilde{O} ignores polynomial factors in ω .

As before, A denotes the row set and X the column set. We use subindices to index the coordinates of M , i.e., $M_{a,x}$ denotes the element in row a and column x . The following example illustrates the methodology we follow.

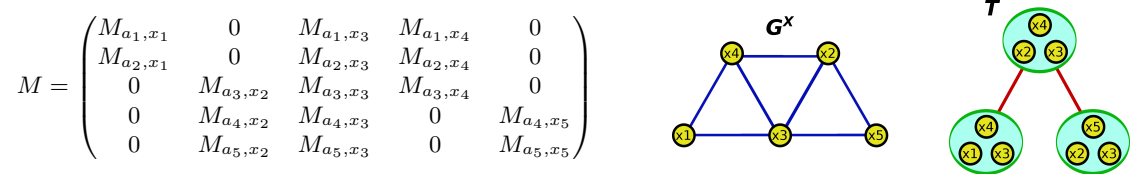


Figure 4-4: Matrix M , its column graph G^X and a tree decomposition T .

Example 4.6. Consider the 5×5 matrix M of Figure 4-4 and the following partition of its rows:

$$A = \{a_1, a_2\} \sqcup \{a_3\} \sqcup \{a_4, a_5\}.$$

There is a simple expansion of $\text{Perm}(M)$ in terms of this partition:

$$\begin{aligned} \text{Perm}(M) = & \text{Perm} \begin{pmatrix} M_{a_1,x_1} & M_{a_1,x_3} \\ M_{a_2,x_1} & M_{a_2,x_3} \end{pmatrix} \text{Perm} \left(M_{a_3,x_4} \right) \text{Perm} \begin{pmatrix} M_{a_4,x_2} & M_{a_4,x_5} \\ M_{a_5,x_2} & M_{a_5,x_5} \end{pmatrix} \\ & + \text{Perm} \begin{pmatrix} M_{a_1,x_1} & M_{a_1,x_4} \\ M_{a_2,x_1} & M_{a_2,x_4} \end{pmatrix} \text{Perm} \left(M_{a_3,x_3} \right) \text{Perm} \begin{pmatrix} M_{a_4,x_2} & M_{a_4,x_5} \\ M_{a_5,x_2} & M_{a_5,x_5} \end{pmatrix} \\ & + \text{Perm} \begin{pmatrix} M_{a_1,x_1} & M_{a_1,x_4} \\ M_{a_2,x_1} & M_{a_2,x_4} \end{pmatrix} \text{Perm} \left(M_{a_3,x_2} \right) \text{Perm} \begin{pmatrix} M_{a_4,x_3} & M_{a_4,x_5} \\ M_{a_5,x_3} & M_{a_5,x_5} \end{pmatrix}. \end{aligned}$$

This expansion implies that to compute $\text{Perm}(M)$ we just need to evaluate two 2×2 permanents corresponding to $\{a_1, a_2\}$, three 1×1 permanents corresponding to $\{a_3\}$, and two 2×2 permanents corresponding to $\{a_4, a_5\}$. This requires only 14 multiplications, compared to $4 \times 5! = 480$ multiplications using the definition. The reason why this formula exists is because the column graph G^X of matrix M has a simple tree decomposition, which is shown in Figure 4-4.

As in the example above, we can always obtain an expansion of $\text{Perm}(M)$ using a tree decomposition of graph G^X . By carefully evaluating this formula we will obtain a dynamic programming method to compute $\text{Perm}(M)$.

4.3.1 Partial permanent

In our notation, the permanent of M can be expressed as

$$\text{Perm}(M) = \sum_{\pi} \prod_{a \in A} M_{a,\pi(a)}$$

where the sum is over all bijections $\pi : A \rightarrow X$. For a given row a , let $X(a)$ denote the column coordinates where it is nonzero. We will refer to a bijection π as a *matching* if $\pi(a) \in X(a)$, i.e., $M_{a,\pi(a)} \neq 0$, for all $a \in A$. Then we can restrict the above sum to be over all matchings.

We consider a tree decomposition (T, χ) of the column graph G^X . Note that by construction of G^X then $X(a)$ is a clique for any $a \in A$. Thus, Lemma 2.4 says that we can assign each row $a \in A$ to some node t , such that $X(a) \subseteq \chi(t)$. From now, we *fix an assignment* of each a to a unique node. Let A_t denote the rows that are assigned to node t . Thus, we have

the partition $A = \bigsqcup_{t \in T} A_t$.

Let T_t be the subtree of T rooted in a node t . Let $\chi(T_t)$ be the union of $\chi(t')$ over all $t' \in T_t$, and similarly let A_{T_t} be the union of $A_{t'}$ over all $t' \in T_t$. For instance, for the root node we have $A_{T_{root}} = A$ and $\chi(T_{root}) = X$.

For a fixed matrix M and some sets $D \subseteq A$ and $Y \subseteq X$ we denote

$$perm(D, Y) := \sum_{\pi} \prod_{a \in D} M_{a, \pi(a)} \quad (4.1)$$

where the sum is over all matchings $\pi : D \rightarrow Y$. Equivalently, it is the permanent of the submatrix of M corresponding to such rows and columns. Clearly, this only makes sense if $|D| = |Y|$, and otherwise we can define $perm(D, Y) = 0$. We refer to $perm(D, Y)$ as a *partial permanent*.

4.3.2 Decomposition algorithm for the permanent

Algorithm 4 presents our dynamic programming method to compute $\text{Perm}(M)$. We will explain and derive this algorithm in the following sections. For each node t of the tree, the algorithm computes a table P_t indexed by subsets \bar{Y} of $\chi(t)$. It starts from the leaves of the tree and recursively computes the tables of all nodes following a topological ordering. The permanent of M is found in the table corresponding to the root.

Algorithm 4 has two main routines:

- For any node t , SUBPERMS computes a table Q_t with the permanents of all submatrices corresponding to t . (See Section 4.3.3).
- EVALRECURSION computes table P_t of an internal node t , by combining table Q_t with the tables P_{c_1}, \dots, P_{c_k} of the node's children. (See Section 4.3.4).

The values $P_t(\bar{Y})$ that we compute correspond to a partial permanent of the matrix, as we explain now. Consider the collection

$$S := \{Y : \chi(T_t) \setminus \chi(t) \subseteq Y \subseteq \chi(T_t)\}.$$

Observe that $Y \in S$ is completely determined by $Y \cap \chi(t)$. Therefore, if we let $\bar{Y} := Y \cap \chi(t)$,

Algorithm 4 Permanent with column decomposition

Input: Matrix M and tree decomposition (T, χ) of column graph $G^X(M)$

Output: Permanent of M

```

1: procedure COLSPERM( $M, T, \chi$ )
2:   assign each  $a \in A$  to some  $t$  with  $X(a) \subseteq \chi(t)$ 
3:    $order :=$  topological ordering of  $T$  starting from its leaves
4:   for  $t$  in  $order$  do
5:      $Q_t :=$  SUBPERMS( $t, M$ )
6:     if  $t$  is a leaf then
7:        $P_t := Q_t$ 
8:     else
9:        $c_1, \dots, c_k :=$  children of  $t$ 
10:       $P_t :=$  EVALRECURSION( $t, Q_t, P_{c_1}, \dots, P_{c_k}$ )
11:   return  $P_{root}(\chi(root))$ 
12: procedure SUBPERMS( $t, M$ )
13:    $A_t :=$  rows assigned to node  $t$ 
14:    $Q_t(\bar{Y}) := perm(A_t, \bar{Y})$  for all  $\bar{Y} \subseteq \chi(t)$ 
15: procedure EVALRECURSION( $t, Q_t, P_{c_1}, \dots, P_{c_k}$ )
16:   for  $c_j$  child of  $t$  do
17:      $\Delta_j := \chi(c_j) \setminus \chi(t)$ ,  $\Lambda_j := \chi(c_j) \cap \chi(t)$ 
18:      $Q_{c_j}(\bar{Y}) := P_{c_j}(\bar{Y} \cup \Delta_j)$  for all  $\bar{Y} \subseteq \Lambda_j$ 
19:    $P_t :=$  SUBSETCONVOLUTION( $Q_t, Q_{c_1}, \dots, Q_{c_k}$ )
20: procedure SUBSETCONVOLUTION( $P_0, P_1, \dots, P_k$ )
21:    $P(\bar{Y}) := \sum_{\bar{Y}_0 \sqcup \dots \sqcup \bar{Y}_k = \bar{Y}} P_0(\bar{Y}_0) P_1(\bar{Y}_1) \dots P_k(\bar{Y}_k)$ 

```

there is a one to one correspondence between S , and the collection $\bar{S} := \{\bar{Y} : \bar{Y} \subseteq \chi(t)\}$. Then the partial permanents that we are interested in are

$$P_t(\bar{Y}) := \text{perm}(A_{T_t}, Y) = \text{perm}(A_{T_t}, \bar{Y} \cup (\chi(T_t) \setminus \chi(t))).$$

The reason why we index table P_t with \bar{Y} instead of Y , is that in this way it becomes clearer that the recursion formula is actually a subset convolution.

Observe that the permanent of M is indeed computed in Algorithm 4, as for the root node we have $P_{\text{root}}(\chi(\text{root})) = \text{perm}(A, X) = \text{Perm}(M)$. Also note that for a leaf node we have $P_t(\bar{Y}) = \text{perm}(A_t, \bar{Y}) = Q_t(\bar{Y})$.

Example 4.7. Consider the matrix M and tree decomposition T of Figure 4-4. Let t_1, t_2, t_3 be the nodes of T , where the central node t_2 is the root. We show the tables computed by Algorithm 4. The tables Q_t with the permanents of all submatrices are:

$$\begin{aligned} Q_{t_1}(\{x, y\}) &= \text{perm}(\{a_1, a_2\}, \{x, y\}), & \text{for } x, y \in \chi(t_1) = \{x_1, x_3, x_4\} \\ Q_{t_3}(\{x, y\}) &= \text{perm}(\{a_4, a_5\}, \{x, y\}), & \text{for } x, y \in \chi(t_3) = \{x_2, x_3, x_5\} \\ Q_{t_2}(\{x\}) &= \text{perm}(\{a_3\}, \{x\}), & \text{for } x \in \chi(t_2) = \{x_2, x_3, x_4\} \end{aligned}$$

We now show the final tables P_t for each node. For the leaves t_1, t_3 we have $P_{t_1} = Q_{t_1}$, $P_{t_3} = Q_{t_3}$. As for the root t_2 , the recursion is:

$$\begin{aligned} P_{t_2}(\{x_2, x_3, x_4\}) &= Q_{t_2}(\{x_4\})P_{t_1}(\{x_1, x_3\})P_{t_3}(\{x_2, x_5\}) + \\ &Q_{t_2}(\{x_3\})P_{t_1}(\{x_1, x_4\})P_{t_3}(\{x_2, x_5\}) + Q_{t_2}(\{x_2\})P_{t_1}(\{x_1, x_4\})P_{t_3}(\{x_3, x_5\}). \end{aligned}$$

Note that this recursion matches the permanent expansion in Example 4.6.

In the following sections we explain the two main routines of Algorithm 4, i.e., SUBPERMS and EVALRECURSION, obtaining complexity bounds for them.

4.3.3 Permanent of all submatrices

Let M_0 be a rectangular matrix with row set A_0 and column set X_0 . As a part of our algorithm, which can be seen as the base case, we require a good method to compute the permanents of

all submatrices of M_0 . In other words, we want to obtain the partial permanents $\text{perm}(D, Y)$ for all pairs (D, Y) . We can do this in a very simple way using an expansion by minors. The following lemma explains such procedure and gives its running time.

Lemma 4.1. *Let M_0 be a matrix of dimensions $n_1 \times n_2$. Let A_0 denote its row set, X_0 its column set and let $S = \{(D, Y) \subseteq A_0 \times X_0 : |D| = |Y|\}$. We can compute $\text{perm}(D, Y)$ for all $(D, Y) \in S$ in $O(n_{\max}^2 2^{n_1+n_2})$, where $n_{\max} = \max\{n_1, n_2\}$.*

Proof. Let $S_i = \{(D, Y) : |D| = |Y| = i\}$ for $1 \leq i \leq \min\{n_1, n_2\}$. We use an expansion by minors to compute $\text{perm}(D, Y)$ for $(D, Y) \in S_i$, using the values of S_{i-1} . Let a_0 be the first element in D , then

$$\text{perm}(D, Y) = \sum_{x \in Y} M_{a_0, x} \text{perm}(D \setminus a_0, Y \setminus x).$$

Thus, for each (D, Y) , we loop over at most n_2 elements, and for each we need $O(n_{\max})$ to find the sets $D \setminus a_0$ and $Y \setminus x$. The result follows. \square

4.3.4 Recursion formula

The heart of Algorithm 4 is given by the recursion formula used, i.e., the procedure to obtain table P_t of node t from the tables of its children. This recursion formula is given in the following lemma.

Lemma 4.2. *Let M be a matrix with associated column graph G^X . Let (T, χ) be a tree decomposition of G^X . Let t be an internal node of T , and let Y be such that*

$$\chi(T_t) \setminus \chi(t) \subseteq Y \subseteq \chi(T_t), \quad |Y| = |A_{T_t}| \tag{4.2}$$

Let c_1, \dots, c_k be the children of t . Then

$$\text{perm}(A_{T_t}, Y) = \sum_Y \text{perm}(A_t, Y_t) \prod_{j=1}^k \text{perm}(A_{T_{c_j}}, Y_{c_j}) \tag{4.3}$$

where $\text{perm}(\cdot, \cdot)$ is as in (4.1) and the sum is over all $\mathcal{Y} = (Y_t, Y_{c_1}, \dots, Y_{c_k})$ such that:

$$Y = Y_t \sqcup (Y_{c_1} \sqcup \dots \sqcup Y_{c_k}) \quad (4.4a)$$

$$\chi(T_{c_j}) \setminus \chi(t) \subseteq Y_{c_j} \subseteq \chi(T_{c_j}) \quad Y_t \subseteq \chi(t). \quad (4.4b)$$

Proof. Observe that A_{T_t} can be partitioned as

$$A_{T_t} = A_t \sqcup (A_{T_{c_1}} \sqcup \dots \sqcup A_{T_{c_k}}).$$

Let $\pi : A_{T_t} \rightarrow Y$ be a matching. Let c be a child of t and let $\pi_c : A_{T_c} \rightarrow Y$ be the restriction of π to A_{T_c} . Let $Y_t := \pi(A_t) \subseteq \chi(t)$ be the range of π restricted to A_t , and let Y_c be the range of π_c . As π is injective, then equation (4.4a) holds. Observe also that $Y_c := \pi(A_{T_c}) \subseteq \chi(T_c)$. Note now that if $x \in \chi(T_c) \setminus \chi(t)$, then it is in the range of π . However, as $x \notin \chi(t)$ then $x \notin \chi(T_{c'})$ for any other child c' (using condition (iii) of Definition 2.6), and thus x has to be in the range of π_c . Thus, the range of π_c , i.e., Y_c , contains $\chi(T_c) \setminus \chi(t)$.

Therefore, for any matching $\pi : A_{T_t} \rightarrow Y$ and for any child c , π induces a matching from A_{T_c} to some Y_c that satisfy equations (4.4). On the other hand, given Y_t, Y_{c_1}, \dots satisfying (4.4) and matchings π_t, π_{c_1}, \dots on $A_t, A_{T_{c_1}}, \dots$ with such ranges, we can merge them into a function on A_{T_t} . Observe that (4.4) ensures that the ranges of these matchings are disjoint and their union is Y . We conclude that

$$\begin{aligned} \text{perm}(A_{T_t}, Y) &= \sum_{\pi: A_{T_t} \rightarrow Y} \prod_a M_{a, \pi(a)} \\ &= \sum_{\mathcal{Y}} \sum_{\substack{\pi_t: A_t \rightarrow Y_t \\ \pi_c: A_{T_c} \rightarrow Y_c}} \left(\prod_{a_t} M_{a_t, \pi_t(a_t)} \right) \prod_{c_j} \left(\prod_{a_c} M_{a_c, \pi_{c_j}(a_c)} \right) \\ &= \sum_{\mathcal{Y}} \text{perm}(A_t, Y_t) \prod_{c_j} \text{perm}(A_{T_{c_j}}, Y_{c_j}). \end{aligned}$$

□

At first sight, the recursion of equation (4.3) looks difficult to evaluate. It turns out that this formula is a subset convolution and thus it can be computed efficiently using the algorithm from [20], as explained in the following lemma. We follow this approach in method

EVALRECURSION of Algorithm 4.

Lemma 4.3. *Given the values of the partial permanents $\text{perm}(A_t, Y_t)$ and $\text{perm}(A_{T_{c_j}}, Y_{c_j})$, we can evaluate equation (4.3) for all Y satisfying (4.2) in $\tilde{O}(k2^\omega)$.*

Proof. Let $\bar{Y} := Y \cap \chi(t)$, $\bar{Y}_t := Y_t$, $\bar{Y}_{c_j} := Y_{c_j} \cap \chi(t)$, and let

$$\begin{aligned} \Delta_t &:= \chi(T_t) \setminus \chi(t) & \Delta_{c_j} &:= \chi(T_{c_j}) \setminus \chi(t) \\ P_t(\bar{Y}) &:= \text{perm}(A_{T_t}, \bar{Y} \cup \Delta_t), & Q_t(\bar{Y}_t) &:= \text{perm}(A_t, \bar{Y}_t), & Q_{c_j}(\bar{Y}_{c_j}) &:= \text{perm}(A_{T_{c_j}}, \bar{Y}_{c_j} \cup \Delta_{c_j}) \end{aligned}$$

Then equation (4.3) can be rewritten as

$$P_t(\bar{Y}) = \sum_{\bar{Y}_t \sqcup \bar{Y}_{c_1} \sqcup \dots \sqcup \bar{Y}_{c_k} = \bar{Y}} Q_t(\bar{Y}_t) \prod_{j=1}^k Q_{c_j}(\bar{Y}_{c_j}) \quad (4.5)$$

where $\bar{Y}_t \subseteq \chi(t)$ and $\bar{Y}_{c_j} \subseteq \chi(c_j) \cap \chi(t)$. Equation (4.5) is a subset convolution over the subsets of $\chi(t)$. Therefore, it can be evaluated in $O(kw^2 2^w)$, where $w := |\chi(t)|$, using the algorithm from [20]. \square

The following theorem gives the running time of Algorithm 4, proving that we can efficiently compute the permanent given a tree decomposition of G^X of small width.

Theorem 4.4. *Let M be a matrix with associated column graph G^X . Let (T, χ) be a tree decomposition of G^X of width ω . Then we can compute $\text{Perm}(M)$ in $\tilde{O}(n4^\omega)$.*

Proof. Let t be some node in T . We compute $\text{perm}(A_{T_t}, Y)$ for every Y satisfying (4.2). In particular, we will obtain $\text{Perm}(A) = \text{perm}(A_{T_{\text{root}}}, \chi(T_{\text{root}}))$. We will show that for each t we can compute $\text{perm}(A_{T_t}, Y)$ for all Y in $\tilde{O}((k_t + 1)4^\omega)$, where k_t is the number of children of t . Note that $\sum_t k_t$ is the number of nodes of tree T . As the tree has $O(n)$ nodes, the total cost is then $\tilde{O}(n4^\omega)$, as wanted.

The base case is when t is a leaf of T , so that $A_{T_t} = A_t$ and $\chi(T_t) = \chi(t)$. Let M_0 be the submatrix of M with rows A_t and columns $\chi(t)$. Then all we have to do is to obtain the permanent of some submatrices of M_0 . Observe that $|A_t| \leq |\chi(t)| \leq \omega$, as otherwise there is no Y satisfying (4.2). Thus, we can do this in $\tilde{O}(2^{2\omega})$ using Lemma 4.1.

Assume now that t is an internal node of T with k_t children and let Y that satisfies (4.2). Then equation (4.3) tells us how to find $\text{perm}(A_{T_t}, Y)$. Lemma 4.3 says that we can evaluate the formula in $\tilde{O}(k_t 2^\omega)$, assuming we know the values of the terms in the recursion. Note that we already found $\text{perm}(A_{T_c}, Y_c)$ for all children and that we can find $\text{perm}(A_t, Y_t)$ for all possible Y_t in $\tilde{O}(2^{2\omega})$ in the same way as for the base case. Thus, it takes $\tilde{O}(4^\omega + k_t 2^\omega) = \tilde{O}((k_t + 1)4^\omega)$ to compute $\text{perm}(A_{T_t}, Y)$ for all Y . \square

Remark 4.3. The factor $\tilde{O}(4^\omega)$ can be improved, but we omit this as for the approach of Section 4.4, based on the bipartite graph, the bound will be $\tilde{O}(n 2^\omega)$.

4.3.5 Computing the determinant

Given the similarity between permanent and determinant, it should be possible to find an analogous algorithm for the determinant. We will derive such algorithm in this section. Ironically, this algorithm is slower than the one for the permanent. The reason is that the approach we follow does not take advantage of linear algebra: we loop over all permutations (carefully) and then compute its sign. We remark that our algorithm does not use divisions and thus can be applied in any commutative ring. The ideas from this section will be used in Section 4.5 to derive a decomposition algorithm for the mixed discriminant.

Example 4.8. Consider again the matrix M of Figure 4-4, and observe that a similar expansion holds for the determinant:

$$\begin{aligned} \text{Det}(M) = & \text{Det} \begin{pmatrix} M_{a_1, x_1} & M_{a_1, x_3} \\ M_{a_2, x_1} & M_{a_2, x_3} \end{pmatrix} \text{Det} \begin{pmatrix} M_{a_3, x_4} \end{pmatrix} \text{Det} \begin{pmatrix} M_{a_4, x_2} & M_{a_4, x_5} \\ M_{a_5, x_2} & M_{a_5, x_5} \end{pmatrix} \\ & - \text{Det} \begin{pmatrix} M_{a_1, x_1} & M_{a_1, x_4} \\ M_{a_2, x_1} & M_{a_2, x_4} \end{pmatrix} \text{Det} \begin{pmatrix} M_{a_3, x_3} \end{pmatrix} \text{Det} \begin{pmatrix} M_{a_4, x_2} & M_{a_4, x_5} \\ M_{a_5, x_2} & M_{a_5, x_5} \end{pmatrix} \\ & + \text{Det} \begin{pmatrix} M_{a_1, x_1} & M_{a_1, x_4} \\ M_{a_2, x_1} & M_{a_2, x_4} \end{pmatrix} \text{Det} \begin{pmatrix} M_{a_3, x_2} \end{pmatrix} \text{Det} \begin{pmatrix} M_{a_4, x_3} & M_{a_4, x_5} \\ M_{a_5, x_3} & M_{a_5, x_5} \end{pmatrix}. \end{aligned}$$

As suggested in the above formula, the recursion used to compute the permanent can also be used to compute the determinant, by appropriately selecting the signs.

We recall now the definition of the parity function, and we extend it to ordered partitions.

Definition 4.4. Let D, Y be ordered sets of the same size. For a bijection $\pi : D \rightarrow Y$ we define its *sign* or *parity* as $\text{sgn}(\pi) := (-1)^{N(\pi)}$, where $N(\pi)$ is its number of inversions:

$$N(\pi) := |\{(a, a') \in D^2 : a < a', \pi(a) > \pi(a')\}|.$$

Let $\mathcal{Y} = (Y_1, \dots, Y_k)$ be an *ordered partition* of Y , i.e., $Y = Y_1 \sqcup \dots \sqcup Y_k$. We define its *sign* to be $\text{sgn}(\mathcal{Y}) := (-1)^{N(\mathcal{Y})}$, where $N(\mathcal{Y})$ is:

$$N(\mathcal{Y}) := |\{(y_i, y_j) : y_i \in Y_i, y_j \in Y_j, i < j, y_i > y_j\}|.$$

Equivalently, we can associate to \mathcal{Y} a permutation $\pi^{\mathcal{Y}} : \{1, 2, \dots, |Y|\} \rightarrow Y$ that consists of blocks: we put first Y_1 (sorted), then Y_2 (sorted), and so on. Then $\text{sgn}(\mathcal{Y}) = \text{sgn}(\pi^{\mathcal{Y}})$.

From the definition above it is clear that we can obtain the sign of a permutation in $O(n^2)$ by counting the number of inversions. However, it is well known that we can find it in $O(n)$ by counting its cycles.

For a matrix M , there is a natural order for its row set A and column set X , namely from top to bottom and from left to right. We recall the definition of the determinant:

$$\text{Det}(M) = \sum_{\pi} \text{sgn}(\pi) \prod_{a \in A} M_{a, \pi(a)}$$

where the sum is over all bijections $\pi : A \rightarrow X$. Similarly, for a fixed matrix M and for some $D \subseteq A$ and $Y \subseteq X$ we define the partial determinants:

$$\det(D, Y) := \sum_{\pi} \text{sgn}(\pi) \prod_{a \in D} M_{a, \pi(a)} \tag{4.6}$$

where the sum is over all bijections $\pi : D \rightarrow Y$. Note that $\text{Det}(M) = \det(A, X)$.

We now provide a recursion formula similar to the one in Lemma 4.2. We need one lemma before.

Lemma 4.5. Let D, Y be ordered sets, and let $\pi : D \rightarrow Y$ be a bijection, which we view as a subset of $D \times Y$. Let $\mathcal{D} = (D_1, \dots, D_k)$ and $\mathcal{Y} = (Y_1, \dots, Y_k)$ be partitions of D and Y . Let

$\pi = \pi_1 \sqcup \dots \sqcup \pi_k$ be a decomposition with $\pi_j \subseteq D_j \times Y_j$. Then

$$\text{sgn}(\pi) = \text{sgn}(\mathcal{D})\text{sgn}(\mathcal{Y}) \prod_{j=1}^k \text{sgn}(\pi_j).$$

Proof. It follows from the multiplicativity of the sign function. \square

Lemma 4.6. *Under the same conditions of Lemma 4.2, then*

$$\det(A_{T_t}, Y) = \text{sgn}(\mathcal{D}) \sum_{\mathcal{Y}} \text{sgn}(\mathcal{Y}) \det(A_t, Y_t) \prod_{j=1}^k \det(A_{T_{c_j}}, Y_{c_j}) \quad (4.7)$$

where $\det(\cdot, \cdot)$ is as in (4.6) the sum is over all $\mathcal{Y} = (Y_s, Y_{c_1}, \dots, Y_{c_k})$ satisfying (4.4), $\mathcal{D} = (A_t, A_{T_{c_1}}, \dots, A_{T_{c_k}})$ and $\text{sgn}(\cdot)$ is as in Definition 4.4.

Proof. The proof is basically the same as the one of Lemma 4.2. The only difference is that we have the additional factor $\text{sgn}(\pi)$, but it factors because of Lemma 4.5. \square

Despite the resemblance between equations (4.3) and (4.7), the latter is not a subset convolution because of the sign factors. Therefore, we cannot use the algorithm from [20] in this case. We now show to the complexity analysis.

Lemma 4.7. *Given the values of the partial determinants $\det(A_t, Y_t)$ and $\det(A_{T_{c_j}}, Y_{c_j})$, we can evaluate equation (4.7) for all Y satisfying (4.2) in $\tilde{O}(k(n + 3^\omega))$.*

Proof. We will first express equation (4.7) in a similar format as formula (4.5) of Lemma 4.3. To simplify the notation, let $\mathcal{Y} = (Y_0, Y_1, \dots, Y_k)$. For each j let $Y_0^j = Y_0 \cup Y_1 \cup \dots \cup Y_j$, and observe that $\text{sgn}(\mathcal{Y}) = \prod_j \text{sgn}(Y_0^{j-1}, Y_j)$. Then equation (4.7) can be rewritten as:

$$D(\bar{Y}) = \text{sgn}(\mathcal{D}) \sum_{\bar{Y}_0 \sqcup \dots \sqcup \bar{Y}_k = \bar{Y}} \prod_{j=0}^k S_j(\bar{Y}_0^{j-1}, \bar{Y}_j) D_j(\bar{Y}_j)$$

where $\bar{Y}_0, \dots, \bar{Y}_k \subseteq \chi(t)$ and

$$\begin{aligned}\Delta &:= \chi(T_t) \setminus \chi(t) & \Delta_0 &:= \emptyset & \Delta_j &:= \chi(T_{c_j}) \setminus \chi(t) \\ D(\bar{Y}) &:= \det(A_{T_t}, \bar{Y} \cup \Delta), & D_0(\bar{Y}_0) &:= \det(A_t, \bar{Y}_0 \cup \Delta_0), & D_j(\bar{Y}_j) &:= \det(A_{T_{c_j}}, \bar{Y}_j \cup \Delta_j) \\ \bar{Y}_0^j &= \bar{Y}_0 \cup \dots \cup \bar{Y}_j & \Delta_0^j &:= \Delta_0 \cup \dots \cup \Delta_j \\ S_j(\bar{Y}_0^{j-1}, \bar{Y}_j) &:= \text{sgn}(\bar{Y}_0^{j-1} \cup \Delta_0^{j-1}, \bar{Y}_j \cup \Delta_j)\end{aligned}$$

For each $0 \leq l \leq k$, and for each $\bar{Y}_0^l \subseteq \chi(t)$, let

$$D_0^l(\bar{Y}_0^l) = \text{sgn}(\mathcal{D}) \sum_{\bar{Y}_0 \sqcup \dots \sqcup \bar{Y}_l = \bar{Y}_0^l} \prod_{j=0}^l S_j(\bar{Y}_0^{j-1}, \bar{Y}_j) D_j(\bar{Y}_j)$$

Note that $D_0^k(\bar{Y}) = D(\bar{Y})$, and thus it is enough to compute D_0^l for all l . We can do this recursively, observing that $D_0^0(\bar{Y}) = \text{sgn}(\mathcal{D}) D_0(\bar{Y})$ and

$$D_0^{l+1}(\bar{Y}_0^{l+1}) = \sum_{\bar{Y}_0^l \sqcup \bar{Y}_{l+1} = \bar{Y}_0^{l+1}} S_{l+1}(\bar{Y}_0^l, \bar{Y}_{l+1}) D_0^l(\bar{Y}_0^l) D_{l+1}(\bar{Y}_{l+1}) \quad (4.8)$$

We reduced the problem to evaluating the above formula, and we will show that for each l we can do this in $\tilde{O}(n + 3^\omega)$. Assume for now that the signs $S_{l+1}(\bar{Y}_0^l, \bar{Y}_{l+1})$ are known. Then for each \bar{Y}_0^{l+1} of cardinality i , we can evaluate (4.8) in $O(2^i)$. Thus, for all \bar{Y}_0^{l+1} we require $O(\sum_i \binom{w}{i} 2^i) = O(3^w)$, where $w = |\chi(t)|$. We will see that after a precomputation that takes $\tilde{O}(n)$, we can obtain $S_{l+1}(\bar{Y}_0^l, \bar{Y}_{l+1})$ in $\tilde{O}(1)$, which will complete the proof.

Observe that

$$S_{l+1}(\bar{Y}_0^l, \bar{Y}_{l+1}) = \text{sgn}(\bar{Y}_0^l, \bar{Y}_{l+1}) \text{sgn}(\bar{Y}_0^l, \Delta_{l+1}) \text{sgn}(\Delta_0^l, \bar{Y}_{l+1}) \text{sgn}(\Delta_0^l, \Delta_{l+1}).$$

Note that the last factor does not depend on the partition and it can be precomputed in $O(n)$. Also note that the first factor can be computed in $O(\omega) = \tilde{O}(1)$, so we can ignore it. We are left with the second and third factor.

For each $x \in \chi(t)$, let

$$N_x^{\Delta_{l+1}} = |\{y \in \Delta_{l+1} : x > y\}|.$$

We can precompute $N_x^{\Delta_{l+1}}$ for all x in $O(\omega n) = \tilde{O}(n)$. Note that $\text{sgn}(\bar{Y}_0^l, \Delta_{l+1}) = (-1)^N$ where $N = \sum_{x \in \bar{Y}_0^l} N_x^{\Delta_{l+1}}$. Thus, after the precomputation, we can obtain this factor in $O(\omega) = \tilde{O}(1)$. A similar procedure can be done for $\text{sgn}(\Delta_0^l, \bar{Y}_{l+1})$. \square

Theorem 4.8. *Let M be a matrix with associated column graph G^X . Let (T, χ) be a tree decomposition of G^X of width ω . Then we can compute $\text{Det}(M)$ in $\tilde{O}(n^2 + n 4^\omega)$.*

Proof. There are two changes with respect to the proof of Theorem 4.4. First, in the base case we need to compute the determinant of all submatrices of M_0 . Using an expansion by minors as in the proof of Lemma 4.1, we can do this in $\tilde{O}(4^\omega)$, i.e., the same as for the permanent. Second, for the recursion formula we use Lemma 4.7. This increases the time per node from $\tilde{O}(k_t 2^\omega)$ to $\tilde{O}(k_t(n + 3^\omega))$. Therefore, the overall cost is $\tilde{O}(n^2 + n 4^\omega)$. \square

We conclude this section by presenting an open question. Given the resemblance in the definition of the permanent and the determinant it is not surprising that they can be computed using very similar tree decomposition methods. The *immanant* of a matrix is another closely related notion:

$$\text{Imm}_\lambda(M) := \sum_{\pi} \chi_\lambda(\pi) \prod_{a \in A} M_{a, \pi(a)}$$

where the sum is over all bijections $\pi : A \rightarrow X$, and χ_λ is an irreducible character of the symmetric group. The immanant reduces to the permanent when χ_λ is the trivial character, and it reduces to the determinant when χ_λ is the sign character. The computational complexity of immanants has been analyzed in e.g., [31]. A natural question that arises is whether a tree decomposition method can be used to compute them. We remark that the recursion in (4.7) does not hold for the immanant as χ_λ is not necessarily multiplicative.

Question. *Given a matrix M of bounded treewidth, can we compute $\text{Imm}_\lambda(M)$ in polynomial time? In particular, can this be done if either the height or the width of the Young diagram is bounded?*

4.4 Bipartite decompositions

In the previous section we showed a decomposition method based on the column graph G^X . We showed that we can compute the permanent in $\tilde{O}(n4^{\omega_X})$, where ω_X is the treewidth of G^X . In this section we will extend this decomposition method to work with the bipartite graph G (see Definition 4.1). We will show that we can compute the permanent in $\tilde{O}(n2^\omega)$, where ω is the treewidth of G . A Matlab implementation of our algorithm is available in www.mit.edu/~diegcif.

Let G be the bipartite graph of M . As in the previous sections, we index the rows with a set A and the columns with X . We now rephrase the definition of a tree decomposition of G . A *bipartite decomposition* of G is a tuple (T, α, χ) , where T is a rooted tree and $\alpha : T \rightarrow \{0, 1\}^A$, $\chi : T \rightarrow \{0, 1\}^X$ assign some $\alpha(t) \subseteq A$ and $\chi(t) \subseteq X$ to each node t of T , that satisfies the following conditions.

- i-1. The union of $\{\alpha(t)\}_{t \in T}$ is the whole row set A .
- i-2. The union of $\{\chi(t)\}_{t \in T}$ is the whole column set X .
- ii. For every edge (a, x) of G there exists a node t of T with $a \in \alpha(t), x \in \chi(t)$.
- iii-1. For every $a \in A$ the set $\{t : a \in \alpha(t)\}$ forms a subtree of T .
- iii-2. For every $x \in X$ the set $\{t : x \in \chi(t)\}$ forms a subtree of T .

The width ω of the decomposition is the largest of $|\alpha(t)| + |\chi(t)|$ among all nodes t . Note that the above literals are consistent with the ones in Definition 2.6.

As before, we now present an example to illustrate the use of the bipartite graph for computing the permanent.

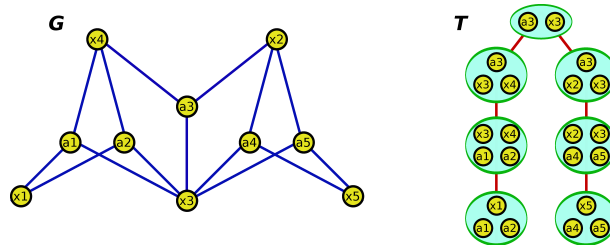


Figure 4-5: Bipartite graph G of the matrix from Figure 4-4, and a tree decomposition T .

Example 4.9. Consider again the matrix M of Figure 4-4. Note that $\text{Perm}(M)$ can also be expressed in the following form:

$$\begin{aligned} \text{Perm}(M) &= \text{perm}(\{a_1, a_2\}, \{x_1, x_4\}) \text{perm}(\{a_3, a_4, a_5\}, \{x_2, x_3, x_5\}) \\ &\quad + \text{perm}(\{a_1, a_2, a_3\}, \{x_1, x_3, x_4\}) \text{perm}(\{a_4, a_5\}, \{x_2, x_5\}) \\ &\quad - M_{a_3, x_3} \text{perm}(\{a_1, a_2\}, \{x_1, x_4\}) \text{perm}(\{a_4, a_5\}, \{x_2, x_5\}) \\ \text{perm}(\{a_1, a_2, a_3\}, \{x_1, x_3, x_4\}) &= M_{a_3, x_3} \text{perm}(\{a_1, a_2\}, \{x_1, x_4\}) + M_{a_3, x_4} \text{perm}(\{a_1, a_2\}, \{x_1, x_3\}) \\ \text{perm}(\{a_3, a_4, a_5\}, \{x_2, x_3, x_5\}) &= M_{a_3, x_3} \text{perm}(\{a_4, a_5\}, \{x_2, x_5\}) + M_{a_3, x_2} \text{perm}(\{a_4, a_5\}, \{x_3, x_5\}) \end{aligned}$$

To evaluate the above formula we need to compute four 2×2 permanents, and we need in total 16 multiplications. It turns out that this formula arises by considering the tree decomposition of the bipartite graph shown in Figure 4-5.

4.4.1 Bipartite decomposition algorithm

Algorithm 5 presents our dynamic programming method to compute $\text{Perm}(M)$ using a bipartite decomposition. As for Algorithm 4, for each node t we compute a table P_t , following a topological ordering of the tree. The permanent of M is in the table corresponding to the root. There are two main routines: SUBPERM computes the permanents of all submatrices, and EVALRECURSION evaluates a recursion formula, which is slightly more complex than the one of Algorithm 4.

As before, the values $P_t(\bar{D}, \bar{Y})$ computed correspond to a partial permanent of the matrix. Consider the collection

$$S = \{(D, Y) : \alpha(T_t) \setminus \alpha(t) \subseteq D \subseteq \alpha(T_t), \chi(T_t) \setminus \chi(t) \subseteq Y \subseteq \chi(T_t)\}.$$

Observe that $(D, Y) \in S$ is completely determined by $(D \cap \alpha(t), Y \cap \chi(t))$. Therefore, if we let $\bar{D} = D \cap \alpha(t)$, $\bar{Y} := Y \cap \chi(t)$, there is a one to one correspondence between S , and the collection $\bar{S} := \{(\bar{D}, \bar{Y}) : \bar{D} \subseteq \alpha(t), \bar{Y} \subseteq \chi(t)\}$. The partial permanents that we are interested in are

$$P_t(\bar{D}, \bar{Y}) := \text{perm}(D, Y) = \text{perm}(\bar{D} \cup (\alpha(T_t) \setminus \alpha(t)), \bar{Y} \cup (\chi(T_t) \setminus \chi(t))).$$

Algorithm 5 Permanent with bipartite decomposition

Input: Matrix M and tree decomposition (T, α, χ) of bipartite graph $G(M)$

Output: Permanent of M

```

1: procedure BIPARTPERM( $M, T, \alpha, \chi$ )
2:    $order :=$  topological ordering of  $T$  starting from its leaves
3:   for  $t$  in  $order$  do
4:      $Q_t :=$  SUBPERMS( $t, M$ )
5:     if  $t$  is a leaf then
6:        $P_t := Q_t$ 
7:     else
8:        $c_1, \dots, c_k :=$  children of  $t$ 
9:        $P_t :=$  EVALRECURSION( $t, Q_t, P_{c_1}, \dots, P_{c_k}$ )
10:  return  $P_{root}(\alpha(root), \chi(root))$ 
11: procedure SUBPERMS( $t, M$ )
12:   $Q_t(\bar{D}, \bar{Y}) := perm(\bar{D}, \bar{Y})$  for all  $\bar{D} \subseteq \alpha(t), \bar{Y} \subseteq \chi(t)$ 
13: procedure EVALRECURSION( $t, Q_t, P_{c_1}, \dots, P_{c_k}$ )
14:  for  $c_j$  child of  $t$  do
15:     $\Delta_j^\alpha := \alpha(c_j) \setminus \alpha(t), \quad \Lambda_j^\alpha := \alpha(c_j) \cap \alpha(t)$ 
16:     $\Delta_j^\chi := \chi(c_j) \setminus \chi(t), \quad \Lambda_j^\chi := \chi(c_j) \cap \chi(t)$ 
17:     $Q_{tc_j}(\bar{D}, \bar{Y}) := (-1)^{|\bar{D}|} Q_t(\bar{D}, \bar{Y})$  for all  $\bar{D} \subseteq \Lambda_j^\alpha, \bar{Y} \subseteq \Lambda_j^\chi$ 
18:     $Q_{cc_j}(\bar{D}, \bar{Y}) := P_{c_j}(\bar{D} \cup \Delta_j^\alpha, \bar{Y} \cup \Delta_j^\chi)$  for all  $\bar{D} \subseteq \Lambda_j^\alpha, \bar{Y} \subseteq \Lambda_j^\chi$ 
19:   $P_t :=$  SUBSETCONVOLUTION( $Q_t, Q_{tc_1}, Q_{cc_1}, \dots, Q_{tc_k}, Q_{cc_k}$ )
20: procedure SUBSETCONVOLUTION( $P_0, P_1, \dots, P_{2k}$ )
21:   $P(\bar{D}, \bar{Y}) := \sum_{\substack{\bar{D}_0 \sqcup \dots \sqcup \bar{D}_{2k} = \bar{D} \\ \bar{Y}_0 \sqcup \dots \sqcup \bar{Y}_{2k} = \bar{Y}}} P_0(\bar{D}_0, \bar{Y}_0) P_1(\bar{D}_1, \bar{Y}_1) \dots P_{2k}(\bar{D}_{2k}, \bar{Y}_{2k})$ 

```

For the root node $P_{root}(\alpha(root), \chi(root)) = perm(A, X) = Perm(M)$.

4.4.2 Recursion formula

The recursion formula that method EVALRECURSION of Algorithm 5 evaluates is given in the following lemma.

Lemma 4.9. *Let M be a matrix with associated bipartite graph G . Let (T, α, χ) be a bipartite decomposition of G . Let t be an internal node of T , let $T_t \subseteq T$ denote the subtree rooted in t , and let D, Y be such that*

$$\alpha(T_t) \setminus \alpha(t) \subseteq D \subseteq \alpha(T_t), \quad \chi(T_t) \setminus \chi(t) \subseteq Y \subseteq \chi(T_t), \quad |D| = |Y| \quad (4.9)$$

Let t_{c_1}, \dots, t_{c_k} be the children of t . Then

$$perm(D, Y) = \sum_{\mathcal{D}, \mathcal{Y}} perm(D_t, Y_t) \prod_{j=1}^k (-1)^{|D_{tc_j}|} perm(D_{tc_j}, Y_{tc_j}) perm(D_{cc_j}, Y_{cc_j}) \quad (4.10)$$

where $perm(\cdot, \cdot)$ is as in (4.1) and the sum is over all $\mathcal{D} = (D_t, D_{tc_1}, \dots)$, $\mathcal{Y} = (Y_t, Y_{tc_1}, \dots)$ satisfying:

$$\begin{aligned} D &= D_t \sqcup (D_{tc_1} \sqcup D_{cc_1} \sqcup \dots \sqcup D_{tc_k} \sqcup D_{cc_k}) \\ Y &= Y_t \sqcup (Y_{tc_1} \sqcup Y_{cc_1} \sqcup \dots \sqcup Y_{tc_k} \sqcup Y_{cc_k}) \\ \alpha(T_{c_j}) \setminus \alpha(t) &\subseteq D_{cc_j} \subseteq \alpha(T_{c_j}) \quad D_t \subseteq \alpha(t) \quad D_{tc_j} \subseteq \alpha(t) \cap \alpha(t_{c_j}) \\ \chi(T_{c_j}) \setminus \chi(t) &\subseteq Y_{cc_j} \subseteq \chi(T_{c_j}) \quad Y_t \subseteq \chi(t) \quad Y_{tc_j} \subseteq \chi(t) \cap \chi(t_{c_j}). \end{aligned}$$

To prove this lemma we need some additional notation. We view a bijection $\pi : D \rightarrow Y$ as a subset of $D \times Y$, by identifying it with the set $\{(a, \pi(a)) : a \in D\}$. For a given node t and for some D, Y satisfying (4.9), we denote

$$perm^*(D, Y) := \sum_{\pi} \prod_{a \in D} M_{a, \pi(a)}$$

where the sum is over all bijections $\pi : D \rightarrow Y$ such that

$$\pi \cap (\alpha(t) \times \chi(t)) = \emptyset. \quad (4.11)$$

We now show a different recursion formula, which is closer to the one in Lemma 4.2.

Lemma 4.10. *Following the same notation as above, the following equation holds:*

$$\text{perm}(D, Y) = \sum_{Y_t, Y_{c_j}, D_t, D_{c_j}} \text{perm}(D_t, Y_t) \prod_{j=1}^k \text{perm}^*(D_{c_j}, Y_{c_j}) \quad (4.12)$$

where the sum is over all $Y_t, Y_{c_j}, D_t, D_{c_j}$ such that

$$D = D_t \sqcup (D_{c_1} \sqcup \cdots \sqcup D_{c_k}) \quad (4.13a)$$

$$Y = Y_t \sqcup (Y_{c_1} \sqcup \cdots \sqcup Y_{c_k}) \quad (4.13b)$$

$$\alpha(T_{c_j}) \setminus \alpha(t) \subseteq D_{c_j} \subseteq \alpha(T_{c_j}) \quad D_t \subseteq \alpha(t) \quad (4.13c)$$

$$\chi(T_{c_j}) \setminus \chi(t) \subseteq Y_{c_j} \subseteq \chi(T_{c_j}) \quad Y_t \subseteq \chi(t). \quad (4.13d)$$

Proof. Let $\pi : D \rightarrow Y$ be a matching, which we view as a subset of $D \times Y$. Note that

$$D = (D \cap \alpha(t)) \sqcup (D \cap \alpha(T_{c_1}) \setminus \alpha(t)) \sqcup \cdots \sqcup (D \cap \alpha(T_{c_k}) \setminus \alpha(t))$$

$$Y = (Y \cap \chi(t)) \sqcup (Y \cap \chi(T_{c_1}) \setminus \chi(t)) \sqcup \cdots \sqcup (Y \cap \chi(T_{c_k}) \setminus \chi(t))$$

Let's decompose π in a similar way as above. Let π_t be the submatching of π with domain contained in $D \cap \alpha(t)$ and range contained in $Y \cap \chi(t)$. Equivalently, $\pi_t = \pi \cap (D \cap \alpha(t)) \times (Y \cap \chi(t))$. Observe that if some $a \in D \cap \alpha(t)$ is not in the domain of π_t , then $a \in \alpha(T_c)$, $\pi(a) \in \chi(T_c)$ for some child c . The reason is that by definition of tree decomposition, there must be some node t_p with $\pi(a) \in \chi(t_p)$, $a \in \alpha(t_p)$. However, the assumption on a says that $\pi(a) \in \chi(T_t) \setminus \chi(t)$ and thus we must have $t_p \in T_c$ for some c . Similarly, if some $x \in Y \cap \chi(t)$ is not in the range of π_t , then $x \in \chi(T_c)$, $\pi^{-1}(x) \in \alpha(T_c)$ for some child c . In the same way,

we have the following

$$\begin{aligned}\pi(D \cap \alpha(T_c) \setminus \alpha(t)) &\subseteq D \cap \chi(T_c) \\ \pi^{-1}(Y \cap \chi(T_c) \setminus \chi(t)) &\subseteq Y \cap \alpha(T_c)\end{aligned}$$

The above paragraph implies that we can decompose

$$\pi = \pi_t \sqcup \pi_{c_1} \sqcup \cdots \sqcup \pi_{c_k} \tag{4.14}$$

in such a way that $\pi_t \subseteq (D \cap \alpha(t)) \times (Y \cap \chi(t))$ and for each child c we have that $\pi_c \subseteq (D \cap \alpha(T_c)) \times (Y \cap \chi(T_c))$ and π_c satisfies (4.11). Moreover, it is easy to see that such decomposition is unique.

Let $Y_t \subseteq Y \cap \chi(t)$ be the range of π_t and let $Y_c \subseteq Y \cap \chi(T_c)$ be the range of π_c . Analogously, define $D_t \subseteq D \cap \alpha(t)$ and $D_c \subseteq D \cap \alpha(T_c)$ as the domains of π_t, π_c . Observe that $\chi(T_c) \setminus \chi(t) \subseteq Y_c$, as if $x \in \chi(T_c) \setminus \chi(t)$ then $(\pi^{-1}(x), x) \in \pi_c$ by construction, and thus $x \in Y_c$. Similarly, $\chi(T_c) \setminus \chi(t) \subseteq Y_c$. Therefore, the equations (4.13) are satisfied.

Thus, for any matching $\pi \subseteq D \times Y$ there is a unique partition as in (4.14) such that the ranges and domains Y_t, Y_c, D_t, D_c satisfy (4.13). On the other hand, assume that Y_t, Y_c, D_t, D_c satisfy (4.13), and we are given some matchings π_t, π_c with these ranges and domains and such that (4.11) holds. Then equations (4.13) tell us that we can merge them into a matching π with domain D and range Y . Condition (4.11) ensures we are not overcounting, as it implies that decomposition (4.14) is unique. These remarks imply equation (4.12). \square

Remark 4.4. Let (T^X, χ) be a tree decomposition of the column graph G^X , and let (T, α, χ) be the corresponding decomposition of the bipartite graph G given in Example 4.3. In such case, the above lemma reduces to Lemma 4.2.

We now derive the recursion formula in Lemma 4.9, which follows from the above lemma by using inclusion-exclusion.

Proof of Lemma 4.9. For a child c of t , we will show that

$$\text{perm}^*(D_c, Y_c) = \sum_{\substack{D_{tc} \sqcup D_{cc} = D_c \\ Y_{tc} \sqcup Y_{cc} = Y_c \\ D_{tc} \subseteq \alpha(t), Y_{tc} \subseteq \chi(t)}} (-1)^{|D_{tc}|} \text{perm}(D_{tc}, Y_{tc}) \text{perm}(D_{cc}, Y_{cc}). \quad (4.15)$$

Combining equations (4.12) and (4.15) we obtain equation (4.10), concluding the proof.

Given a matching $\pi_c : D_c \rightarrow Y_c$, let $I(\pi_c) := \pi_c \cap (\alpha(t) \times \chi(t))$ and let $I_\alpha(\pi_c) \subseteq \alpha(t)$, $I_\chi(\pi_c) \subseteq \chi(t)$ be the domain and range of $I(\pi_c)$. For some $D_{tc} \subseteq D_c \cap \alpha(t)$, $Y_{tc} \subseteq Y_c \cap \chi(t)$ with $|D_{tc}| = |Y_{tc}|$, let

$$\text{perm}^*(D_c, Y_c; D_{tc}, Y_{tc}) := \sum_{\substack{\pi_c : D_c \rightarrow Y_c \\ I_\alpha(\pi_c) = D_{tc} \\ I_\chi(\pi_c) = Y_{tc}}} \prod_a M_{a, \pi_c(a)}.$$

Note that $\text{perm}^*(D_c, Y_c) = \text{perm}^*(D_c, Y_c; \emptyset, \emptyset)$. Observe now that given matchings $\pi_{tc} : D_{tc} \rightarrow Y_{tc}$ and $\pi_{cc} : D_c \setminus D_{tc} \rightarrow Y_c \setminus Y_{tc}$, we can merge them into a matching $\pi_c^* : D_c \rightarrow Y_c$ that satisfies $I_\alpha(\pi_c^*) \supseteq D_{tc}$, $I_\chi(\pi_c^*) \supseteq Y_{tc}$. Therefore, we have the following equation

$$\text{perm}(D_{tc}, Y_{tc}) \text{perm}(D_c \setminus D_{tc}, Y_c \setminus Y_{tc}) = \sum_{\substack{D_{tc}^* \supseteq D_{tc} \\ Y_{tc}^* \supseteq Y_{tc}}} \text{perm}^*(D_c, Y_c; D_{tc}^*, Y_{tc}^*).$$

Based on the above formula, we can now find $\text{perm}^*(D_c, Y_c)$ using inclusion-exclusion (or Möbius inversion):

$$\text{perm}^*(D_c, Y_c; \emptyset, \emptyset) = \sum_i \sum_{\substack{D_{tc} \subseteq D_c \cap \alpha(t) \\ Y_{tc} \subseteq Y_c \cap \chi(t) \\ |D_{tc}| = |Y_{tc}| = i}} (-1)^i \text{perm}(D_{tc}, Y_{tc}) \text{perm}(D_c \setminus D_{tc}, Y_c \setminus Y_{tc}).$$

Rewriting the above equation leads to (4.15), as wanted. \square

4.4.3 Complexity analysis

We just derived the recursion formula (4.10) which is used in Algorithm 5. As in the proof of Lemma 4.3, this formula is a subset convolution and thus it can be evaluated efficiently using the algorithm from [20]. The overall running time of Algorithm 5 is stated now.

Theorem 4.11. *Let M be a matrix with associated bipartite graph G . Let (T, α, χ) be a bipartite decomposition of G of width ω . Then we can compute $\text{Perm}(M)$ in $\tilde{O}(n 2^\omega)$.*

Proof. The proof is very similar to the one of Theorem 4.4. For each node $t \in T$, we will compute $\text{perm}(D, Y)$ for every pair D, Y that satisfies equation (4.9). We will show that for each t we can compute $\text{perm}(D, Y)$ for all such D, Y in $\tilde{O}((k_t + 1)2^\omega)$, where k_t is the number of children of t . Observe that for the root node t_r we will compute $\text{Perm}(M) = \text{perm}(\alpha(T_r), \chi(T_r))$. This will conclude the proof.

The base case is when t is a leaf of T , so that $T_t = \{t\}$. Let M_0 be the submatrix of M with rows $\alpha(t)$ and columns $\chi(t)$. We need to obtain the permanent of all submatrices of M_0 . As $|\alpha(t)| + |\chi(t)| \leq \omega$, we can do this in $\tilde{O}(2^\omega)$ using Lemma 4.1.

Assume now that t is an internal node of T with k_t children and let D, Y that satisfy (4.9). Then equation (4.10) tells us how to find $\text{perm}(D, Y)$. Similarly as in Lemma 4.3, we can evaluate this formula in $\tilde{O}(k_t 2^\omega)$, assuming we know the values of the terms in the recursion. Note that we already found $\text{perm}(D_{cc}, Y_{cc})$ for all children in the recursion. We can find $\text{perm}(D_t, Y_t)$ for all D_t, Y_t in $\tilde{O}(2^\omega)$ in the same way as for the base case, and this includes the values $\text{perm}(D_{tc}, Y_{tc})$. Then, it takes $\tilde{O}(2^\omega + k_t 2^\omega) = \tilde{O}((k_t + 1)2^\omega)$ to compute $\text{perm}(D, Y)$ for all D, Y . \square

Similarly as in Theorem 4.8, we can find an analogous algorithm for the determinant.

Theorem 4.12. *Let M be a matrix with associated bipartite graph G . Let (T, α, χ) be a bipartite decomposition of G of width ω . Then we can compute $\text{Det}(M)$ in $\tilde{O}(n^2 + n 3^\omega)$.*

Proof. We just need to follow the steps of Section 4.3.5. For instance, the recursion is

$$\det(D, Y) = \sum_{\mathcal{D}, \mathcal{Y}} \text{sgn}(\mathcal{D}) \text{sgn}(\mathcal{Y}) \det(D_t, Y_t) \prod_{j=1}^k (-1)^{|D_{tc_j}|} \det(D_{tc_j}, Y_{tc_j}) \det(D_{cc_j}, Y_{cc_j})$$

where $\mathcal{Y} = (Y_t, Y_{tc}, Y_{cc})$, $\mathcal{D} = (D_t, D_{tc}, D_{cc})$. The complexity analysis is basically the same as in the proof of Theorem 4.8. The base case can be done in $\tilde{O}(2^\omega)$ using an expansion by minors. The recursion can be evaluated in $\tilde{O}(k_t(n + 3^\omega))$ in a similar way as in Lemma 4.7. \square

4.5 Mixed discriminant and higher dimensions

The mixed discriminant of n matrices is a common generalization of the permanent and the determinant. As such, it is also hard to compute in the general case. We show now that the techniques presented earlier generalize to compute mixed discriminants. Even more, we show that this method extends to compute similar functions in higher dimensional tensors.

4.5.1 Mixed discriminant

Let M be a list of n matrices of size $n \times n$. Equivalently, we can think of M as a $n \times n \times n$ array. We index the first coordinate with a set A , and the second and third coordinates with sets X^1, X^2 . The mixed discriminant of M is given by

$$\text{Disc}(M) := \sum_{\pi^1, \pi^2} \text{sgn}(\pi^1) \text{sgn}(\pi^2) \prod_{a \in A} M_{a, \pi^1(a), \pi^2(a)}$$

where the sum is over all bijections $\pi^1 : A \rightarrow X^1$ and $\pi^2 : A \rightarrow X^2$, and sgn is the parity function. For $a \in A$, let M_a denote the $n \times n$ matrix obtained by fixing the first coordinate. Observe that if $M_a = m$ for some matrix m and for all $a \in A$, then $\text{Disc}(M) = n! \text{Det}(m)$. In the case that M_a is diagonal for all $a \in A$, then $\text{Disc}(M) = \text{Perm}(D)$ where D is the matrix obtained by concatenating these diagonals. We refer to [8, 120] for a further discussion of mixed discriminants.

Remark 4.5. The definition of mixed discriminant often assumes that $\{M_a : a \in A\}$ are real symmetric matrices (although the above formula makes sense for any M) due to the connections that arise in such case with several quantities of convex bodies [120]. In particular, mixed discriminants of symmetric positive semidefinite matrices are closely tied to mixed volumes of ellipsoids.

In the case of a $n \times n$ matrix, a bipartite graph was the natural structure to represent its sparsity. Similarly, if we are given a sparse $n \times n \times n$ array M , a natural structure is a *tripartite graph* G , as follows. Let G be the graph on $A \cup X^1 \cup X^2$, where for each nonzero entry M_{a, x_1, x_2} we put a triangle $\{a, x_1, x_2\}$.

We rephrase the definition of a tree decomposition of a tripartite graph G . A *tripartite decomposition* of G is a tuple $(T, \alpha, \chi^1, \chi^2)$, where T is a rooted tree, $\alpha : T \rightarrow \{0, 1\}^A$,

$\chi^1 : T \rightarrow \{0, 1\}^{X^1}$ and $\chi^2 : T \rightarrow \{0, 1\}^{X^2}$, that satisfies the following conditions.

- i. The union of $\{\alpha(t)\}_{t \in T}$ (resp. χ^1, χ^2) is the whole A (resp. X^1, X^2).
- ii. For every triangle (a, x_1, x_2) in G there is a t with $(a, x_1, x_2) \in (\alpha \times \chi^1 \times \chi^2)(t)$.
- iii. For every $a \in A$ (resp. X^1, X^2) the set $\{t : a \in \alpha(t)\}$ is a subtree of T .

The width of the decomposition is the largest of $|\alpha(t)| + |\chi^1(t)| + |\chi^2(t)|$ among all nodes t . Note that the above literals are consistent with the ones in Definition 2.6. In particular, observe that the second condition does not impose additional constraints due to Lemma 2.4.

We proceed to extend the previous results to the mixed discriminant. For some sets $D \subseteq A$, $Y^1 \subseteq X^1$ and $Y^2 \subseteq X^2$ we denote

$$disc(D, Y^1, Y^2) := \sum_{\pi^1, \pi^2} \text{sgn}(\pi^1) \text{sgn}(\pi^2) \prod_{a \in D} M_{\pi^1(a) \pi^2(a)} \quad (4.16)$$

where the sum is over all bijections $\pi^1 : D \rightarrow Y^1$ and $\pi^2 : D \rightarrow Y^2$. This only makes sense if $|D| = |Y^1| = |Y^2|$, and otherwise we can define $disc(D, Y^1, Y^2) = 0$.

As for the case of the permanent, the dynamic program to compute $\text{Disc}(M)$ has two main steps: computing the mixed discriminant of all subarrays of M , and evaluating some recursion formula. For the first step, it is easy to see that the approach from Lemma 4.1 extends, as we show now.

Lemma 4.13. *Let M_0 be a $n_1 \times n_2 \times n_3$ array. Let A_0, X_0^1, X_0^2 be its set of coordinates, and let $S = \{(D, Y^1, Y^2) \subseteq A_0 \times X_0^1 \times X_0^2 : |D| = |Y^1| = |Y^2|\}$. We can compute $disc(D, Y^1, Y^2)$ for all triples in S in $O(n_{max}^3 2^{n_1+n_2+n_3})$, where $n_{max} = \max\{n_1, n_2, n_3\}$.*

Proof. For $i = 1, 2, \dots, \min\{n_1, n_2, n_3\}$ let

$$S_i := \{(D, Y^1, Y^2) \subseteq A_0 \times X_0^1 \times X_0^2 : |D| = |Y^1| = |Y^2| = i\}.$$

We can find $disc(D, Y^1, Y^2)$ for all $(D, Y^1, Y^2) \in S_i$ using the values of S_{i-1} as follows. Let a_0 be the first element in D , it is easy to see that

$$disc(D, Y^1, Y^2) = \sum_{x_1 \in Y^1, x_2 \in Y^2} \epsilon(x_1, x_2) M_{a_0, x_1, x_2} disc(D \setminus a_0, Y^1 \setminus x_1, Y^2 \setminus x_2)$$

where $\epsilon(x_1, x_2)$ is either $+1$ or -1 . To be concrete, if we identify Y^1, Y^2 with the set $\{1, \dots, i\}$, then $\epsilon(j_1, j_2) = (-1)^{j_1+j_2}$. Thus, for each triple (D, Y^1, Y^2) we just need to loop over $n_2 n_3$ terms, and for each we need $O(n_{max})$ to find $D \setminus a_0, Y^1 \setminus x_1, Y^2 \setminus x_2$. \square

The recursion formula we need to evaluate is given in the following lemma.

Lemma 4.14. *Let M be a list of n matrices of size $n \times n$, with associated tripartite graph G . Let $(T, \alpha, \chi^1, \chi^2)$ be a tripartite decomposition of G . Let t be an internal node of T , let $T_t \subseteq T$ denote the subtree rooted in t , and let D, Y^1, Y^2 be such that*

$$\alpha(T_t) \setminus \alpha(t) \subseteq D \subseteq \alpha(T_t), \quad |D| = |Y^1| = |Y^2| \quad (4.17a)$$

$$\chi^1(T_t) \setminus \chi^1(t) \subseteq Y^1 \subseteq \chi^1(T_t), \quad \chi^2(T_t) \setminus \chi^2(t) \subseteq Y^2 \subseteq \chi^2(T_t) \quad (4.17b)$$

Let c_1, \dots, c_k be the children of t . Then

$$\begin{aligned} disc(D, Y^1, Y^2) = & \sum_{\mathcal{D}, \mathcal{Y}^1, \mathcal{Y}^2} \text{sgn}(\mathcal{Y}^1) \text{sgn}(\mathcal{Y}^2) disc(D_t, Y_t^1, Y_t^2) \\ & \prod_{j=1}^k (-1)^{|D_{tc_j}|} disc(D_{tc_j}, Y_{tc_j}^1, Y_{tc_j}^2) disc(D_{cc_j}, Y_{cc_j}^1, Y_{cc_j}^2) \end{aligned} \quad (4.18)$$

where $disc(\cdot, \cdot, \cdot)$ is as in (4.16), $\text{sgn}(\cdot, \cdot)$ as in Definition 4.4, and the sum is over all $\mathcal{D} = (D_t, D_{tc_1}, \dots)$, $\mathcal{Y}^1 = (Y_t^1, Y_{tc_1}^1, \dots)$, $\mathcal{Y}^2 = (Y_t^2, Y_{tc_1}^2, \dots)$ satisfying:

$$\begin{aligned} Z &= Z_t \sqcup (Z_{tc_1} \sqcup Z_{cc_1} \sqcup \dots \sqcup Z_{tc_k} \sqcup Z_{cc_k}) & \text{where } Z \in \{D, Y^1, Y^2\} \\ \zeta(T_{c_j}) \setminus \zeta(t) &\subseteq Z_{cc_j} \subseteq \zeta(T_{c_j}) \quad Z_t \subseteq \zeta(t) \quad Z_{tc_j} \subseteq \zeta(t) \cap \zeta(T_{c_j}) & \text{where } \zeta \in \{\alpha, \chi^1, \chi^2\}. \end{aligned}$$

Proof. Let $\pi^1 : D \rightarrow Y^1$ and $\pi^2 : D \rightarrow Y^2$ be matchings. Observe that Lemma 4.5 says that $\text{sgn}(\pi^1)$ will factor in the tree decomposition, leading to the term $\text{sgn}(\mathcal{D})\text{sgn}(\mathcal{Y}^1)$. Similarly, $\text{sgn}(\pi^2)$ leads to the term $\text{sgn}(\mathcal{D})\text{sgn}(\mathcal{Y}^2)$ (note that $\text{sgn}(\mathcal{D})$ cancels). Therefore, for the rest of the proof we can ignore all sign factors. We can think of the pair (π^1, π^2) as a subset of $D \times Y^1 \times Y^2$. In a similar way as we did in the proof of Lemma 4.10, there is a unique decomposition of (π^1, π^2) of the form

$$(\pi^1, \pi^2) = (\pi_t^1, \pi_t^2) \sqcup (\pi_{c_1}^1, \pi_{c_1}^2) \sqcup \dots \sqcup (\pi_{c_k}^1, \pi_{c_k}^2) \quad (4.19)$$

where $(\pi_t^1, \pi_t^2) \subseteq (\alpha \times \chi^1 \times \chi^2)(t)$, and for each child c we have that $(\pi_c^1, \pi_c^2) \subseteq (\alpha \times \chi^1 \times \chi^2)(T_c)$ and

$$(\pi_c^1, \pi_c^2) \cap (\alpha \times \chi^1 \times \chi^2)(t) = \emptyset. \quad (4.20)$$

Note that by construction π_t^1, π_t^2 have the same domain. Let D_t denote this domain, and let Y_t^1, Y_t^2 denote the respective ranges. Similarly, let D_c, Y_c^1, Y_c^2 be the domain and ranges of π_c^1, π_c^2 . Then we have

$$Z = Z_t \sqcup (Z_{c_1} \sqcup \dots \sqcup Z_{c_k}) \quad \text{where } Z \in \{D, Y^1, Y^2\} \quad (4.21a)$$

$$\zeta(T_c) \setminus \zeta(t) \subseteq Z_c \subseteq \zeta(T_c) \quad Z_t \subseteq \zeta(t) \quad \text{where } \zeta \in \{\alpha, \chi^1, \chi^2\} \quad (4.21b)$$

Thus, for matchings $(\pi^1, \pi^2) \in D \times Y^1 \times Y^2$ there is a unique partition as in (4.19) and the corresponding domains and ranges satisfy (4.21). On the other hand, assume that $D_t, D_c, Y_t^1, Y_c^1, Y_t^2, Y_c^2$ satisfy (4.13), and we are given some matchings $\pi_t^1, \pi_c^1, \pi_t^2, \pi_c^2$ with these domains and ranges and such that (π_c^1, π_c^2) satisfy (4.20). Then equations (4.21) tell us that we can merge them into matchings π^1, π^2 with domain D and ranges Y^1, Y^2 . Condition (4.20) ensures we are not overcounting. Then we have

$$\text{disc}(D, Y^1, Y^2) = \sum_{D_t, D_c, Y_t^1, Y_c^1, Y_t^2, Y_c^2} \text{disc}(D_t, Y_t^1, Y_t^2) \prod_{j=1}^k \text{disc}^*(D_{c_j}, Y_{c_j}^1, Y_{c_j}^2) \quad (4.22)$$

where the sum is over all triples as in (4.21), and where $\text{disc}^*(D_c, Y_c^1, Y_c^2)$ is similar to $\text{disc}(D_c, Y_c^1, Y_c^2)$, except that it only uses matchings (π_c^1, π_c^2) satisfying (4.20).

Finally, we can obtain $\text{disc}^*(D_c, Y_c^1, Y_c^2)$ using inclusion-exclusion in a similar way as in the proof of Lemma 4.9:

$$\text{disc}^*(D_c, Y_c^1, Y_c^2) = \sum_{\substack{D_{tc} \sqcup D_{cc} = D_c, \quad D_{tc} \subseteq \alpha(t) \\ Y_{tc}^1 \sqcup Y_{cc}^1 = Y_c^1, \quad Y_{tc}^1 \subseteq \chi^1(t) \\ Y_{tc}^2 \sqcup Y_{cc}^2 = Y_c^2, \quad Y_{tc}^2 \subseteq \chi^2(t)}} (-1)^{|D_{tc}|} \text{disc}(D_{tc}, Y_{tc}^1, Y_{tc}^2) \text{disc}(D_{cc}, Y_{cc}^1, Y_{cc}^2). \quad (4.23)$$

Combining equations (4.22) and (4.23), we obtain equation (4.18). \square

We proceed to the complexity analysis.

Theorem 4.15. *Let M be a list of matrices with associated tripartite graph G . Let $(T, \alpha, \chi^1, \chi^2)$ be a tripartite decomposition of G of width ω . Then we can compute $\text{Disc}(M)$ in $\tilde{O}(n^2 + n3^\omega)$.*

Proof. The proof is very similar to the one of Theorem 4.11. For each node $t \in T$, we compute $\text{disc}(D, Y^1, Y^2)$ for every triple D, Y^1, Y^2 that satisfies equation (4.17). We will show that for each t we can get $\text{disc}(D, Y^1, Y^2)$ for all such D, Y^1, Y^2 in $\tilde{O}((k_t + 1)(n + 3^\omega))$, where k_t is the number of children of t .

The base case is when t is a leaf of T . Let M_0 be the subarray of M given by indices $(\alpha(t), \chi^1(t), \chi^2(t))$. Then we need to find the mixed discriminant of all subarrays of M_0 . We can do this in $\tilde{O}(2^\omega)$ using of Lemma 4.13.

Assume now that t is an internal node of T with k_t children and let D, Y^1, Y^2 that satisfy (4.17). Then equation (4.18) tells us how to find $\text{disc}(D, Y^1, Y^2)$. Similarly as in Lemma 4.7, we can evaluate this formula in $\tilde{O}(k_t(n + 3^\omega))$, assuming we know all terms in the recursion. We already found $\text{disc}(D_{cc}, Y_{cc}^1, Y_{cc}^2)$ for all children in the recursion, and we can find $\text{disc}(D_t, Y_t^1, Y_t^2)$ for all D_t, Y_t^1, Y_t^2 in $\tilde{O}(2^\omega)$ in the same way as for the base case. This leads to a bound of $\tilde{O}((k_t + 1)(n + 3^\omega))$ to compute $\text{disc}(D, Y^1, Y^2)$ for all D, Y^1, Y^2 . \square

4.5.2 Higher dimensions

It is easy to see that our methods extend to compute generalizations of the permanent and determinant in higher dimensions. We consider a square $(d + 1)$ -dimensional array (or tensor) M of length n , i.e., of size $n \times \cdots \times n$ ($d + 1$ times). Here we assume d to be constant. Let's index the first coordinate of M with a set A , and the following coordinates with sets X^1, \dots, X^d . Consider a function F of the form

$$F(M) = \sum_{\pi_1, \dots, \pi_d} \prod_{a \in A} \epsilon_1(\pi_1) \cdots \epsilon_d(\pi_d) M_{a, \pi_1(a), \dots, \pi_d(a)} \quad (4.24)$$

where the sum is over all bijections $\pi_l : A \rightarrow X^l$, and where $\epsilon_l(\pi_l)$ is either 1 or $\text{sgn}(\pi_l)$.

Let's consider two special cases of equation (4.24). The simplest case is when $\epsilon_l(\pi_l) = 1$ for all l . We refer to such function as the $(d + 1)$ -dimensional permanent and we denote it as $\text{Perm}(M)$ [55]. Some applications of this permanent are shown in [6, 132].

Consider now the case when $d + 1$ is even and $\epsilon_l(\pi_l) = \text{sgn}(\pi_l)$ for all l . This is perhaps

the simplest generalization of the determinant, and it is usually referred to as the first Cayley hyperdeterminant [32]. Some applications of the hyperdeterminant are shown in [9, 94]. As opposed to the 2-dimensional case, computing the hyperdeterminant is #P-hard [72].

We now proceed to extend our decomposition methods to this setting. We associate a $(d+1)$ -partite graph G where for each nonzero entry of M we put a $(d+1)$ -clique in the respective coordinates. A tree decomposition of G can be seen as a tuple $(T, \alpha, \chi^1, \dots, \chi^d)$. The width ω of the decomposition is the largest of $|\alpha(t)| + |\chi^1(t)| + \dots + |\chi^d(t)|$ among all nodes t .

As before, for some sets $D \subseteq A, Y^l \subseteq X^l$, we consider the function

$$f(D, Y^1, \dots, Y^d) := \sum_{\pi_1, \dots, \pi_d} \prod_{a \in A} \epsilon_1(\pi_1) \cdots \epsilon_d(\pi_d) M_{a, \pi_1(a), \dots, \pi_d(a)} \quad (4.25)$$

where the sum is over all bijections $\pi_l : D \rightarrow Y^l$.

There are two steps in order to generalize our results to this setting: evaluate f in all subarrays, and evaluate the recursion formula. For the former, the approach from Lemma 4.1 (and Lemma 4.13) has a simple generalization. Indeed, Barvinok shows this for the case of the hyperdeterminant [9]. The proof is the same for an arbitrary function F as in (4.24). Thus, we have the following.

Proposition 4.16 ([9]). *Let M_0 be a $(d+1)$ -dimensional array of size $n_0 \times \dots \times n_d$, and let f be as in (4.25). Let A_0, X_0^1, \dots, X_0^d be its set of coordinates, and let*

$$S := \{(D, Y^1, \dots, Y^d) \subseteq A_0 \times X_0^1 \times \dots \times X_0^d : |D| = |Y^1| = \dots = |Y^d|\}$$

We can compute $f(D, Y^1, \dots, Y^d)$ for all tuples in S in $O(n_{max}^{d+1} 2^{n_0 + \dots + n_d})$, where $n_{max} = \max\{n_0, \dots, n_d\}$.

Repeating the same analysis as in the proof of Lemma 4.14, the recursion formula is:

$$f(D, Y^1, \dots, Y^d) = \sum_{\mathcal{D}, \mathcal{Y}^1, \dots, \mathcal{Y}^d} \delta_1(\mathcal{D}, \mathcal{Y}^1) \cdots \delta_d(\mathcal{D}, \mathcal{Y}^d) f(D_t, Y_t^1, \dots, Y_t^d) \prod_{j=1}^k (-1)^{|D_{tc_j}|} f(D_{tc_j}, Y_{tc_j}^1, \dots, Y_{tc_j}^d) f(D_{cc_j}, Y_{cc_j}^1, \dots, Y_{cc_j}^d) \quad (4.26)$$

where $\delta_l(\mathcal{D}, \mathcal{Y}^l)$ is either 1 or $\text{sgn}(\mathcal{D})\text{sgn}(\mathcal{Y}^l)$ depending on ϵ_l , and the sum is over all tuples $\mathcal{D}, \mathcal{Y}^1, \dots, \mathcal{Y}^d$ such that

$$\begin{aligned} Z &= Z_t \sqcup (Z_{tc_1} \sqcup Z_{cc_1} \sqcup \dots \sqcup Z_{tc_k} \sqcup Z_{cc_k}) & \text{where } Z \in \{D, Y^1, \dots, Y^d\} \\ \zeta(T_{c_j}) \setminus \zeta(t) &\subseteq Z_{cc_j} \subseteq \zeta(T_{c_j}) \quad Z_t \subseteq \zeta(t) \quad Z_{tc_j} \subseteq \zeta(t) \cap \zeta(t_{c_j}) & \text{where } \zeta \in \{\alpha, \chi^1, \dots, \chi^d\} \end{aligned}$$

The complexity of the decomposition algorithm is as follows.

Theorem 4.17. *Let M be a square $(d+1)$ -dimensional array of length n , with $(d+1)$ -partite graph G . Let F be a generalized determinant/permanent as in (4.24). Let $(T, \alpha, \chi^1, \dots, \chi^d)$ be a tree decomposition of G of width ω . Then we can compute $F(M)$ in $\tilde{O}(n^2 + n3^\omega)$.*

Proof. The proof is very similar to past ones. For each node t , we compute $f(D, Y^1, \dots, Y^d)$ for all valid tuples. We show that for each t we can do this in $\tilde{O}((k_t + 1)(n + 3^\omega))$, where k_t is the number of children of t .

The base case, i.e., leaf nodes, reduces to Proposition 4.16, leading to a bound of $\tilde{O}(2^\omega)$.

For an internal node t , equation (4.26) tells us how to find $f(D, Y^1, \dots, Y^d)$. Similarly as in Lemma 4.7, we can evaluate this formula in $O(k_t(n + 3^\omega))$. \square

For the special case of the permanent, we can give a better bound.

Theorem 4.18. *Let M be a square $(d+1)$ -dimensional array of length n , with $(d+1)$ -partite graph G . Let $(T, \alpha, \chi^1, \dots, \chi^d)$ be a tree decomposition of G of width ω . Then we can compute $\text{Perm}(M)$ in $\tilde{O}(n2^\omega)$.*

Proof. If there are no sign factors we can follow the procedure of Lemma 4.3 for the recursion, leading to a bound of $\tilde{O}((k_t + 1)2^\omega)$ per node. \square

4.6 Mixed volume of zonotopes

The mixed volume MVol of n convex bodies K_1, \dots, K_n in \mathbb{R}^n is the unique real function that satisfies the following properties.

- MVol is multilinear and symmetric in its arguments.
- $\text{MVol}(K, \dots, K) = n! \text{vol}(K)$, where vol denotes the volume.

Alternatively, it can be shown that the function $f(\lambda) := \text{vol}(\sum_i \lambda_i K_i)$ for $\lambda_i \geq 0$, is a homogeneous polynomial, and MVol is the coefficient of $\lambda_1 \cdots \lambda_n$. For more information about mixed volumes, see e.g., [120]. We focus here in the case that all bodies K_i are zonotopes, which are a special class of polytopes.

4.6.1 Mixed volumes and permanents

Definition 4.5. A *zonotope* z is a polytope that is a Minkowski sum of line segments, i.e.,

$$z = [0, 1]z_1 + [0, 1]z_2 + \cdots + [0, 1]z_m = \{r_1 z_1 + \cdots + r_m z_m : 0 \leq r_i \leq 1\}$$

where $z_i \in \mathbb{R}^n$ are vectors. In case z_1, \dots, z_m are linearly independent, we say that z is a *parallelotope*.

The mixed volume of zonotopes has a simple description as follows.

Proposition 4.19. Let $z^i = \sum_{j \in J_i} [0, 1]z_j^i$ be a zonotope, for $i = 1, \dots, n$. Then

$$\text{MVol}(z^1, \dots, z^n) = \sum_{j_1 \in J_1, \dots, j_n \in J_n} |\text{Det}(z_{j_1}^1, z_{j_2}^2, \dots, z_{j_n}^n)| \quad (4.27)$$

Proof. The multilinearity of the mixed volume implies

$$\text{MVol}(z^1, \dots, z^n) = \sum_{j_1 \in J_1, \dots, j_n \in J_n} \text{MVol}([0, 1]z_{j_1}^1, \dots, [0, 1]z_{j_n}^n).$$

Thus, we just need to argue that

$$\text{MVol}([0, 1]z_{j_1}^1, \dots, [0, 1]z_{j_n}^n) = |\text{Det}(z_{j_1}^1, \dots, z_{j_n}^n)|$$

which follows by noting that $\sum_i [0, 1]\lambda_i z_{j_i}^i$ is a parallelepiped with sides $\lambda_i z_{j_i}^i$, and thus its volume is given by the absolute volume of the determinant. \square

The mixed volume of n parallelotopes reduces to a permanent when their main axes are aligned, as shown now.

Corollary 4.20. *Let $u_1, \dots, u_n \in \mathbb{R}^n$ and let $M \in \mathbb{R}_{\geq 0}^{n \times n}$ be a nonnegative matrix. Let $z^i = \sum_j [0, 1] M_{i,j} u_j$ be a zonotope. Then*

$$\text{MVol}(z^1, \dots, z^n) = |\text{Det}(u_1, \dots, u_n)| \text{Perm}(M).$$

Proof. We just need to use equation (4.27), and cancel out all terms that contain a repeated vector u_j , as the determinant is zero. The remaining terms have $|\text{Det}(u_1, \dots, u_n)|$ as a factor and we get the desired formula. \square

4.6.2 Graph representation

To use a decomposition method for mixed volumes we need to have a graph description of the zonotopes. We consider now two different graphs that can be associated to a set of zonotopes, and more generally to polytopes. The first one is a bipartite graph that can be thought of as the analogue of the bipartite graph of a matrix. The second one has to do with the sparsity in the standard basis representation and it is a more intuitive notion for general polytopes.

Definition 4.6. Let Q be a set of n polytopes in \mathbb{R}^n . Let U denote the set of all vectors (up to scaling) that are parallel to some edge in Q . We refer to U as the *edge directions* of Q . The *edge graph* $G(Q)$ is a bipartite graph with vertices $Q \cup U$ and edges (q, u) if q contains an edge parallel to u .

Definition 4.7. Let Q be a set of n polytopes in \mathbb{R}^n . Let $X = \{x_1, \dots, x_n\}$ denote the coordinates. The *coordinates graph* $G^X(Q)$ has X as vertex set, and for each polytope $q \in Q$ we form a clique in all its non constant components. Note that if $q = \sum_j [0, 1] z_j$ is a zonotope, we form a clique in the nonzero coordinates of $\sum_j z_j$.

Remark 4.6. Note that the edge graph $G(Q)$ is invariant under affine transformations of Q , whereas the coordinates graph $G^X(Q)$ is not.

Example 4.10 (Zonotopes of bounded treewidth). Consider the following zonotopes:

$$z^1 = [0, 1](a_1 e_n + e_1) + [0, 1]e_1 \tag{4.28a}$$

$$z^i = [0, 1](a_i e_n + e_i - e_{i-1}) + [0, 1](e_i - e_{i-1}) \quad \text{for } i = 2, \dots, n-1 \tag{4.28b}$$

$$z^n = [0, 1](a_n e_n - e_{n-1}) \tag{4.28c}$$

where $\{e_i\}_i$ is the canonical basis and $a_i \in \mathbb{Z}$ are some integers.

Note that the segments of the above zonotopes are all nonparallel (there are $2n + 1$ edge directions). This means that the edge graph G has n connected components, one for each of the zonotopes, and each component is either a 2-path or a 1-path. Thus, G has treewidth 1. As for the coordinates graph G^X , it is the union of the triangles: $X_i := \{x_i, x_{i+1}, x_n\}$. It is easy to see that G^X has treewidth 2.

The following example shows the relationship between these graphs and the matrix graphs we used before.

Example 4.11 (Relationship with matrix graphs). Let $z^i = \sum_j [0, 1] M_{i,j} u_j$ be zonotopes as in Corollary 4.20. The edge graph G of the zonotopes has vertices $Z \cup U$ where $Z = \{z^i\}_i$ and $U = \{u_j\}_j$, and it has an edge (z^i, u_j) whenever $M_{i,j} \neq 0$. If we replace Z with the row set and U with the column set, this is precisely the bipartite graph of matrix M .

On the other hand, the coordinates graph G^X depends on the sparsity structure of vectors U . Assume now that $U = \{e_j\}_j$ is the canonical basis. Then G^X has an edge (x_j, x_k) whenever there is some z^i with $M_{i,j} \neq 0$ and $M_{i,k} \neq 0$. This corresponds to the column graph of M .

Because of the above example it is expected that the tree decomposition methods derived for the permanent should allow to compute mixed volumes of certain families of zonotopes. Indeed, we now show if *there are few edge directions* and the edge graph G has bounded treewidth then the mixed volume can be computed efficiently.

Theorem 4.21. *Let Z be a set of n zonotopes in \mathbb{R}^n . Let U be the set of edge directions of Z , and assume that $d := |U| - n$ is constant. Let G denote the edge graph of Z . Given a tree decomposition of G of width ω , we can compute $\text{MVol}(Z)$ in $\tilde{O}(n^{d+3} + n^{d+1} 2^\omega)$.*

Proof. If $|U| < n$ it follows from (4.27) that $\text{MVol}(Z) = 0$. If $|U| = n$, then Corollary 4.20 tells us that we just need to compute the determinant of the u_i 's and the permanent of some matrix M . We can find the determinant in $O(n^3)$ with linear algebra. For the permanent, as the edge graph G corresponds to the bipartite graph of M , we can use Theorem 4.11. Thus, we can find this permanent in $\tilde{O}(n 2^\omega)$.

If $|U| > n$, let $W \subseteq U$ of size n . For each $z \in Z$ assume $z = \sum_{u \in U} [0, 1] c_u u$ for some scalars c_u . Let $z_W = \sum_{u \in W} [0, 1] c_u u$, and let $Z_W = \{z_W : z \in Z\}$. It follows from equation (4.27)

that

$$\text{MVol}(Z) = \sum_{W \subseteq U, |W|=n} \text{MVol}(Z_W).$$

For each such W the associated graph is a subgraph of G , so we can compute $\text{MVol}(Z_W)$ in $\tilde{O}(n^3 + n2^\omega)$. As there are $\binom{n+d}{n} = O(n^d)$ possible W , the result follows. \square

Example 4.12 (Zonotopes with few edge directions). Consider the following zonotopes:

$$z^i = [0, 1]a_i e_i + [0, 1]b_i e \quad \text{for } i = 1, \dots, n$$

where $\{e_i\}_i$ is the canonical basis, $e := \sum_j e_j = (1, \dots, 1)$ and $a_i, b_i \in \mathbb{Z}$ are some integers. Observe that there are $n + 1$ edge directions: e_1, \dots, e_n, e . Also note that the edge graph G is a tree, consisting of pairs (z^i, e_i) and (z^i, e) . Therefore, Theorem 4.21 says that can compute the mixed volume of these polytopes in polynomial time.

4.6.3 Hardness result

Theorem 4.21 shows that it is possible to exploit tree decompositions for mixed volume computations. However, it restricts the zonotopes to have a small number of edge directions. This is a strong requirement which is not satisfied in many cases (such as in Example 4.10). Unfortunately, we will see that we need this condition. We remark that the same condition appears in discrete optimization, where it allows to derive (strongly) polynomial time algorithms for certain discrete convex optimization problems [103].

We now show that computing the mixed volume of the zonotopes in Example 4.10 is $\#P$ -hard. This shows that mixed volumes of zonotopes continue to be hard, even if both G and G^X have bounded treewidth. We use a similar reduction as in [56], where they prove that the volume of zonotopes is $\#P$ -hard.

Lemma 4.22. *The determinant of the following $n \times n$ matrix is $s_1 + s_2 + \dots + s_n$.*

$$M = \begin{pmatrix} 1 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \\ s_1 & s_2 & s_3 & s_4 & \dots & s_{n-1} & s_n \end{pmatrix}$$

Proof. If we perform Gaussian elimination we end up with an upper triangular matrix where the diagonal is: $M_{i,i} = 1$ for $i < n$ and $M_{n,n} = s_1 + \dots + s_n$. \square

Proposition 4.23. *The following problem is #P-hard. Given integers a_1, \dots, a_n , compute the mixed volume of the n zonotopes of equations (4.28).*

Proof. We consider the #P-complete problem Subset-Sum: given a set of integers A , determine the number of subsets $S \subseteq A$ with sum zero. Let $k = n - 1$, $A = \{a_1, \dots, a_k\}$ and let $a_n = \delta$ be a parameter. We will show that the solution to the Subset-Sum problem is given by

$$\frac{1}{2} \text{MVol}(z^1, \dots, z_{\delta=-1}^n) - \text{MVol}(z^1, \dots, z_{\delta=0}^n) + \frac{1}{2} \text{MVol}(z^1, \dots, z_{\delta=1}^n). \quad (4.29)$$

Let's evaluate equation (4.27). Consider the following $n \times (2n - 1)$ matrix.

$$M_\delta = \begin{pmatrix} 1 & 1 & -1 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & -1 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & \dots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 1 & -1 \\ a_1 & 0 & a_2 & 0 & a_3 & 0 & \dots & a_{k-1} & 0 & a_k & 0 & \delta \end{pmatrix}$$

Observe that columns $2i - 1$ and $2i$ of M_δ correspond to z^i , and the last column corresponds to z_δ^n . Then formula (4.27) considers submatrices of M_δ that use columns j_1, \dots, j_n where $j_i \in \{2i - 1, 2i\}$ for $i = 1, \dots, k$ and $j_n = 2n - 1$. Note now that for any subset $S \subseteq A$, there is a natural submatrix M_δ^S to consider: if $a_i \in S$ then $j_i = 2i - 1$ and otherwise $j_i = 2i$. This correspondence is a bijection. Observe also that each submatrix M_δ^S has the form of Lemma 4.22. Thus, we have the following equation:

$$\text{MVol}(z^1, \dots, z_\delta^n) = \sum_{S \subseteq A} \left| \delta + \sum_{a_i \in S} a_i \right|.$$

Finally, observe that for any integer s we have

$$\frac{1}{2} |(-1) + s| - |s| + \frac{1}{2} |1 + s| = \begin{cases} 1 & \text{if } s = 0 \\ 0 & \text{otherwise} \end{cases}$$

The last two equations imply that (4.29) indeed counts the subsets S with sum zero. \square

4.6.4 Application to sparse polynomial systems

Sparse polynomial systems are closely connected to lattice polytopes. In particular, the generic number of solutions of sparse systems is given by a mixed volume. Consequently, the results from this section have natural implications about the complexity of solving sparse systems. We now recall Bernstein's Theorem [18] (see also [47, §7.5]), also known as the BKK bound, which establishes the relationship between sparse systems and mixed volumes.

Definition 4.8. Let $f = \sum_{\alpha} c_{\alpha} x^{\alpha} \in \mathbb{C}[X]$ be a multivariate polynomial in $X = (x_1, \dots, x_n)$. The *Newton polytope* of f is the convex hull of the set of exponents α , considered as vectors in \mathbb{R}^n .

Theorem 4.24 (Bernstein [18]). *Let $f_1, \dots, f_n \in \mathbb{C}[X]$ be sparse polynomial equations in variables $X = (x_1, \dots, x_n)$, and let $Q_1, \dots, Q_n \subseteq \mathbb{R}^n$ be their Newton polytopes. If the number of common zeros of these equations in $(\mathbb{C}^*)^n$ is finite, then it is upper bounded by $\text{MVol}(Q_1, \dots, Q_n)$. Moreover, the bound is achieved when the coefficients of the equations are generic.*

Bernstein's theorem allows us to translate the results of this section to sparse polynomial systems. In particular, Theorem 4.21 implies that for certain polynomial systems, whose Newton polytopes have few edge directions, we can efficiently determine the number of solutions.

Example 4.13. Consider the system of equations:

$$0 = c_{i,1} + c_{i,2} x_i^{a_i} + c_{i,3} \prod_j x_j^{b_i} + c_{i,4} x_i^{a_i} \prod_j x_j^{b_i} \quad \text{for } i = 1, \dots, n.$$

Since the Newton polytopes of these equations are the zonotopes from Example 4.12, then we can efficiently compute the (generic) number of solutions of this system.

We can also derive a hardness result on the complexity of solving generic polynomial systems of small treewidth. Let $F = \{f_1, \dots, f_n\} \subseteq \mathbb{C}[X]$ be a sparse polynomial system. Recall from Chapter 3 that we can describe the sparsity structure of the system with a graph $G^X(F)$ with vertex set X . It is easy to see that this graph $G^X(F)$ agrees with the coordinates graph $G^X(Q)$ of their Newton polytopes. The following is a consequence of Proposition 4.23.

Corollary 4.25 (Hardness of solving generic sparse systems). *The following problem is #P-hard: count the number of solutions of a generic sparse polynomial system of treewidth two.*

Part II

Polynomial optimization

Chapter 5

Sampling algebraic varieties for SOS optimization

In this chapter we study sum of squares (SOS) relaxations to optimize polynomial functions over the real trace of an algebraic variety \mathbf{V} . We propose a new methodology that, rather than relying on some algebraic description, represents \mathbf{V} with a generic set of complex samples. Our methods leverage the coordinate ring structure to reduce the size of the corresponding semidefinite program (SDP). The content of this chapter is based on [42].

5.1 Introduction

Consider the ring $\mathbb{R}[x] := \mathbb{R}[x_1, \dots, x_n]$ of multivariate polynomials and an algebraic variety $\mathbf{V} \subseteq \mathbb{C}^n$. For a given polynomial $p \in \mathbb{R}[x]$, we are interested in deciding whether

$$p(x) \geq 0 \text{ for all } x \in \mathbf{V} \cap \mathbb{R}^n. \quad (5.1)$$

More generally, we can consider the problem of finding lower bounds for a polynomial on a real variety.

The decision problem in (5.1) is computationally hard. As mentioned in Section 2.1.3, there are tractable relaxations based on the SOS method [105]. Recall that a polynomial $F \in \mathbb{R}[x]$ is SOS if it can be written in the form $F(x) = \sum_i f_i(x)^2$ for some $f_i \in \mathbb{R}[x]$. Given a bound $d \in \mathbb{N}$, a sufficient condition for (5.1) to hold is the existence of a polynomial $F \in \mathbb{R}[x]$

such that

$$p(z) = F(z) \text{ for all } z \in \mathbf{V} \quad (\text{i.e., } p \equiv F \bmod \mathbf{I}(\mathbf{V})); \quad F(x) \text{ is SOS;} \quad \deg(F) \leq 2d. \quad (5.2)$$

We refer to such an F as a d -SOS(\mathbf{V}) *certificate*. The main problem we address in this chapter is the following.

Problem. *Given a bound $d \in \mathbb{N}$, a polynomial $p(x)$ and a variety \mathbf{V} , find a d -SOS(\mathbf{V}) certificate (if it exists).*

It was shown in [104] that, given a Gröbner basis of the ideal $\mathbf{I}(\mathbf{V})$, the above problem reduces to a semidefinite program (SDP). To the best of our knowledge, this is the only known method to address it. This approach is quite effective for varieties with simple Gröbner bases, such as the hypercube $\{0, 1\}^n$, or hypersurfaces. Unfortunately, besides these simple cases, Gröbner bases computation is typically too expensive.

Given defining equations of the variety $\{h_j(x) = 0\}_j$, there is a *weaker* class of certificates based on writing $p(x)$ in the form $F(x) + \sum_j g_j(x)h_j(x)$; see (2.4). This approach is widely used in practice [22, 85], thanks to the convenience of allowing any set of defining equations. But this simplicity comes with a price, since the success of the relaxation now depends on the choice of a good set of equations $\{h_j\}_j$. Furthermore, the corresponding SDP is larger. Indeed, for fixed \mathbf{V} the number of unknowns is $O(d^{2n})$, whereas for (5.2) is $O(d^{2 \dim \mathbf{V}})$; see Remark 5.3. We also point out that for several parametric varieties, notably secant varieties [84], the defining equations are not explicitly known, thus making this type of certificates unfeasible.

Sampling certificates In this chapter we propose an alternative geometric approach to compute SOS(\mathbf{V}) certificates. Rather than depending on an algebraic description of the variety, we rely on a generic set of samples $Z := \{z_1, \dots, z_S\} \subseteq \mathbf{V}$. By specializing the condition in (5.2) to such samples, we get the following.

Definition 5.1. Let $\mathbf{V} \subseteq \mathbb{C}^n$ be a variety and let $p \in \mathbb{R}[x]$ be nonnegative on $\mathbf{V} \cap \mathbb{R}^n$. Given a bound $d \in \mathbb{N}$, a *sampling d -SOS pre-certificate* is a pair (F, Z) , where $F(x)$ is a polynomial

and $Z = \{z_1, \dots, z_S\} \subseteq \mathbf{V}$ a sample set, such that

$$p(z_s) = F(z_s) \text{ for } s = 1, \dots, S; \quad F(x) \text{ is SOS}; \quad \deg(F) \leq 2d; \quad (5.3)$$

The pre-certificate is *correct* if F is a d -SOS(\mathbf{V}) certificate, i.e., it satisfies (5.2).

Computing a sampling pre-certificate reduces to an SDP. We show that suitable genericity assumptions guarantee its correctness, thus giving us an SOS(\mathbf{V}) certificate. An interesting feature of our sampling methodology is that the only information needed about the variety is a *sampling oracle*, i.e., a procedure that generates generic samples. Note that sampling points is very simple when the variety has a known parametrization (e.g., for $SO(n)$, Grassmannians, rank k tensors). For a general variety \mathbf{V} , the field of *numerical algebraic geometry* (see Section 2.1.2) provides practical methods to sample generic points.

Contributions

This chapter presents the following contributions.

- We introduce a new methodology to compute SOS certificates over an algebraic variety \mathbf{V} . This is a geometric formulation that represents \mathbf{V} with a generic set of complex samples, instead of relying on some algebraic description. In this way, we avoid algebraic issues such as multiplicity and the dependence on the specific generators used. We analyze the correctness of our formulation, establishing sufficient conditions on the samples and the variety.
- Our methodology takes advantage of the coordinate ring structure to simplify the SDP. Moreover, it is the first such relaxation independent of Gröbner bases. This makes our methods appealing for many varieties that are easy to sample from but for which Gröbner bases computation is intractable. Examples of such varieties include $SO(n)$, Stiefel manifolds, Grassmannians and secant varieties.
- We apply for the first time techniques from numerical algebraic geometry to SOS programs. In this way, we inherit some of the main strengths from this area. We highlight that these methods are trivially parallelizable, since they rely on homotopy continuation

of many independent paths. They also allow us to work with straight-line programs (i.e., polynomials do not need to be expanded).

Related work

Our sampling SOS methodology extends the ideas from Löfberg and Parrilo in [92], where they first consider sampling formulations for unconstrained SOS problems. They show that sampling formulations offer some numerical advantages over the standard approach. Most remarkably, the SDP has a low rank structure, which leads to a significant complexity improvement in interior point methods. In particular, low rank structure is exploited in the solvers SDPT3 and DSDP [17, 133]. Secondly, the SDP is usually better conditioned, as it relies on a set of orthogonal polynomials instead of a monomial basis. These properties make sampling formulations appealing, as seen in [91, 109, 110]. We will see that these properties are preserved in the variety case considered in this chapter. We remark that our use of the samples differs from [92] in that for us samples carry additional information about the underlying variety \mathbf{V} .

Different methods have been proposed to reduce complexity in SOS programs, in particular by exploiting symmetries, sparsity, and quotient ring structure; see [22, §3.3], [86, §8] and the references therein. This chapter is only concerned with the last item, but we point out that all these techniques can be combined together. The Gröbner bases method to compute quotient ring SOS certificates was introduced in [104]; some further improvements were made in [106]. This is the default method for several varieties with simple Gröbner bases, particularly from combinatorial optimization [86, §8]. Quotient ring methods have also been used for unconstrained optimization [101]. We point out that there was no “direct” method (without computing the radical \sqrt{I}) to obtain SOS certificates on the coordinate ring (i.e., $\text{SOS}(\mathbf{V})$ certificates).

Although the existence (or degree bounds) of SOS certificates is beyond the scope of this chapter (see e.g., [118]), we review some known results for completeness. In particular, $\text{SOS}(\mathbf{V})$ certificates exist if: (i) \mathbf{V} is zero-dimensional, (ii) \mathbf{V} is one-dimensional and p is both strictly positive and bounded [122], (iii) \mathbf{V} is compact and p is strictly positive [119], (iv) \mathbf{V} is a variety of minimal degree and p is quadratic [23]. For most varieties there exist nonnegative

polynomials which are not SOS. Nonetheless, such instances can always be approximated by SOS polynomials (possibly of higher degree) [85, §2.6].

Solution outline

Our approach to compute d -SOS(\mathbf{V}) certificates follows three main steps.

- (i) *Sampling*: Obtain a “good” set of samples Z on the variety. It will be sufficient for us to consider generic (random) samples on each component of the variety.
- (ii) *SDP*: Given a sample set Z , compute a sampling pre-certificate (F, Z) using an SDP.
- (iii) *Verification*: Check that the pre-certificate (F, Z) is correct. This reduces to the identity testing problem.

The structure of this chapter is as follows. Section 5.2 presents some basic algebraic preliminaries. Afterwards, we approach each of the problems from above, although in a different order to simplify the exposition. Section 5.3 deals with (ii), Section 5.4 with (iii), and Section 5.5 with (i). Section 5.6 presents the complete sampling SOS methodology. Finally, Section 5.7 shows several examples to illustrate our methods.

5.2 Preliminaries

5.2.1 Algebraic geometry

Let \mathbb{K} denote a field which is either \mathbb{R} or \mathbb{C} , and let $\mathbb{K}[x] = \mathbb{K}[x_1, \dots, x_n]$ denote the ring of polynomials with coefficients in \mathbb{K} . Let $I \subseteq \mathbb{K}[x]$ be a polynomial ideal, and let $\mathbf{V} = \mathbf{V}_{\mathbb{C}}(I)$ be the associated complex variety. The *quotient ring* $\mathbb{K}[x]/I$ is the set of equivalence classes where $f \sim_I g$ if $f - g \in I$. The *coordinate ring* of \mathbf{V} is the quotient ring $\mathbb{K}[\mathbf{V}] := \mathbb{K}[x]/\sqrt{I}$. Equivalently, $\mathbb{K}[\mathbf{V}]$ is the set of equivalence classes of polynomials where $f \sim_{\mathbf{V}} g$ if they define the same function on \mathbf{V} .

In this chapter we only consider complex varieties defined by real polynomials. It is easy to see that a complex variety \mathbf{V} can be defined by real polynomials if and only if it is *self-conjugate*, i.e. its complex conjugate $\overline{\mathbf{V}}$ is itself. Recall that any variety can be decomposed

into irreducible components \mathbf{V}_i as in (2.1). If \mathbf{V} is self-conjugate, then either \mathbf{V}_i is also self-conjugate, or there is a pair $(\mathbf{V}_i, \mathbf{V}_j)$ of conjugate components.

5.2.2 Sampling varieties

Our technique requires a sampling oracle for the complex variety \mathbf{V} . More precisely, we need to sample generic (random) points in each irreducible component of \mathbf{V} . Observe that sampling points is easy whenever we have a parametrization. For instance, we can sample points from $SO(n)$ using the *Cayley parametrization*:

$$A \mapsto (\text{id}_n - A)(\text{id}_n + A)^{-1}, \quad \text{for skew symmetric } A. \quad (5.4)$$

Other parametric varieties include Grassmannians, Stiefel manifolds, and secant varieties.

For a general variety \mathbf{V} , a practical way to compute sample points is through the tools of numerical algebraic geometry; we refer to [14, 124] for an introduction. Homotopy continuation tools such as Bertini [13] and PHCpack [137] allow to compute the irreducible decomposition of \mathbf{V} , and afterwards to sample an arbitrary number of points in any component. Typically the most expensive part is to produce the decomposition; sampling points is relatively cheap. These numerical methods offer the following advantages with respect to symbolic methods such as Gröbner bases: they are trivially parallelizable (each path can be tracked independently), they allow for straight-line programs (polynomials do not need to be expanded), and they offer better numerical stability.

Remark 5.1 (Complex samples). Even though we are only interested in real polynomials, our methods allow the sample points to be complex. This is an important feature, since computing real points on a variety is significantly harder than computing complex points.

Remark 5.2 (Zero-dimensional case). The results from this chapter are most useful for positive-dimensional varieties, particularly if the number of components is relatively small. The reason is that we treat each irreducible component separately. In particular, we take care of the zero-dimensional part of the variety exhaustively, i.e., we check for all such points that $p(x)$ is indeed nonnegative. If the whole variety is zero-dimensional our algorithm reduces to a brute-force search.

5.2.3 SOS certificates on varieties

Consider a variety \mathbf{V} defined by equations $h = \{h_j\}_j$, and let $I = \langle h \rangle$ be the generated ideal. There are two traditional SOS methods to certify nonnegativity on $\mathbf{V} \cap \mathbb{R}^n$. An *equations d -SOS* certificate is a tuple of polynomials (F, g_1, \dots, g_m) such that

$$p(x) = F(x) + \sum_j g_j(x)h_j(x); \quad F(x) \text{ is SOS}; \quad \deg(F), \deg(g_j h_j) \leq 2d. \quad (5.5)$$

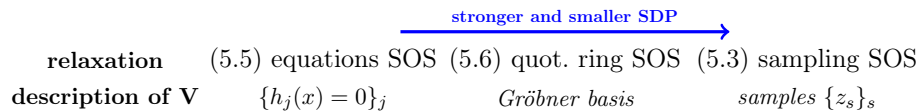
Finding such a certificate reduces to an SDP [105]. A *quotient ring d -SOS* certificate is a polynomial F such that

$$p - F \in I \quad (\text{i.e., } p \equiv F \bmod I); \quad F(x) \text{ is SOS}; \quad \deg(F) \leq 2d. \quad (5.6)$$

Given a Gröbner basis of I , the above reduces to an SDP [22, §3.3.5]. For an introduction to Gröbner bases and quotient ring computations we refer to [46].

Quotient ring formulations are appealing for two main reasons. Firstly, they are *stronger* than equations SOS (i.e., if (5.5) is feasible then so is (5.6), but the converse is not true). And secondly, the size of the associated SDP is *smaller*, not only because of the absence of the equations g_j , but since it also takes into account the structure of the quotient ring. Consequently, quotient ring SOS has become the default approach for varieties with simple Gröbner bases (e.g., the hypercube $\{0, 1\}^n$). However, the expense of Gröbner bases computation limits its application to further cases.

Our sampling SOS methodology can be seen as a “better” quotient ring formulation. The reason being that we work modulo the *radical* ideal $\sqrt{I} = \mathbf{I}(\mathbf{V})$, and thus the underlying space is the coordinate ring. The following diagram illustrates the relations among these three types of certificates.



Remark 5.3 (Hilbert function). For $k \in \mathbb{N}$, let $\mathcal{L}_k \subseteq \mathbb{R}[\mathbf{V}]$ be the linear space spanned by the polynomials of degree at most k . The function $H_{\mathbf{V}}(k) := \dim(\mathcal{L}_k)$ is known as the *Hilbert function*, and it plays an important role in sampling SOS (also in quotient ring SOS). Indeed,

the size of the PSD matrix in the SDP is precisely $H_{\mathbf{V}}(d)$. We will also see in Section 5.5 that the number of samples we require is given by $H_{\mathbf{V}}(2d)$. The Hilbert function can be bounded as follows [34]:

$$H_{\mathbf{V}}(k) \leq \binom{n+k}{k}, \quad \text{and} \quad H_{\mathbf{V}}(k) \leq \deg \mathbf{V} \binom{\dim \mathbf{V} + k}{k} \text{ if } \mathbf{V} \text{ is equidimensional,} \quad (5.7)$$

where \deg, \dim denote the degree and dimension. The second bound implies that, for fixed \mathbf{V} , the size of the PSD matrix in sampling SOS is $O(d^{\dim \mathbf{V}})$. In contrast, for equations SOS we get $O(d^n)$.

5.3 Computing pre-certificates

In this section we show how, given a candidate sample set Z , computing sampling SOS pre-certificates reduces to an SDP. We will also determine the condition we need on the sample set in order for such pre-certificate to be correct. The answer will be given by the concept of *poisedness* from polynomial interpolation. Finally, we will show how to reduce the size of the SDP in order to take advantage of the coordinate ring structure.

5.3.1 Sampling SDP

For a degree bound d , let $u(x) \in \mathbb{R}[x]^N$ denote the vector with all $N = \binom{n+d}{d}$ monomials of degree at most d . Recall from Proposition 2.2 that a polynomial $F \in \mathbb{R}[x]$ is d -SOS if and only if $F(x) = Q \bullet u(x)u(x)^T$ for some $Q \succeq 0$. Computing a polynomial F satisfying (5.3) reduces to the following SDP:

$$\boxed{\begin{array}{ll} \text{find} & Q \in \mathcal{S}^N, \quad Q \succeq 0 \\ \text{subject to} & p(z_s) = Q \bullet u(z_s)u(z_s)^T, \quad \text{for } s = 1, \dots, S \end{array}} \quad (5.8)$$

where \mathcal{S}^N denotes the space of $N \times N$ real symmetric matrices. Note that the matrix Q is real, whereas $p(z_s)$ and $u(z_s)$ are complex. Thus, each equality imposes a constraint on both

the real and the imaginary part, i.e.,

$$\Re(p(z_s)) = Q \bullet \Re(u(z_s)u(z_s)^T), \quad \Im(p(z_s)) = Q \bullet \Im(u(z_s)u(z_s)^T).$$

The above SDP has two important features: the polynomial p can be given as a *straight-line program* (i.e., it does not need to be expanded) and the constraint matrices have *low rank*. Indeed, the rank of the constraint matrices $\Re(u(z_s)u(z_s)^T)$ and $\Im(u(z_s)u(z_s)^T)$ is at most two. This special rank structure can be exploited in interior point methods, as discussed in [17, 92, 110]. In particular, the Hessian assembly takes only $O(N^3)$ operations for low rank matrices, as opposed to $O(N^4)$ for unstructured matrices.

Observe that the monomial vector $u(x)$ can be replaced by any other polynomial set with the same linear span. In particular, we will see in Section 5.3.3 that $u(x)$ can be chosen to be an orthogonal basis with respect to a natural inner product supported on the samples. Remarkably, this orthogonalization reduces complexity in the SDP by exploiting the algebraic *dependencies* of the coordinate ring $\mathbb{R}[\mathbf{V}]$. In addition, the conditioning of the problem might improve, as explained in [92] for the unconstrained case $\mathbf{V} = \mathbb{C}^n$.

Remark 5.4 (Kernel/Image form). The feasible set of (5.8) has the form $Q \succeq 0, Q \in \mathcal{Q}$, where

$$\mathcal{Q} = \{Q \in \mathcal{S}^N : Q \bullet A_i = b_i\}$$

is an affine subspace. We refer to the above representation of \mathcal{Q} as the *kernel form*. Alternatively, we can describe \mathcal{Q} explicitly by giving some generators, i.e.,

$$\mathcal{Q} = \{Q_0 + \sum_j \lambda_j Q_j : \lambda_j \in \mathbb{R}\}$$

where $Q_0 \bullet A_i = b_i$ and $Q_j \bullet A_i = 0$. We refer to this representation as the *image form*. Depending on the problem, either of them might be more convenient. In particular, if the number of constraints is close to the dimension of \mathcal{S}^N then the latter representation is more compact. This will be the case in the applications shown in Sections 5.7.2 and 5.7.3. For a given problem, we can decide which representation is better by estimating the number of variables used in both of them, as discussed in [105].

5.3.2 Poisedness implies correctness

We just showed how to compute a sampling SOS pre-certificate for a given sample set. However, this pre-certificate might be incorrect unless we are cautious with the sample set, as illustrated in the next example.

Example 5.1 (Incorrect pre-certificate). Let $\mathbf{V} \subseteq \mathbb{C}^2$ be the zero set of $h(x) := x_2^2 - 1$, that consists of two complex lines: $\mathbb{C} \times \{1\}$ and $\mathbb{C} \times \{-1\}$. Let $p(x) := x_1^2 - x_2 + 1$, which is nonnegative on $\mathbf{V} \cap \mathbb{R}^2$. Let $Z := \{(k, 1)\}_{k=1}^S \subseteq \mathbf{V}$ be a set of samples and let $F(x) := x_1^2$. Observe that (F, Z) is a sampling SOS pre-certificate, but it is not correct because $p(0, -1) \neq F(0, -1)$. This example illustrates that a sample set, regardless of its size, might lead to incorrect pre-certificates if it does not capture well the geometry of the variety (in this case Z misses one of the components of \mathbf{V}).

We now present a condition that guarantees the correctness of a pre-certificate. Let $\mathcal{R} = \mathbb{R}[\mathbf{V}]$ be the coordinate ring of the variety, which is the space where we will work on. In particular, we will see the entries of the polynomial vector $u(x)$, as well as $p(x)$, as elements of \mathcal{R} . We need the following definition.

Definition 5.2. Let $\mathbf{V} \subseteq \mathbb{C}^n$ be a self-conjugate variety and let $\mathcal{R} = \mathbb{R}[\mathbf{V}]$. Let $\mathcal{L} \subseteq \mathcal{R}$ be a linear subspace and let $Z \subseteq \mathbf{V}$ be a set of samples. We say that (\mathcal{L}, Z) is *poised*¹ if the only polynomial $q \in \mathcal{L}$ such that $q(z) = 0$ for all $z \in Z$ is the zero polynomial.

Remark 5.5. For any finite dimensional \mathcal{L} there is a finite set Z such that (\mathcal{L}, Z) is poised.

Let $\mathcal{L}_d \subseteq \mathcal{R}$ be the linear space spanned by the entries of $u(x)$, and let $\mathcal{L}_{2d} \subseteq \mathcal{R}$ be spanned by the entries of $u(x)u(x)^T$. Note that $F(x) = Q \bullet u(x)u(x)^T \in \mathcal{L}_{2d}$. The following proposition tells us that poisedness guarantees the correctness of a sampling SOS pre-certificate. Thus, a *good set of samples* is one such that (\mathcal{L}_{2d}, Z) is poised.

Proposition 5.1. Let $\mathbf{V} \subseteq \mathbb{C}^n$ be a self-conjugate variety, let $\mathcal{R} = \mathbb{R}[\mathbf{V}]$ and let $p \in \mathcal{R}$ be nonnegative on $\mathbf{V} \cap \mathbb{R}^n$. Let (F, Z) be a sampling SOS pre-certificate and let $\mathcal{L}_{2d} \subseteq \mathcal{R}$ be a linear subspace such that $p, F \in \mathcal{L}_{2d}$. If (\mathcal{L}_{2d}, Z) is poised then (F, Z) is correct.

¹In polynomial interpolation it is usually further required that $|Z| = D$, where D is the dimension of \mathcal{L} [115]. We do not impose such condition.

Proof. Let $g := p - F \in \mathcal{L}_{2d}$, and observe that $g(z) = 0$ for $z \in Z$. As (\mathcal{L}_{2d}, Z) is poised, this implies that $g = 0$ and thus $p = F \in \mathcal{R}$. \square

For the rest of this section we assume that the poisedness condition from above is satisfied. In Section 5.5 we will discuss how to choose the samples in order to satisfy this requirement.

5.3.3 Reducing complexity

The size of the PSD matrix Q from (5.8) is $\binom{n+d}{d}$. We can reduce the size of this matrix by taking advantage of the coordinate ring structure. The size of the new matrix will be given by the Hilbert function $H_{\mathbf{V}}(d)$; see Remark 5.3. To do so, we simply need to find a basis of the linear subspace $\mathcal{L}_d \subseteq \mathcal{R}$ spanned by the entries of $u(x)$. We now explain how to get an orthogonal basis $u^o(x)$ with respect to the inner product given in the next proposition.

Proposition 5.2. *Let $\mathbf{V} \subseteq \mathbb{C}^n$ be a self-conjugate variety and let $\mathcal{R} = \mathbb{R}[\mathbf{V}]$. Let $\mathcal{L}_d \subseteq \mathcal{R}$ be a linear subspace and let $Z \subseteq \mathbf{V}$ be a set of samples. Let $\langle \cdot, \cdot \rangle_Z : \mathcal{L}_d \times \mathcal{L}_d \rightarrow \mathbb{R}$ be*

$$\langle f, g \rangle_Z = \sum_{z \in Z} (f(z)g(\bar{z}) + f(\bar{z})g(z)).$$

If (\mathcal{L}_d, Z) is poised then $(\mathcal{L}_d, \langle \cdot, \cdot \rangle_Z)$ is a real inner product space.

Proof. It is clear that $\langle \cdot, \cdot \rangle_Z$ is bilinear and symmetric. Thus, we only need to check positive-ness. Observe that $\langle f, f \rangle_Z = \sum_{z \in Z} 2|f(z)|^2 \geq 0$, which is zero only if $f(z) = 0$ for all $z \in Z$. As $f \in \mathcal{L}_d$, the poisedness condition implies $f = 0$. \square

Remark 5.6. Note that if (\mathcal{L}_{2d}, Z) is poised then (\mathcal{L}_d, Z) is also poised.

To find an orthogonal basis, we will operate on the evaluation matrix U with columns $u(z)$ for $z \in Z$. Consider the real matrix $W := [\Re(U) | \Im(U)]$. Observe that $u(x)$ is an orthogonal basis with respect to $\langle \cdot, \cdot \rangle_Z$ if and only if the rows of W are orthogonal with respect to the standard real inner product. Thus, we just need to orthogonalize the rows of W . Using an SVD (or rank revealing QR), we can obtain a decomposition $W = TW^o$, where W^o has orthogonal rows and T is a real full rank transformation matrix. Let U^o be such that $W^o = [\Re(U^o) | \Im(U^o)]$. The matrix U^o encodes the new vector of orthogonal polynomials $u^o(x)$. We note that directly orthogonalizing the matrix U does not work, as the transformation matrix T would be complex.

Algorithm 6 Orthogonal basis on the coordinate ring

Input: Polynomial vector $u(x)$, samples Z of variety \mathbf{V}

Output: Orthogonal basis $u^o(x)$ and its evaluation matrix U^o

- 1: **procedure** ORTHBASIS($u(x), Z$)
 - 2: $U :=$ evaluation matrix with columns $u(z)$ for $z \in Z$
 - 3: $W := [\Re(U) \mid \Im(U)]$
 - 4: orthogonalize $W =: TW^o$, where $W^o(W^o)^T = \text{id}$
 - 5: let U^o be such that $W^o = [\Re(U^o) \mid \Im(U^o)]$
 - 6: let $u^o(x)$ be such that $u(x) = Tu^o(x)$
 - 7: **return** $u^o(x), U^o$
-

Example 5.2. Let \mathbf{V} be the complex variety of the set of rotation matrices $SO(2)$, i.e.,

$$\mathbf{V} = \{X \in \mathbb{C}^{2 \times 2} : X^T X = \text{id}_2, \text{Det}(X) = 1\}. \quad (5.9)$$

Let $p(X) = 4X_{21} - 2X_{11}X_{22} - 2X_{12}X_{21} + 3$, which is nonnegative on $\mathbf{V} \cap \mathbb{R}^{2 \times 2}$. We want to find a sampling SOS certificate. We can sample points on \mathbf{V} using the Cayley parametrization (5.4). Consider the following 3 complex samples:

$$z_1 = \begin{bmatrix} -0.6+0.8i & 1.2+0.4i \\ -1.2-0.4i & -0.6+0.8i \end{bmatrix}, \quad z_2 = \begin{bmatrix} -1.2+0.4i & 0.6+0.8i \\ -0.6-0.8i & -1.2+0.4i \end{bmatrix}, \quad z_3 = \begin{bmatrix} -0.75+0.25i & 0.75+0.25i \\ -0.75-0.25i & -0.75+0.25i \end{bmatrix}.$$

We fix the degree bound $d = 1$, and let $u(x) = (1, X_{11}, X_{12}, X_{21}, X_{22})$ be the monomials of degree at most d . The matrix of evaluations is:

$$U = \begin{bmatrix} 1 & 1 & 1 \\ -0.6+0.8i & -1.2+0.4i & -0.75+0.25i \\ -1.2-0.4i & -0.6-0.8i & -0.75-0.25i \\ 1.2+0.4i & 0.6+0.8i & 0.75+0.25i \\ -0.6+0.8i & -1.2+0.4i & -0.75+0.25i \end{bmatrix}$$

Using an SVD we obtain the orthogonalized matrix U^o and the corresponding polynomial basis $u^o(x)$. Note that $u^o(x)$ has only 3 elements, as opposed to $u(x)$.

$$U^o = \begin{bmatrix} -0.5955+0.1005i & -0.5955-0.1005i & -0.5201i \\ 0.3058+0.6116i & -0.3058+0.6116i & 0.2548i \\ -0.0708+0.6411i & -0.0708-0.6411i & 0.4100 \end{bmatrix} \quad \begin{aligned} u^o(X) = (X_{21} + X_{22} - .8054, & X_{21} - X_{22}, \\ & X_{21} + X_{22} + 2.4831) \end{aligned}$$

The sampling SDP is

$$\begin{aligned} & \text{find} && Q \in \mathcal{S}^3, \quad Q \succeq 0 \\ & \text{subject to} && p(z_s) = Q \bullet u_s u_s^T, \quad \text{for } s = 1, 2, 3 \end{aligned}$$

where u_s denotes the s -th column of U° . Solving the SDP we obtain the sampling pre-certificate (F, Z) , where $F(X) = (2X_{21} + 1)^2$.

5.4 Verifying sampling pre-certificates

We now address the problem of testing the correctness of a pre-certificate (F, Z) . This problem is equivalent to determining whether the polynomial $f := p - F$ is identically zero on the variety \mathbf{V} , and it is known as the *identity testing* problem (see e.g., [117] and the references therein). Note that the problem is nontrivial even if $\mathbf{V} = \mathbb{C}^n$, since f can be given as a straight-line program (such as a determinant). Nonetheless, there is a nice randomized algorithm to solve it, provided that we can efficiently sample the variety. Recall that generic samples can be obtained as explained in Section 5.2.2. We now proceed to review the notion of genericity, as well as showing the solution to the identity testing problem.

5.4.1 Genericity

The notion of *genericity* is fundamental in algebraic geometry. Let $\mathbf{V} \subseteq \mathbb{C}^n$ be an irreducible variety of positive dimension. We say that a property holds *generically* on \mathbf{V} if there is a nonzero polynomial $q \in \mathbb{C}[\mathbf{V}]$ such that the property holds for any $z \in \mathbf{V}$ such that $q(z) \neq 0$. Informally, this means that the property holds outside of the small bad region given by $q(z) = 0$. In Section 5.5.2 we will use a variation of this notion of genericity that is better suited for dealing with real polynomials.

Example 5.3. Let $\mathbf{V} = \mathbb{C}^{m \times m}$ be the space of $m \times m$ matrices. The property of being “nonsingular” is satisfied generically on \mathbf{V} , since a matrix $A \in \mathbf{V}$ is singular only if $\text{Det}(A) = 0$.

We often say that a sample point $z \in \mathbf{V}$ is *generic* if some property of interest (such as the conclusion of a theorem) is satisfied generically on \mathbf{V} . For instance, we may say “a generic

$m \times m$ matrix is nonsingular”. A generic point can be understood as a random point on the variety.

Proposition 5.3. *Let \mathbf{V} be an irreducible variety, let $f \in \mathbb{C}[\mathbf{V}]$ be a nonzero polynomial and let $z \in \mathbf{V}$ be a generic sample. Then $f(z)$ is nonzero.*

Proof. The conclusion holds except in the bad region defined by $f(z) = 0$. \square

5.4.2 Identity testing

Genericity allows us to derive *randomized* algorithms that succeed with *probability one* with respect to any distribution on \mathbf{V} with full support. In particular, Proposition 5.3 gives rise to Algorithm 7. This method efficiently solves the identity testing problem for an irreducible variety (provided that it can be sampled). Surprisingly, no efficient deterministic algorithm to this problem is known, and it is likely that finding such algorithm is very hard [76].

Algorithm 7 Identity testing over \mathbb{C}

Input: Polynomial $f \in \mathbb{C}[x]$, irreducible variety \mathbf{V}

Output: “True”, if f is identically zero on \mathbf{V} . “False”, otherwise.

- 1: **procedure** ISZERO(f, \mathbf{V})
 - 2: $z :=$ generic sample from \mathbf{V}
 - 3: **return** True **if** $f(z) = 0$ **else** False
-

Remark 5.7 (Reducible varieties). If the variety \mathbf{V} is reducible, we can still solve the identity testing problem provided that we can sample each of its irreducible components. We simply need to apply Algorithm 7 to each component.

Remark 5.8 (Probability one). Randomized algorithms derived from genericity statements provably work with probability one in exact arithmetic. However, in floating point arithmetic there is a nonzero probability of error, thus leading to Monte Carlo algorithms; for further discussion see [124, §4].

5.5 Selecting the samples

The missing step to complete our sampling SOS methodology is to describe how to obtain a good set of samples Z . Recall from Section 5.3.2 that a good sample set must be such that

(\mathcal{L}_{2d}, Z) is poised. Thus the question we address here is the following: given a linear space $\mathcal{L}_{2d} \subseteq \mathcal{R} = \mathbb{R}[\mathbf{V}]$, how can we get a sample set Z such that (\mathcal{L}_{2d}, Z) is poised. We will see in this section that this condition can be satisfied with a generic set of samples.

5.5.1 Poisedness again

Before proceeding to the selection of the samples, we first present an alternative characterization of poisedness. Let $\mathcal{L} \subseteq \mathcal{R}$ be a finite dimensional subspace. Let $v(x) \in \mathcal{R}^N$ be a polynomial vector whose entries span \mathcal{L} . Let U be the matrix with columns $v(z)$ for $z \in Z$ and let $\hat{U} := [U | \bar{U}]$. We will refer to the (complex) rank of matrix \hat{U} as the *empirical dimension* of \mathcal{L} with respect to Z . It is easy to see that it does not depend on the choice of generators.

Lemma 5.4. *(\mathcal{L}, Z) is poised if and only if the dimension of \mathcal{L} is equal to its empirical dimension.*

Proof. Let $D := \dim(\mathcal{L})$ and assume that $v(x) \in \mathcal{R}^D$ is a basis. Assume first that the $\text{rk}(\hat{U}) = D$. Note that any $q \in \mathcal{L}$ can be written uniquely in the form $q(x) = \mu^T u(x)$ for some $\mu \in \mathbb{R}^D$. The condition that $q(z) = 0$ for $z \in Z \cup \bar{Z}$ implies that $\mu^T \hat{U} = 0$. As \hat{U} has full row rank then $\mu = 0$, and thus (\mathcal{L}, Z) is poised. Assume now that \hat{U} does not have full row rank. Then there is some nonzero $\lambda \in \mathbb{C}^D$ such that $\lambda^T \hat{U} = 0$. Observe that this implies that $\Re(\lambda)^T U = \Im(\lambda)^T U = 0$, where \Re, \Im denote the real/imaginary part. Thus, there is a nonzero $\mu \in \mathbb{R}^D$ such that $\mu^T U = 0$. Considering the polynomial $q(x) := \mu^T u(x)$, we conclude that (\mathcal{L}, Z) is not poised. \square

Remark 5.9. Since matrix \hat{U} has $2|Z|$ columns, it follows from the lemma that if (\mathcal{L}, Z) is poised then $2|Z| \geq \dim \mathcal{L}$.

Example 5.4. Let $\mathbf{V} = \mathbb{C}$ and $\mathcal{R} = \mathbb{R}[x]$ be the space of univariate polynomials. Let \mathcal{L} be the set of polynomials of degree less than N and let $v(x) = (1, x, \dots, x^{N-1})$. Let $Z \in \mathbb{C}^{N/2}$ be a tuple of complex samples, and let \hat{Z} be the concatenation of Z and \bar{Z} . The evaluation matrix \hat{U} in this case is the Vandermonde matrix of \hat{Z} , which is singular only if there are repeated elements in \hat{Z} . Therefore, (\mathcal{L}, Z) is poised if and only if the elements of \hat{Z} are all distinct.

5.5.2 How many samples

We return to the problem of finding a sample set Z such that (\mathcal{L}_{2d}, Z) is poised. As we decided that the samples will be random, the only missing point is to determine how many samples to take. Remark 5.9 tells us that we need at least $\lceil D/2 \rceil$ samples, where $D := \dim(\mathcal{L}_{2d})$. We wonder if this condition is *generically sufficient* to guarantee poisedness.

Question. Let \mathbf{V} be a self-conjugate variety. Let $\mathcal{L}_{2d} \subseteq \mathbb{R}[\mathbf{V}]$ be a D -dimensional linear subspace and let $Z \subseteq \mathbf{V}$ be a generic set of samples with $|Z| \geq D/2$. Is (\mathcal{L}_{2d}, Z) poised?

In order to make sense of the above question, we have to be more precise about the meaning of a generic set of samples. In Section 5.4.1 we saw the definition of a generic sample of an irreducible variety. We have to extend this definition to multiple samples, taken possibly from a reducible variety. Below we formalize the exact notion of genericity we use. It is slightly different from the one in Section 5.4.1 as it includes the complex conjugates of the samples. The reason for including the conjugates is that it reflects the fact that we are working with real polynomials.

Definition 5.3. Let $\mathbf{W} \subseteq \mathbb{C}^n$ be an irreducible variety and let $Z = (z_1, \dots, z_S)$ be a tuple of S samples in \mathbf{W} . We say that Z satisfies a property *c-generically* (conjugate generically) if there is a polynomial $q \in \mathbb{C}[z_1, \dots, z_S, \overline{z_1}, \dots, \overline{z_S}]$ such that:

- $q(z_1, \dots, z_S, \overline{z_1}, \dots, \overline{z_S})$ is not identically zero when $z_1, \dots, z_S \in \mathbf{W}$.
- the property holds whenever $q(z_1, \dots, z_S, \overline{z_1}, \dots, \overline{z_S}) \neq 0$.

Let $\mathbf{W}_1, \dots, \mathbf{W}_r$ be irreducible varieties and let $Z_1 \subseteq \mathbf{W}_1, \dots, Z_r \subseteq \mathbf{W}_r$ be tuples of samples. We say that (Z_1, \dots, Z_r) satisfies a property *c-generically* if there are polynomials $q_1 \in \mathbb{C}[Z_1, \overline{Z_1}], \dots, q_r \in \mathbb{C}[Z_r, \overline{Z_r}]$ such that:

- $q_i(Z_i, \overline{Z_i})$ is not identically zero on \mathbf{W}_i , for $1 \leq i \leq r$.
- the property holds whenever $q_1(Z_1, \overline{Z_1}) \neq 0, \dots, q_r(Z_r, \overline{Z_r}) \neq 0$.

We say that Z (resp. Z_1, \dots, Z_r) is a *c-generic* set of samples if it satisfies some property of interest *c-generically*.

In the next section we will show that for an irreducible variety (or a conjugate pair of irreducible varieties) the answer to the question from above is positive. However, for reducible varieties, we need to make sure that we have enough samples in each irreducible component, as will be discussed in Section 5.5.5. Example 5.1 illustrates what might go wrong if we do not have enough samples in some component.

5.5.3 The irreducible case

Assume now that $\mathbf{V} = \mathbf{W} \cup \overline{\mathbf{W}}$, where $\mathbf{W} \subseteq \mathbb{C}^n$ is an irreducible variety. This means that either \mathbf{V} is a self-conjugate irreducible variety, or it is a conjugate pair of irreducible varieties. In the latter case, note that we can assume without loss of generality that $Z \subseteq \mathbf{W}$, by possibly exchanging some samples with their complex conjugates. We show now that if the samples Z are c-generic and are at least as many as the dimensionality of the problem, then the poisedness property is satisfied.

Theorem 5.5. *Let $\mathbf{W} \subseteq \mathbb{C}^n$ be an irreducible variety, let $\mathbf{V} = \mathbf{W} \cup \overline{\mathbf{W}}$ and let $\mathcal{R} = \mathbb{R}[\mathbf{V}]$. Let $\mathcal{L}_{2d} \subseteq \mathcal{R}$ be a linear subspace and let $Z \subseteq \mathbf{W}$ be a c-generic set of samples. If $|Z| \geq D/2$, where $D := \dim(\mathcal{L}_{2d})$, then (\mathcal{L}_{2d}, Z) is poised².*

Proof. Let $v(x) \in \mathcal{R}^D$ be a basis of \mathcal{L}_{2d} . Let $Z_j := \{z_1, \dots, z_j\}$, let $V_j \in \mathbb{C}^{D \times j}$ be the matrix with columns $\{v(z)\}_{z \in Z_j}$ and let $\hat{V}_j := [\Re(V_j) | \Im(V_j)] \in \mathbb{R}^{D \times 2j}$. Also denote $W_j := [\hat{V}_j | \Im(v(z_{j+1}))] \in \mathbb{R}^{D \times 2j+1}$. Because of Lemma 5.4, we just need to show that the matrix \hat{V}_S has rank D . To this end, we will show the following statements:

- if \hat{V}_{j-1} is full rank then W_{j-1} is full rank c-generically.
- if W_{j-1} is full rank then \hat{V}_j is full rank c-generically.

Clearly these statements imply that \hat{V}_S is full rank. Given the similarity between the two of them, we only prove the latter.

Let $j \leq D/2$ and assume that W_{j-1} is full rank. We will show that there is a polynomial $Q \in \mathbb{C}[Z_j, \overline{Z_j}]$ which is not identically zero on \mathbf{W} , and such that \hat{V}_j is full rank whenever $Q(Z_j, \overline{Z_j}) \neq 0$.

²This theorem is a special instance of the dimensionality problem in polynomial interpolation, and more elaborate versions can be found in the literature [43].

Assume that \hat{V}_j is not full rank. Then there must exist a vector $\lambda \in \mathbb{R}^{2j-1}$ such that

$$v(z_j) + v(\overline{z_j}) = 2\Re(v(z_j)) = W_{j-1}\lambda.$$

As W_{j-1} has less than D columns, there is some nonzero vector $\mu \in \mathbb{R}^D$ in its left kernel. Note that $\mu = \mu(Z_j, \overline{Z_j})$ can be parametrized as a rational function of $Z_j, \overline{Z_j}$, given that W_{j-1} is full rank. Let $q_\mu(x) := \mu^T v(x) \in \mathcal{R}$, which is nonzero due to the linear independence of $v(x)$. Observe that

$$q_\mu(z_j) + q_\mu(\overline{z_j}) = \mu^T W_{j-1} \lambda = 0.$$

As the coefficients of q_μ are rational functions on $Z_j, \overline{Z_j}$, we conclude that the samples satisfy a nonzero algebraic equation $Q \in \mathbb{C}[Z_j, \overline{Z_j}]$. \square

Remark 5.10. If the samples are real, it can be shown in a similar way that we need $|Z| \geq D$.

5.5.4 Verifying the number of samples

We just showed that, under genericity assumptions, the poisedness property is satisfied whenever we have as many samples as the dimension of the space. Concretely, we need to have $\lceil D/2 \rceil$ complex samples, where $D = \dim(\mathcal{L}_{2d})$. However, as the dimension D is not known a priori, it is uncertain how many samples to take. Therefore, we need some way to estimate such dimension, and the natural quantity to consider is the empirical dimension D_e . The following corollary gives us a simple test that guarantees that $D = D_e$.

Proposition 5.6. *Let $\mathbf{W} \subseteq \mathbb{C}^n$ be an irreducible variety, let $\mathbf{V} = \mathbf{W} \cup \overline{\mathbf{W}}$ and let $\mathcal{R} = \mathbb{R}[\mathbf{V}]$. Let $\mathcal{L}_{2d} \subseteq \mathcal{R}$ be a linear subspace and let $Z \subseteq \mathbf{W}$ be a c -generic set of samples. Let D be the dimension of \mathcal{L}_{2d} and let D_e be its empirical dimension with respect to Z . If $D_e < 2|Z|$ then (\mathcal{L}_{2d}, Z) is poised (i.e., $D = D_e$).*

Proof. If $2|Z| < D$ it follows from the proof of Theorem 5.5 that $D_e = 2|Z|$. Therefore, we must have that $2|Z| \geq D$, and thus (\mathcal{L}_{2d}, Z) is poised because of Theorem 5.5. \square

The above corollary suggests a simple strategy that is summarized in Algorithm 8. We form the vector $u_2(x) = \text{vec}(u(x)u(x)^T)$, whose entries span \mathcal{L}_{2d} . Then we build the matrix

Algorithm 8 Test samples

Input: Polynomial vector $u(x)$, samples Z of a variety \mathbf{V}

Output: “True”, if generically we must have that (\mathcal{L}_{2d}, Z) is poised, where $\mathcal{L}_{2d} \subseteq \mathbb{R}[\mathbf{V}]$ is spanned by $u(x)u(x)^T$. “False”, if we cannot guarantee it.

- 1: **procedure** GOODSAMPLES($u(x), Z$)
 - 2: $\hat{U}_2 :=$ matrix with columns $\text{vec}(u(z)u(z)^T)$, for $z \in Z \cup \bar{Z}$
 - 3: **return** False **if** \hat{U}_2 has full column rank **else** True
-

of evaluations \hat{U}_2 with columns $u_2(z)$ for $z \in Z \cup \bar{Z}$. The rank of this matrix is the empirical dimension D_e . If \hat{U}_2 does not have full column rank the above corollary holds.

Remark 5.11. Consider the Hermitian matrix $\hat{U}_2^* \hat{U}_2$, where $*$ denotes the conjugate transpose. This matrix is often much smaller than \hat{U}_2 , and it can be constructed efficiently as

$$\hat{U}_2^* \hat{U}_2 = [\langle u(z_i), u(z_j) \rangle^2]_{z_i, z_j \in Z \cup \bar{Z}} = (\hat{U}^* \hat{U}) \circ (\hat{U}^* \hat{U})$$

where \circ denotes the Hadamard product. Therefore, it is practical to use matrix $\hat{U}_2^* \hat{U}_2$ instead of \hat{U}_2 , given that they have the same rank.

Example 5.5. Consider the case of Example 5.2. We used $S = 3$ samples to compute the pre-certificate. To verify that the number of samples was sufficient, we construct the matrix

$$\hat{U}_2^* \hat{U}_2 = \begin{bmatrix} 1.5581 & -0.2937+0.2562i & 0.1730-0.1158i & 0.0902+0.1118i & 0.0981 & -0.0676-0.0720i \\ -0.2937-0.2562i & 1.5581 & 0.1730+0.1158i & 0.0981 & 0.0902-0.1118i & -0.0676+0.0720i \\ 0.1730+0.1158i & 0.1730-0.1158i & 0.2535 & -0.0676-0.0720i & -0.0676+0.0720i & 0.1396 \\ 0.0902-0.1118i & 0.0981 & -0.0676+0.0720i & 1.5581 & -0.2937-0.2562i & 0.1730+0.1158i \\ 0.0981 & 0.0902+0.1118i & -0.0676-0.0720i & -0.2937+0.2562i & 1.5581 & 0.1730-0.1158i \\ -0.0676+0.0720i & -0.0676-0.0720i & 0.1396 & 0.1730-0.1158i & 0.1730+0.1158i & 0.2535 \end{bmatrix}$$

The rank of this matrix is 5, and thus the condition from Proposition 5.6 is satisfied. Therefore, the number of samples is sufficient.

5.5.5 Reducible varieties

The analysis made so far makes an irreducibility assumption on the variety \mathbf{V} . This assumption is satisfied for many varieties, in particular for any variety parametrized by \mathbb{C}^n . Even if \mathbf{V} is not irreducible, we can always work with each of its irreducible components independently. Indeed, note that $p \geq 0$ on some variety if and only if $p \geq 0$ on each irreducible component.

Nonetheless, there are circumstances in which we may not want to impose an irreducibility assumption. For example, if the variety has bad numerical properties and thus its irreducible

components cannot be accurately estimated. In such situations, we can repeat the same analysis from before if we have some method that samples points from each irreducible component. For instance, if we intersect the variety \mathbf{V} with a generic hyperplane of complementary dimension, the intersection is a finite set that contains points in each irreducible component. Note that we do not know which component do the samples belong to, but we are certain that there is at least one sample in each component.

The following corollary shows that if we have a sample set with enough points on each irreducible component, then (\mathcal{L}_{2d}, Z) is poised.

Proposition 5.7. *Let $\mathbf{W} \subseteq \mathbb{C}^n$ be a variety, let $\mathbf{V} = \mathbf{W} \cup \overline{\mathbf{W}}$ and let $\mathcal{R} = \mathbb{R}[\mathbf{V}]$. Let $\mathcal{L}_{2d} \subseteq \mathcal{R}$ be a linear subspace. Let $\mathbf{W} = \mathbf{W}_1 \cup \dots \cup \mathbf{W}_r$ be the irreducible decomposition, and let $Z_1 \subseteq \mathbf{W}_1, \dots, Z_r \subseteq \mathbf{W}_r$ be c -generic sets of samples. If $|Z_i| \geq D/2$ for all i , where $D := \dim(\mathcal{L}_{2d})$, then (\mathcal{L}_{2d}, Z) is poised.*

Proof. Let $f \in \mathcal{L}_{2d}$ be such that $f(z) = 0$ for all $z \in Z$. We want to show that f is the zero polynomial in $\mathbb{C}[\mathbf{V}]$. Let $\mathbf{V}_i := \mathbf{W}_i \cup \overline{\mathbf{W}_i}$ and let $\psi_i : \mathbb{C}[\mathbf{V}] \rightarrow \mathbb{C}[\mathbf{V}_i]$ be the restriction operator. It is clear that the dimension of $\psi_i(\mathcal{L}_{2d})$ is at most D . Thus, Theorem 5.5 says that $(\psi_i(\mathcal{L}_{2d}), Z_i)$ is poised whenever $q_i(Z_i, \overline{Z_i}) \neq 0$, for some polynomial q_i which is nonzero on \mathbf{W}_i . Note that $\psi_i(f)$ evaluates to zero on Z_i , and thus $\psi_i(f)$ must be the zero element in $\mathbb{C}[\mathbf{V}_i]$ whenever $q_i(Z_i, \overline{Z_i}) \neq 0$. Finally, observe that $(\psi_1 \times \dots \times \psi_k) : \mathbb{C}[\mathbf{V}] \rightarrow \mathbb{C}[\mathbf{V}_1] \times \dots \times \mathbb{C}[\mathbf{V}_k]$ is injective. We conclude that whenever $q_i(Z_i, \overline{Z_i}) \neq 0$ then $(\psi_1 \times \dots \times \psi_k)(f)$ is zero and thus f must be zero. \square

5.6 Computing sampling certificates

We already developed all the tools needed to find a sampling SOS certificate, and we now put them together. Algorithm 9 summarizes our method for the case of an irreducible variety \mathbf{V} . Naturally, the most computationally expensive part is solving the SDP. Recall from Theorem 5.5 that the number of samples required is

$$S_{\min} := \lceil H_{\hat{\mathbf{V}}}(2d)/2 \rceil, \quad H_{\hat{\mathbf{V}}}(2d) := \dim(\mathcal{L}_{2d}) \leq \min \left\{ \binom{n+2d}{2d}, \deg \hat{V} \binom{\dim \hat{V} + 2d}{2d} \right\}, \quad (5.10)$$

where $\hat{\mathbf{V}} = \mathbf{V} \cup \overline{\mathbf{V}}$, and where we used the bound from (5.7). Since $H_{\hat{\mathbf{V}}}(2d)$ is unknown in general, we use a simple search strategy in Algorithm 9. The algorithm terminates when the number of samples is at least S_{\min} .

In the case of a reducible variety, we might use Algorithm 9 for each of its irreducible components separately. If we cannot reliably identify such components we need to take into account the considerations from Section 5.5.5.

Remark 5.12 (Zero-dimensional case). Note that a zero-dimensional variety is reducible, each component consisting of a single point. Thus, in such case our algorithm reduces to a brute-force enumeration over all solutions, and better strategies may exist. The main problem to address is that of producing small poised sets. We leave this as an open problem.

Algorithm 9 Sampling SOS

Input: Polynomial $p \in \mathbb{R}[x]$ (given by an evaluation oracle), irreducible variety $\mathbf{V} \subseteq \mathbb{C}^n$ (given by a sampling oracle), degree bound $d \in \mathbb{N}$

Output: d -SOS(\mathbf{V}) certificate F , if it exists. “Null”, if no certificate exists.

```

1: procedure SAMPLINGSOS( $p, \mathbf{V}, d$ )
2:    $u(x) :=$  vector with all monomials up to degree  $d$ 
3:    $S :=$  initial guess on the number of samples (an upper bound is given in (5.10))
4:    $Z :=$  generic set of  $S$  samples from  $\mathbf{V}$ 
5:    $u(x) := \text{ORTHBASIS}(u(x), Z)$  ▷ find basis of  $\mathbb{R}[\mathbf{V}]$ 
6:   if not GOODSAMPLES( $u(x), Z$ ) then ▷ check if there are enough samples
7:     increase  $S$  and go to 4
8:    $Q :=$  solution of SDP (5.8) (if none return Null) ▷ solve SDP
9:    $F(x) := Q \bullet u(x)u(x)^T$ 
10:  if not ISZERO( $p - F, \mathbf{V}$ ) then ▷ verify correctness
11:    the sample set was not generic enough; go to 4
12:  return  $F$ 
```

5.7 Examples

We now show several examples and numerical evaluations to illustrate our methodology. We implemented our algorithms in Matlab, using SDPT3 [133] to solve the semidefinite programs. We also use Macaulay2 [67] for Gröbner bases, and Bertini [13] for numerical algebraic geometry computations. The experiments are performed on an i7 PC with 8 cores of 3.40 GHz, 15.6 GB RAM, running Ubuntu 14.04. We will compare our techniques with the following two methods: equations SOS (5.5) and the (Gröbner bases based) quotient ring SOS (5.6).

5.7.1 Nilpotent matrices

Let $\mathbf{V} := \{X \in \mathbb{C}^{n \times n} : X^n = 0\}$ be the variety of nilpotent matrices. Let $p(X) := \text{Det}(X + \text{id}_n)$, which is nonnegative on \mathbf{V} (moreover, it is identically one). We compare different SOS methodologies to certify this.

First, consider the sampling approach. Let the degree bound $d = 1$, and let us take $S = \binom{n+2}{2}$ samples, which are always sufficient. Note that it is very easy to sample nilpotent matrices. For instance, we can generate a random triangular matrix with zero diagonal, and then apply a similarity transformation. For each sample $X_s \in \mathbf{V}$, we can efficiently evaluate $p(X_s)$ with Gaussian elimination. As $p(X_s) = 1$ for all samples X_s , we will obtain the trivial SOS decomposition $p(X) \equiv_{\mathbf{V}} (1)^2$.

Consider now the Gröbner bases approach. Let $h \subseteq \mathbb{R}[X]$ be the n^2 equations given by $X^n = 0$. We want to compute a Gröbner basis of h . Note, however, that the total number of terms in h is on the order of n^{n+1} , and the polynomials are all of degree n . Therefore, this Gröbner basis computation is extremely complicated.

If we are smarter, we can take a different set of defining equations of \mathbf{V} . Consider the polynomial $Q_X(t) := \text{Det}(t \text{id}_n - X) - t^n$, and let $h' \subseteq \mathbb{R}[X]$ be the equations given by the coefficients of $Q_X(t)$. It turns out that h' generates the radical ideal of $\langle h \rangle$, and moreover, it is a Gröbner basis [74, §7]. However, h' has more than $n!$ terms. Once we have the Gröbner basis h' , we need to compute the normal form of p . To obtain this normal form we need to consider p as a dense polynomial. As both p and h' have on the order of $n!$ terms, performing this reduction is computationally intractable. If we are able to reduce it, we will conclude that $p(X) \equiv_{\mathbf{V}} 1$, as before.

Finally, note that equations SOS suffers from the same problems of the Gröbner bases approach. For this method there is an additional problem, which is that the monomial basis $u(X)$ will be very large in order to account for all the monomials in $p(X)$ and $h(X)$. This problem was avoided in the previous methods because of the quotient ring reductions.

This example illustrates two of the advantages of the sampling formulation: it avoids the algebraic problem of deciding which equations to use (e.g., h vs. h'), and it allows the use of straight-line programs (e.g., Gaussian elimination) for more efficient evaluations.

5.7.2 Weighted orthogonal Procrustes

We consider a family of optimization problems over varieties of orthogonal matrices. The Stiefel manifold $St(k, \mathbb{R}^n)$ is the set of orthonormal k -frames in \mathbb{R}^n . We identify it with the set of matrices $X \in \mathbb{R}^{n \times k}$ such that $X^T X = \text{id}_k$. Note that we can easily sample points from this variety, for instance, by using the Cayley parametrization. Alternatively, we can orthogonalize a random real matrix.

The weighted orthogonal Procrustes problem, also known as Penrose regression problem, asks for a matrix $X \in St(k, \mathbb{R}^n)$ that minimizes $\|AXC - B\|_F$, for some matrices $A \in \mathbb{R}^{m_1 \times n}$, $B \in \mathbb{R}^{m_1 \times m_2}$, $C \in \mathbb{R}^{k \times m_2}$. There is no closed form solution for this problem, and several local optima may exist [35, 138].

Let $u(x)$ consist of all monomials up to some degree bound d . The sampling SDP is:

$$\begin{aligned} & \max_{\gamma \in \mathbb{R}, Q \succeq 0} \quad \gamma \\ & \text{subject to} \quad \|AX_s C - B\|_F^2 - \gamma = Q \bullet u(X_s)u(X_s)^T, \quad \text{for } s = 1, \dots, S \\ & \quad \quad \quad X_s \in St(k, \mathbb{R}^n) \end{aligned}$$

Example 5.6 ([35], Ex 2). Let $(n, k, m_1, m_2) = (4, 3, 5, 3)$ and consider the matrices

$$A^T = \begin{bmatrix} 0.2190 & 0.0470 & 0.6789 & 0.6793 & 0.9347 \\ 0.3835 & 0.5194 & 0.8310 & 0.0346 & 0.0535 \\ 0.5297 & 0.6711 & 0.0077 & 0.3834 & 0.0668 \\ 0.4175 & 0.6868 & 0.5890 & 0.9304 & 0.8462 \end{bmatrix}, \quad B^T = \begin{bmatrix} 0.6526 & 0.2110 & 0.2229 & -0.4104 & -0.9381 \\ 0.6942 & 0.2204 & 0.2015 & 0.2994 & 1.0943 \\ 0.8299 & 1.1734 & -0.1727 & 0.0474 & -0.2351 \end{bmatrix}, \quad C = \text{id}_3.$$

We consider the degree 1 SOS relaxation. Following Algorithm 9, we find out that $S = 43$ complex (or 85 real) samples are sufficient. More generally, the required number of samples is a half of $H_{\mathbf{V}}(2) = \binom{n+k+2}{2} - \binom{k+1}{2}$. By solving the above SDP we obtain a lower bound of 1.118147 on the minimum norm $\|AXC - B\|_F$. Furthermore, the dual SDP matrix has rank one, and thus we can recover a solution achieving such lower bound:

$$(X^*)^T = \begin{bmatrix} -0.0895 & 0.7472 & 0.2732 & -0.5992 \\ 0.7726 & -0.1843 & 0.6035 & -0.0702 \\ -0.5277 & 0.0163 & 0.7309 & 0.4324 \end{bmatrix}.$$

Table 5.1 compares different SDP formulations of the degree 1 SOS relaxation of the weighted orthogonal Procrustes problem. We consider the case where $m_1 = n$ and $m_2 = k$. The table shows the number of variables/constraints and the computation time for the

equations SDP and the sampling SDP. The computation is performed on random instances, in which matrices A , B , C are generated from the standard normal distribution. For the sampling SDP we use the image form of the SDP (see Section 5.3.1), given that it has low codimension. We remark that for the sampling SDP we include the preprocessing time, i.e., Algorithms 6 and 8.

We point out that the better performance of sampling SDP is due to the fact that it makes use of the quotient ring structure. Although a similar sized SDP could be derived using Gröbner bases, Table 5.1 shows that Gröbner bases computation is very expensive, much more than solving the (larger) equations SDP. In particular, Macaulay2 ran out of memory for $n = 7$, $k = 5$.

Table 5.1: Degree 1 SOS relaxations for the weighted orthogonal Procrustes problem

n	k	Equations SDP			Sampling SDP			Gröbner bases time(s)
		variables	constraints	time(s)	variables	constraints	time(s)	
4	2	178	73	0.52	46	42	0.10	0.00
5	3	682	233	0.65	137	130	0.11	0.03
6	4	1970	576	1.18	326	315	0.15	9.94
7	5	4727	1207	3.56	667	651	0.31	out of mem.
8	6	9954	2255	13.88	1226	1204	0.70	out of mem.
9	7	19028	3873	42.14	2081	2052	2.11	out of mem.
10	8	33762	6238	124.43	3322	3285	5.07	out of mem.

5.7.3 Trace ratio problem

We now consider a problem on the Grassmannian manifold $Gr(k, \mathbb{R}^n)$, which is the set of all k -dimensional subspaces of \mathbb{R}^n . Note that we can easily sample points on $Gr(k, \mathbb{R}^n)$ by considering the subspace spanned by k random vectors. By identifying a subspace with the orthogonal projection onto it, we can view $Gr(k, \mathbb{R}^n)$ as the set of matrices $X \in \mathcal{S}^n$ satisfying $X^2 = X$ and $\text{tr}(X) = k$; so this is indeed a variety. The trace ratio problem looks for the maximizer of $\frac{\text{tr}(AX)}{\text{tr}(BX)}$ on $Gr(k, \mathbb{R}^n)$, for some given matrices $A, B \in \mathcal{S}^n$, $B \succ 0$. This problem arises in machine learning, and it can be efficiently solved by iterative methods, given that it has a unique local maximum [145]. We consider the following variation:

$$\max_{X \in Gr(k, \mathbb{R}^n)} \frac{\text{tr}(AX)}{\text{tr}(BX)} + \text{tr}(CX)$$

for some $A, B, C \in \mathcal{S}^n$, $B \succ 0$. This problem may have several local maxima and thus local methods may not converge to the global optimum [151, 152].

To obtain an SOS relaxation, note that the problem is equivalent to minimizing γ such that $\text{tr}(BX)(\gamma - \text{tr}(CX)) - \text{tr}(AX)$ is nonnegative on $Gr(k, \mathbb{R}^n)$. Thus, the SDP to consider is:

$$\begin{aligned} \min_{\gamma \in \mathbb{R}, Q \succeq 0} \quad & \gamma \\ \text{subject to} \quad & \text{tr}(BX_s)(\gamma - \text{tr}(CX_s)) - \text{tr}(AX_s) = Q \bullet u(X_s)u(X_s)^T, \quad \text{for } s = 1, \dots, S \\ & X_s \in Gr(k, \mathbb{R}^n) \end{aligned}$$

Example 5.7 ([152], Ex 3.1). Let $n = 3, k = 2$ and consider the matrices A, B, C from below. For the degree bound $d = 1$, Algorithm 9 gives that $S = 8$ complex (or 15 real) samples are sufficient. In general, the number of samples is a half of $H_{\mathbf{V}}(2) = \binom{\frac{1}{2}(n^2+n)}{2}$. Solving the above SDP we get an upper bound of 28.692472. As the dual matrix has rank one, we can recover the optimal solution X^* .

$$A = \begin{bmatrix} 11 & 5 & 8 \\ 5 & 10 & 9 \\ 8 & 9 & 5 \end{bmatrix}, \quad B = \begin{bmatrix} 7 & 7 & 7 \\ 7 & 10 & 8 \\ 7 & 8 & 8 \end{bmatrix}, \quad C = \begin{bmatrix} 15 & 10 & 9 \\ 10 & 7 & 6 \\ 9 & 6 & 6 \end{bmatrix}, \quad X^* = \begin{bmatrix} 0.61574 & 0.15424 & 0.46132 \\ 0.15424 & 0.93809 & -0.18517 \\ 0.46132 & -0.18517 & 0.44617 \end{bmatrix}$$

As before, we compare the equations SDP and the sampling SDP of the degree 1 SOS relaxation. Table 5.2 shows the number of variables/constraints and the computation time on random instances for both methods. It also shows the computation time of Gröbner bases.

Table 5.2: SOS relaxations for the trace ratio problem

Degree 1 SOS relaxations for the trace ratio problem									
n	k	Equations SDP			Sampling SDP			Gröbner bases	
		variables	constraints	time(s)	variables	constraints	time(s)	time(s)	
4	2	342	188	0.47	56	45	0.10	0.00	
5	3	897	393	0.71	121	105	0.11	0.02	
6	4	2062	738	1.34	232	210	0.15	0.20	
7	5	4265	1277	3.62	407	378	0.19	6.04	
8	6	8106	2073	9.06	667	630	0.34	488.17	
9	7	14387	3198	23.83	1036	990	0.61	out of mem.	
10	8	24142	4733	58.17	1541	1485	1.18	out of mem.	

5.7.4 Low rank approximation

Consider the problem of finding the nearest rank k tensor. Let $\mathbb{C}^{n_1 \times \cdots \times n_\ell}$ denote the set of tensors of order ℓ and dimensions (n_1, \dots, n_ℓ) and let $\mathbb{C}_{\leq k}^{n_1 \times \cdots \times n_\ell}$ be the closure of the space of tensors of rank at most k . Note that we can easily generate generic samples of rank k tensors. Given a real tensor $T \in \mathbb{R}^{n_1 \times \cdots \times n_\ell}$, the rank k approximation problem asks for the nearest point $X \in \mathbb{C}_{\leq k}^{n_1 \times \cdots \times n_\ell}$, i.e., the minimizer of $\|T - X\|^2$ where $\|\cdot\|$ denotes the norm of the vectorization.

Let $d := \lfloor k/2 \rfloor + 1$ and let $u(X)$ be the vector with all monomials of degree at most d . Denoting $\varsigma(X) := \|X\|^2$, we consider the following SDP relaxation:

$$\begin{aligned} & \max_{\gamma \in \mathbb{R}, Q \succeq 0} \quad \gamma \\ & \text{subject to} \quad (\|T - X_s\|^2 - \gamma) \varsigma(X_s)^{d-1} = Q \bullet u(X_s)u(X_s)^T, \quad \text{for } s = 1, \dots, S \\ & \quad \quad \quad X_s \in \mathbb{C}_{\leq k}^{n_1 \times \cdots \times n_\ell} \end{aligned}$$

We remark that computing the defining equations of the variety $\mathbb{C}_{\leq k}^{n_1 \times \cdots \times n_\ell}$ is very complicated [84]. This means that using traditional SOS methods is usually not possible.

Example 5.8 ([48], Ex 3). Let $T \in \mathbb{R}^{2 \times 2 \times 2 \times 2}$ be the tensor whose nonzero entries are

$$T_{1111} = 25.1, \quad T_{1121} = 0.3, \quad T_{1212} = 25.6, \quad T_{2111} = 0.3, \quad T_{2121} = 24.8, \quad T_{2222} = 23.$$

Consider the rank one approximation problem. Solving the above SDP ($d = 1, S = 49$) we obtain the lower bound 42.1216 on the minimum distance $\|T - X\|$. From the dual solution we recover the minimizer X^* , whose only nonzero entry is $X_{1212}^* = 25.6$.

Consider now the rank three approximation problem. The above SDP ($d = 2, S = 2422$) gives a lower bound of 23.0000. Again, we can recover the minimizer X^* , whose nonzero entries are

$$X_{1111}^* = 25.1, \quad X_{1121}^* = 0.3, \quad X_{1212}^* = 25.6, \quad X_{2111}^* = 0.3, \quad X_{2121}^* = 24.8$$

To see that X^* is rank three, note that after removing the entry 25.6 we are left with a 2×2 matrix.

5.7.5 Certifying infeasibility

Given a complex variety $\mathbf{V} \subseteq \mathbb{C}^n$ consider the problem of certifying that $\mathbf{V} \cap \mathbb{R}^n$ is empty. A *Positivstellensatz* infeasibility certificate consists in showing that the constant polynomial -1 is SOS on the variety \mathbf{V} [105]. For instance, if $\mathbf{V} = \{i, -i\} \subseteq \mathbb{C}$, a Positivstellensatz certificate is that $-1 = x^2$ on the variety \mathbf{V} . We take an approach from numerical algebraic geometry, where we first compute a numerical irreducible decomposition of \mathbf{V} , and then use sampling SOS to obtain the infeasibility certificate. For a given vector $u(x)$ the SDP problem to solve is:

$$\begin{aligned} & \text{find} && Q \succeq 0 \\ & \text{subject to} && -1 = Q \bullet u(z_s)u(z_s)^T, \quad \text{for } s = 1, \dots, S \\ & && z_s \in \mathbf{V} \end{aligned}$$

Example 5.9. Let $\mathbf{V} \subseteq \mathbb{C}^9$ be the positive dimensional part of the cyclic 9-roots problem. The cyclic 9-roots equations are:

$$\begin{aligned} & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 \\ & x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7 + x_7x_8 + x_8x_9 + x_9x_1 \\ & x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_6 + x_5x_6x_7 + x_6x_7x_8 + x_7x_8x_9 + x_8x_9x_1 + x_9x_1x_2 \\ & \vdots \\ & x_1x_2x_3x_4x_5x_6x_7x_8 + x_2x_3x_4x_5x_6x_7x_8x_9 + \dots + x_9x_1x_2x_3x_4x_5x_6x_7 \\ & x_1x_2x_3x_4x_5x_6x_7x_8x_9 - 1 \end{aligned}$$

The zero set of these equations consists of a two-dimensional variety \mathbf{V} of degree 18, and 6156 isolated solutions [59]. We remark that computing a Gröbner basis of these equations is very complicated unless its special structure is exploited. Indeed, Macaulay2 ran out of memory after 5 hours of computation.

We computed the irreducible decomposition of \mathbf{V} using Bertini; it took $2h\ 45m$ with the default parameters. The variety \mathbf{V} decomposes into three pairs of conjugate irreducible varieties (each pair of degree 6). For each component we proceed to compute a sampling 2-SOS certificate. We require 31 complex samples on each component, which we obtained from Bertini in less than a second. Note that the upper bound from (5.10) predicted $\frac{1}{2} \cdot \min\{\binom{13}{4}, 6\binom{6}{4}\} = 45$ samples. For each $j = 0, \dots, 5$ we solved the respective sampling SDP, obtaining an infeasibility

bility certificate of the form

$$-1 = (R_j u(x))^T (R_j u(x)), \quad \text{for } x \in \mathbf{V}_j.$$

This allows us to conclude that each irreducible component of \mathbf{V} is purely complex. For instance, for the first irreducible component \mathbf{V}_0 it takes only 0.74s to obtain the certificate shown in Figure 5-1.

5.7.6 Amoeba membership

The (unlog) amoeba $\mathcal{A}_{\mathbf{V}} \subseteq \mathbb{R}_+^n$ of a variety $\mathbf{V} \subseteq \mathbb{C}^n$ is the image of \mathbf{V} under the absolute value function, i.e., $\mathcal{A}_{\mathbf{V}} = \{|z| : z \in \mathbf{V}\}$. The amoeba membership problem is to determine whether some point $\lambda \in \mathbb{R}_+^n$ belongs to $\mathcal{A}_{\mathbf{V}}$. Theobald and De Wolff recently proposed the use of Positivstellensatz certificates to prove that $\lambda \notin \mathcal{A}_{\mathbf{V}}$ [131]. We now briefly describe this approach.

For some $f \in \mathbb{C}[z]$, let $\Re(f), \Im(f) \in \mathbb{R}[x, y]$ be such that

$$f(x + i y) = \Re(f)(x, y) + i \Im(f)(x, y).$$

Consider the following sets of equations in $\mathbb{R}[x, y]$:

$$J_{\mathbf{V}} := \{\Re(f_j), \Im(f_j)\}_{j=1}^m, \quad h_{\lambda} := \{x_i^2 + y_i^2 - \lambda_i^2\}_{i=1}^n$$

where f_j are the defining equations of \mathbf{V} . Theobald and De Wolff suggest computing a Gröbner basis of $J_{\mathbf{V}} \cup h_{\lambda}$ and then search for a Positivstellensatz infeasibility certificate.

Consider the following approach based on a set of samples $Z \subseteq \mathbf{V}$. Let $\hat{\mathbf{V}} \in \mathbb{C}^{2n}$ be the zero set of $J_{\mathbf{V}} \subseteq \mathbb{R}[x, y]$. Note that if $z \in \mathbf{V}$ then $(\Re(z), \Im(z)) \in \hat{\mathbf{V}}$. Thus, given some monomial vectors $u(x, y)$ and $v(x, y)$, we can formulate the following SDP:

$$\begin{aligned} & \text{find} && Q \succeq 0, C \\ & \text{subject to} && -1 = Q \bullet u(x_s, y_s)u(x_s, y_s)^T + h_{\lambda}(x_s, y_s)^T C v(x_s, y_s), \quad \text{for } s = 1, \dots, S \\ & && z_s = x_s + i y_s \in \mathbf{V} \end{aligned}$$

Example 5.10. Let $\mathbf{V} \subseteq \mathbb{C}^{nk}$ be the complex variety associated to the Stiefel manifold $St(k, \mathbb{R}^n)$. Let $\lambda = (1/n, 1/n, \dots, 1/n)$, and let us show that $\lambda \notin \mathcal{A}_{\mathbf{V}}$ using the SDP from above. We consider the degree 1 SOS relaxation for the case $n = 6, k = 4$. We require 1205 complex samples on \mathbf{V} , which we obtain using the Cayley parametrization. It takes only 0.79s to compute the Positivstellensatz certificate from below. On the other hand, Macaulay2 ran out of memory while computing a Gröbner basis of $J_{\mathbf{V}}$.

$$-1 = (Ru(x, y))^T (Ru(x, y)) - 1.2 \sum_{i=1}^6 h_i(x, y), \quad \text{for } (x, y) \in \hat{\mathbf{V}}$$

$$u(x, y) = (y_6, y_5, y_4, y_3, y_2, y_1) \quad h_i(x, y) = x_i^2 + y_i^2 - 1/n^2$$

$$R = \begin{bmatrix} 0.1765714 & 0.8458754 & -0.3371163 & -1.0598462 & 0.0269367 & 0.6447252 \\ 0.2893688 & 0.1328983 & -1.4142041 & 0.4346374 & 0.1677938 & -0.2855976 \\ -0.4505154 & -0.6521358 & -0.3240160 & 0.2748310 & -0.0022626 & 1.2614402 \\ 1.0819066 & 0.4199281 & 0.3317461 & 0.7231132 & -0.3725210 & 0.5304889 \\ 0.8377745 & -1.0150421 & -0.1600336 & -0.6991182 & -0.3744590 & -0.1150085 \\ 0.4579696 & -0.1868200 & 0.2138378 & -0.0250102 & 1.4464173 & 0.1299494 \end{bmatrix}$$

$$\begin{aligned}
u(x) &= (x_8^2, x_7x_9, x_6^2, x_5x_9, x_5x_7, x_4^2, x_3x_6, x_2x_7, x_2x_6, x_2^2, x_1x_3, x_1^2, x_8, x_7, x_6, x_5, x_3, x_1, 1) \\
R_0 &= \begin{bmatrix}
-0.9638686 & -0.3445318 & 0.8395791 & -1.9531033 & 0.6329543 & -0.0152284 & 0.0238164 & 0.4701138 & -1.9766327 & -0.8363703 \\
0.3474835 & -0.3993919 & 0.5501348 & -1.2198730 & -0.2314149 & 0.0354563 & 1.0086575 & 0.4018444 & 1.0316339 & -0.6193326 \\
0.0117704 & -0.5278490 & 0.6157589 & -0.3131173 & 0.2207819 & -0.0080541 & 0.4038186 & 0.1500184 & -0.2618475 & 0.3089739 \\
-0.0131866 & 0.1597228 & 0.1191077 & -0.1088218 & 0.0697348 & -0.1149430 & -0.5067092 & -0.1883695 & -0.5993569 & 0.0244521 \\
-0.4504113 & -0.0761266 & 0.0056933 & 0.1535964 & -0.0860039 & 0.0007534 & 0.1264270 & 0.0880389 & -0.0927822 & -0.1429983 \\
-0.0804265 & 0.1450405 & -0.0077285 & -0.1657304 & -0.3240087 & 0.0014097 & 0.0631496 & -0.4083965 & 0.0191162 & 0.0854950 \\
0.0192110 & 0.1019831 & -0.1208989 & -0.1821975 & -0.1203214 & 0.0405222 & 0.0595267 & -0.1921851 & -0.0669972 & -0.1710978 \\
-0.1242984 & 0.1450764 & -0.2725352 & -0.1145423 & 0.0498037 & 0.0036466 & -0.0705293 & 0.2012444 & 0.0873671 & -0.2016367 \\
-0.0724047 & -0.0072012 & -0.0659910 & 0.1174698 & -0.0830511 & 0.0339559 & 0.1872153 & -0.0010648 & -0.1488432 & 0.1014557 \\
-0.1440253 & 0.0026597 & -0.1198142 & 0.0147434 & 0.1104305 & 0.0249783 & 0.0246079 & -0.0286915 & -0.0633959 & 0.0786306 \\
0.0872131 & -0.0503333 & -0.0426310 & -0.0108485 & -0.1538320 & 0.0351373 & -0.0895051 & 0.0994606 & -0.0858176 & 0.0033518 \\
0.0508126 & 0.0850471 & -0.0581353 & -0.0654513 & 0.0413446 & -0.0119532 & 0.0757765 & 0.0459333 & -0.0169990 & 0.1009677 \\
0.0112157 & 0.0251923 & -0.0096306 & -0.0680984 & -0.0005761 & 0.0111481 & -0.0373533 & 0.0293210 & 0.0134771 & 0.1119457 \\
0.0240035 & -0.1089833 & -0.0750114 & -0.0316807 & 0.0593823 & -0.0045386 & -0.0385441 & -0.0541272 & 0.0162211 & 0.0093965 \\
0.0727416 & 0.0067146 & -0.0258618 & 0.0206007 & 0.0284529 & -0.0125337 & 0.0450957 & -0.0142651 & -0.0460162 & -0.0377493 \\
0.0018262 & 0.0096039 & 0.0092749 & -0.0098153 & 0.0116513 & 0.0124708 & -0.0166840 & -0.0307406 & 0.0039079 & -0.0005090 \\
0.0167276 & 0.0418916 & 0.0339853 & 0.0127189 & 0.0353915 & 0.0352643 & -0.0230590 & 0.0037635 & -0.0020096 & -0.0118329 \\
-0.0000118 & 0.0028062 & 0.0010184 & -0.0000054 & 0.0000008 & -0.0076840 & -0.0004971 & 0.0000191 & 0.0000175 & 0.0000097 \\
0.0000403 & 0.0016845 & -0.0003547 & -0.0000443 & -0.0000077 & 0.0102089 & 0.0023837 & -0.0001760 & -0.0000343 & -0.0000479 \\
0.1130541 & 0.0243746 & -1.0601901 & -0.5184653 & 0.5389394 & -0.5290480 & 1.4666654 & -0.3021666 & -0.0722647 & \\
-0.4838530 & 0.0446110 & 0.0443714 & 0.0400056 & -1.4678116 & -0.8155807 & -0.3859305 & 0.4715178 & 0.0326426 & \\
-0.1549109 & 0.0903295 & 0.6966522 & 0.1005015 & 0.1763720 & 1.0574739 & -0.3501351 & -0.0462028 & 0.1140547 & \\
0.5739993 & -0.0684158 & 0.3571626 & 0.0861604 & -0.6655387 & -0.1886137 & -0.4910517 & 0.1000489 & 0.1425230 & \\
-0.0397509 & 0.0175350 & -0.4837186 & 0.1313369 & 0.0071916 & 0.1063667 & -0.5799628 & 0.0186656 & -0.1366172 & \\
0.0288713 & -0.0270858 & -0.1711768 & 0.0516328 & -0.2256508 & 0.3277098 & 0.1917983 & -0.0453268 & -0.0388412 & \\
-0.1128892 & 0.0088688 & 0.1873327 & 0.1850600 & 0.2445545 & -0.1280363 & -0.1585725 & -0.1034274 & 0.0225831 & \\
0.0175299 & -0.0263491 & 0.0802817 & -0.1067258 & -0.0673210 & 0.2239093 & 0.0003892 & -0.0262654 & 0.1438365 & \\
-0.1171573 & 0.0113433 & 0.0483356 & -0.1727556 & -0.1006414 & -0.0966047 & -0.0033401 & -0.1640771 & 0.2015432 & \\
-0.1481463 & 0.0152680 & 0.0863339 & 0.1057026 & -0.0780024 & -0.0238758 & 0.0753908 & 0.1662874 & -0.0984320 & \\
-0.0868955 & 0.0296646 & -0.0024721 & -0.0139792 & -0.0210549 & 0.0241345 & -0.0051360 & 0.0220213 & -0.0734852 & \\
0.0428166 & -0.0263798 & -0.0151096 & -0.0341225 & 0.0015975 & -0.0045648 & -0.0459362 & -0.0584079 & -0.1230225 & \\
-0.0239812 & -0.0019790 & -0.0630874 & 0.0477949 & 0.0312372 & -0.0118692 & -0.0260705 & 0.0443067 & 0.1156214 & \\
-0.0084732 & 0.0434794 & -0.0322293 & -0.0031909 & -0.0252367 & 0.0031306 & -0.0173622 & -0.0663103 & -0.0011153 & \\
-0.0013241 & -0.0085007 & -0.0350008 & 0.0115394 & -0.0048681 & 0.0243468 & -0.0006443 & 0.0436652 & 0.0255582 & \\
-0.0261973 & -0.0090256 & -0.0019749 & -0.0604685 & 0.0112659 & 0.0022552 & -0.0232959 & 0.0276527 & -0.0063107 & \\
-0.0469611 & 0.0047957 & -0.0132026 & 0.0243877 & -0.0253729 & 0.0101839 & -0.0006742 & -0.0451327 & -0.0022650 & \\
-0.0020972 & 0.0071188 & -0.0000255 & -0.0008521 & 0.0000673 & 0.0000274 & -0.0002656 & 0.0000749 & -0.0000434 & \\
0.0054816 & 0.0120243 & 0.0000808 & -0.0010943 & 0.0006406 & 0.0000603 & 0.0000083 & 0.0026580 & 0.0001099 &
\end{bmatrix}
\end{aligned}$$

Figure 5-1: Positivstellensatz infeasibility certificate for the cyclic 9-roots problem.

Chapter 6

Local stability of semidefinite relaxations

Consider semidefinite programming (SDP) relaxations of a parametric family of polynomial optimization problems. This chapter concerns the stability analysis of these relaxations. We assume that the SDP relaxation is exact for a nominal value of the parameters, and analyze the behavior of the relaxation nearby this nominal parameter. The content of this chapter is based on [37].

6.1 Introduction

Polynomial optimization problems are hard to solve in general, and SDP relaxations have become a standard approach to tackle them. We focus in this chapter on optimization problems over algebraic sets that involve *parameters*, and we assume that for a given value of the parameters the SDP relaxation is *tight*, i.e., it correctly solves the problem. We study the behavior of the relaxation under *small perturbations* of the special parameters, identifying sufficient conditions under which the relaxation continues to be tight.

An important class of problems we consider is that of finding the point y^* on an algebraic variety $Y \subseteq \mathbb{R}^n$ that minimizes a loss function, e.g., the Euclidean distance $\|y - \theta\|$ to some given θ . These problems arise often in statistical estimation problems, such as low rank approximation, camera triangulation, rotation synchronization, approximate matrix completion

and approximate GCD. In these problems the case where $\theta \in Y$ is trivial, and SDP relaxations are tight. Our methods allow a systematic analysis of the behavior of these relaxations when θ is close to Y . In particular, we recover tightness results previously shown in the context of camera triangulation [1] and rotation synchronization [112].

Since polynomial optimization problems can always be rewritten as *quadratically constrained quadratic programs* (QCQP's) (see e.g., [97, 123]), in the rest of this chapter we will only consider QCQP's. As illustrated in the next example, there is a natural SDP relaxation of a QCQP; namely, its Lagrangian dual. More generally, SDP relaxations coming from the SOS method correspond to the Lagrangian dual of a suitable QCQP.

Example 6.1 (Nearest point to the twisted cubic). Let $Y := \{(t, t^2, t^3) : t \in \mathbb{R}\}$ be the twisted cubic curve in \mathbb{R}^3 . Given $\theta \in \mathbb{R}^3$, the problem of finding the nearest point in Y to θ can be phrased as:

$$\min_{y \in Y} \|y - \theta\|^2, \quad \text{where} \quad Y = \{y \in \mathbb{R}^3 : y_2 = y_1^2, y_3 = y_1 y_2\}. \quad (6.1)$$

The above is a QCQP, and its Lagrangian dual is the following SDP:

$$\max_{\gamma, \lambda_1, \lambda_2 \in \mathbb{R}} \quad \gamma, \quad \text{s.t.} \quad \begin{pmatrix} \gamma + \|\theta\|^2 & -\theta_1 & \lambda_1 - \theta_2 & \lambda_2 - \theta_3 \\ -\theta_1 & 1 - 2\lambda_1 & -\lambda_2 & 0 \\ \lambda_1 - \theta_2 & -\lambda_2 & 1 & 0 \\ \lambda_2 - \theta_3 & 0 & 0 & 1 \end{pmatrix} \succeq 0. \quad (6.2)$$

We will show that when θ is sufficiently close to Y the above relaxation is tight. Equivalently, the *duality-gap* $\text{val}(6.1) - \text{val}(6.2)$ is zero in a neighborhood of Y . This is illustrated in Figure 6-1, by showing the projection of Y onto the $y_1 y_3$ -plane, and the duality-gap for parameters θ of the form $(\theta_1, \theta_1^2, \theta_3)$. Besides the fact that there is zero-duality-gap when θ is close to Y , we will also see that we can *recover the minimizer* of (6.1) from the SDP.

We shall say that a variety Y is *quadratic* if it is the zero set of some quadratic equations $f_i \in \mathbb{R}[y]$ (e.g., the twisted cubic (6.1)). As stated next, the problem of finding the nearest point from θ to a quadratic variety Y has a tight SDP relaxation when θ is close to Y . More precisely, we show that if $\bar{\theta} \in Y$ satisfies a certain regularity condition, then the relaxation is tight when θ deviates slightly from $\bar{\theta}$.

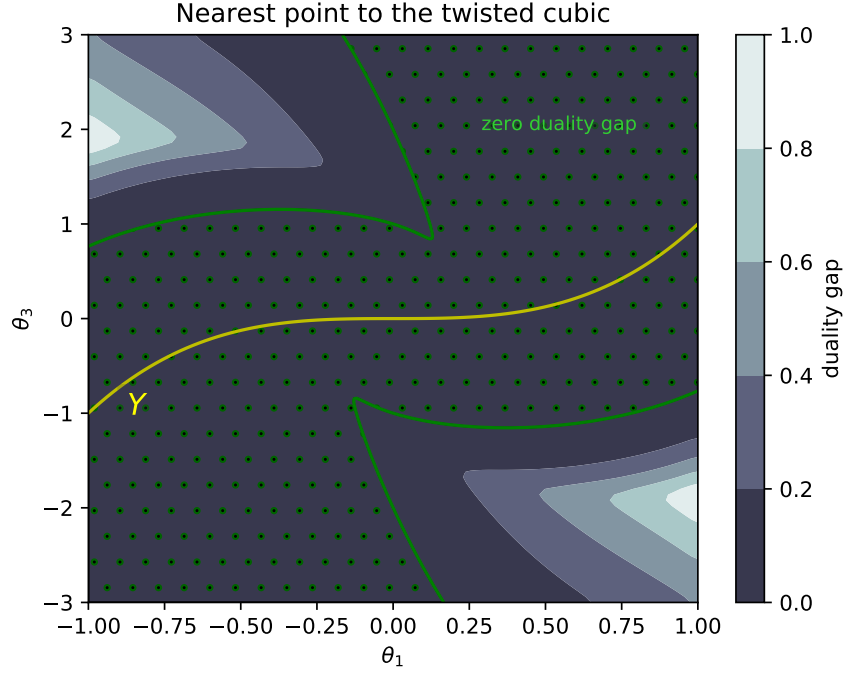


Figure 6-1: Duality gap in problem (6.1) for parameters θ of the form $(\theta_1, \theta_1^2, \theta_3)$. There is no duality-gap in the dotted region.

Theorem 6.1 (Nearest point to a quadratic variety). *Consider the problem*

$$\min_{y \in Y} \|y - \theta\|^2, \quad \text{where} \quad Y := \{y \in \mathbb{R}^n : f_1(y) = \cdots = f_m(y) = 0\}, \quad f_i \text{ quadratic.}$$

Let $\bar{\theta} \in Y$ be such that $\text{rk}(\nabla f(\bar{\theta})) = n - \dim_{\bar{\theta}} Y$. Then there is zero-duality-gap for any $\theta \in \mathbb{R}^n$ that is sufficiently close to $\bar{\theta}$.

Theorem 6.1 is a special case of a more general result, Theorem 6.8, that we will prove in Section 6.4. Both of these theorems rely on the objective function being strictly convex. These results can be readily applied to many estimation problems, such as rank one tensor approximation, the triangulation problem, and rotation synchronization (see Section 6.7.1). The proof is relatively elementary.

Unfortunately, having strictly convex objective is a strong requirement that is often not satisfied. In particular, the nearest point problem to a cubic curve cannot be written as a QCQP with a strictly convex objective.

Example 6.2. Consider the nearest point problem to the plane curve $Y := \{y \in \mathbb{R}^2 : y_2^2 = y_1^3\}$.

This curve is not defined by quadratic equations in (y_1, y_2) . But by introducing the auxiliary variable z_1 we can rewrite the nearest point problem as a QCQP:

$$\min_{z_1 \in \mathbb{R}, y \in \mathbb{R}^2} \|y - \theta\|^2, \quad \text{s.t.} \quad y_2 = y_1 z_1, \quad y_1 = z_1^2, \quad y_2 z_1 = y_1^2. \quad (6.3)$$

Note that the presence of the “auxiliary” variable z_1 means that the objective function is not strictly convex. Nonetheless, we will see that the resulting QCQP has zero-duality-gap when θ is close enough to the curve Y .

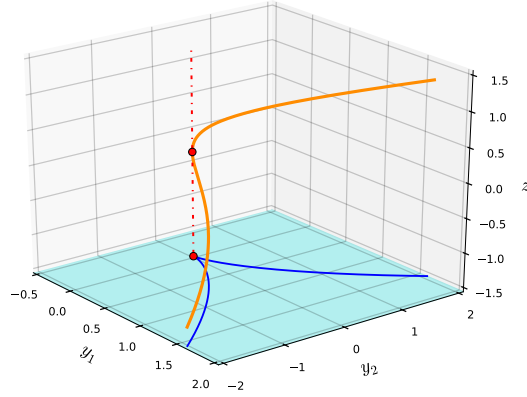


Figure 6-2: The plane curve $y_2^2 = y_1^3$ is the projection of the twisted cubic.

In this chapter we consider a general family of QCQP’s parametrized by θ , and we assume that there is zero-duality-gap for a fixed parameter $\bar{\theta}$. The main contribution of this chapter is to identify sufficient conditions that guarantee zero-duality-gap as $\theta \rightarrow \bar{\theta}$. Our results, in particular, can be used in nearest point problems to varieties, such as the plane curve of Example 6.2. Note that the conditions we require are nontrivial, and it is easy to find problems with positive-duality-gap for θ arbitrarily close to $\bar{\theta}$ (see Example 6.7).

The organization of this chapter is as follows. In Section 6.2 we formalize our problem of study. In Section 6.3 we introduce the main tools needed for our results. In Section 6.4 we prove Theorem 6.8, a generalization of Theorem 6.1. In Section 6.5 we state Theorem 6.14, the most general result of this chapter. In Section 6.6 we prove Theorem 6.14 using the implicit function theorem. In Section 6.7 we show some applications of our results.

Related work

Several results concerning tightness of SDP relaxations exist. There are two main kinds of results. The first class is based on structural assumptions on the equations (or the feasible set). In particular, the Lagrangian dual of several classes of QCQP's are exact [16, 81, 150, 153], such as trust-region problems (there is a single ball constraint) [126], and S-lemma type problems [107]. Similarly, certain classes of combinatorial optimization problems are known to have tight SDP relaxations [66, 93]. We point out that there has been plenty of research in this area, and our references are far from extensive.

A second class of results, typically tailored to statistical estimation problems, shows tightness of SDP relaxations under low noise assumptions. In particular, it was shown in [1] that the first SOS relaxation of the triangulation problem is tight in the low noise regime under genericity assumptions. Similarly, it has been shown that the first SOS relaxation of the rotation synchronization problem is tight under low noise [63, 112, 146]. We generalize these types of results by systematically studying the behavior of SDP relaxations under small perturbations of the problem.

Perturbation analysis of nonlinear optimization problems is a well studied subject [28, 60, 90]. In particular, sufficient conditions for continuity and differentiability of the optimal value/solutions are known [28, 60]. Similarly, the Lipschitzian stability of general optimization problems, together with concepts such as tilt/full stability, has received a lot of attention [90, 100]. The stability analysis of special classes of nonlinear programs, such as semidefinite programs [28, §5.3.6] and convex quadratic programs [69], has also been considered. In this chapter we apply these techniques to the study of SDP relaxations of equality constrained QCQP's. As opposed to previous work we are mostly interested in the correctness of the SDP relaxation, i.e., whether we can recover an optimal solution of the QCQP from the relaxation.

6.2 Formalizing the problem

In this section we formalize our problem of study. We also begin to explain the role of Lagrange multipliers in certifying zero-duality-gap.

6.2.1 Problem statement

Consider a family of QCQP's parametrized by $\theta \in \Theta$:

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & g_\theta(x) \\ & h_\theta^i(x) = 0, \quad \text{for } i = 1, \dots, m \end{aligned} \tag{P_\theta}$$

where g_θ, h_θ^i are *quadratic* in x , and the dependence on θ is *continuously differentiable*. Note that g_θ may not be convex. To simplify the notation, we will focus in this chapter on the *homogeneous* case. By that we mean that g_θ, h_θ^i do not have linear terms, i.e.,

$$g_\theta(x) := x^T G_\theta x, \quad h_\theta^i(x) = x^T H_\theta^i x + b_i \quad \text{for some } G_\theta, H_\theta^i \in \mathcal{S}^N, b_i \in \mathbb{R},$$

and that at least one b_i is nonzero.

Remark 6.1 (Sign invariance). Since a homogeneous problem is invariant under the involution $x \mapsto -x$, we can only recover the solution up to sign changes.

Remark 6.2 (Homogenization). We can always get rid of linear terms by introducing a *homogenizing* variable z_0 . For instance, the homogenized form of (6.1) is:

$$\min_{z_0 \in \mathbb{R}, y \in \mathbb{R}^3} \quad \|y - \theta z_0\|^2 \quad \text{s.t.} \quad z_0^2 = 1, \quad y_2 z_0 = y_1^2, \quad y_3 z_0 = y_1 y_2.$$

Recall that a QCQP has an associated dual pair of SDP relaxations. In the case of (P_θ) we get:

$$\begin{aligned} \min_{S \in \mathcal{S}^N} \quad & G_\theta \bullet S \\ & H_\theta^i \bullet S + b_i = 0, \quad i = 1, \dots, m \quad (P_\theta^*) \\ & S \succeq 0 \end{aligned} \quad \begin{aligned} \max_{\lambda \in \mathbb{R}^m} \quad & d(\lambda) := \sum_i \lambda_i b_i \\ & \mathcal{Q}_\theta(\lambda) \succeq 0 \end{aligned} \tag{D_\theta}$$

where $\mathcal{Q}_\theta(\lambda)$ is the Hessian of the Lagrangian function:

$$\mathcal{Q}_\theta(\lambda) := G_\theta + \sum_i \lambda_i H_\theta^i \in \mathcal{S}^N.$$

Note that (D_θ) is the Lagrangian dual of (P_θ) , and that the inequalities $\text{val}(P_\theta) \geq \text{val}(P_\theta^*) \geq$

$\text{val}(D_\theta)$ always hold. We are concerned here with the *zero-duality-gap* condition $\text{val}(P_\theta) = \text{val}(D_\theta)$.

Throughout this chapter we denote by $\bar{\theta}$ the nominal value of the parameter θ , such that $(P_{\bar{\theta}})$ has zero-duality-gap, and we investigate the behavior of the relaxation under small perturbations around $\bar{\theta}$.

6.2.2 Lagrange multipliers and zero-duality-gap

Given a feasible solution x_θ of (P_θ) , recall that $\lambda \in \mathbb{R}^m$ is a *Lagrange multiplier* at x_θ if

$$\lambda^T \nabla h_\theta(x_\theta) = -\nabla g_\theta(x_\theta) \iff \sum_i \lambda_i H_\theta^i x_\theta = -G_\theta x_\theta \iff \mathcal{Q}_\theta(\lambda) x_\theta = 0.$$

We denote by $\Lambda_\theta(x_\theta)$ the affine space of *Lagrange multipliers* at x_θ .

Lemma 6.2. *Let $x_\theta \in \mathbb{R}^N, \lambda \in \mathbb{R}^m$. Then x_θ is optimal to (P_θ) and λ is optimal to (D_θ) with $\text{val}(P_\theta) = \text{val}(D_\theta)$ if and only if the following conditions hold:*

6.2(i) $h_\theta(x_\theta) = 0$ (*primal feasibility*).

6.2(ii) $\mathcal{Q}_\theta(\lambda) \succeq 0$ (*dual feasibility*).

6.2(iii) $\lambda \in \Lambda_\theta(x_\theta)$ (*complementarity*).

If furthermore $\mathcal{Q}_\theta(\lambda)$ has corank-one, then the unique optimal solution of (P_θ^) is $x_\theta x_\theta^T$.*

Proof. Assume that the conditions 6.2(i-iii) hold. Since $\mathcal{Q}_\theta(\lambda) x_\theta = 0$, then

$$0 = x_\theta^T \mathcal{Q}_\theta(\lambda) x_\theta = x_\theta^T G_\theta x_\theta + \sum_i \lambda_i x_\theta^T H_\theta^i x_\theta = g_\theta(x_\theta) - d(\lambda), \quad (6.4)$$

and thus (x_θ, λ) are primal/dual optimal. The converse is similar.

Assume now that $\mathcal{Q}_\theta(\lambda)$ has corank-one. Let S^* be an optimal solution of (P_θ^*) . By complementary slackness we have $\mathcal{Q}_\theta(\lambda) \bullet S^* = 0$, and since both matrices lie in \mathcal{S}_+^N , then $\text{rk}(\mathcal{Q}_\theta(\lambda)) + \text{rk}(S^*) \leq N$. Thus, any optimal solution of (P_θ^*) has rank one. It follows that there is a unique optimal, namely $S^* = x_\theta x_\theta^T$. \square

The above lemma gives necessary and sufficient conditions for having zero-duality-gap. Moreover, if an additional assumption holds (corank-one Hessian), then we can also recover

the minimizer of the QCQP from the SDP. We point out that items 6.2(i-iii) correspond to the optimality conditions of a nonlinear program, and that variants of Lemma 6.2 have appeared before (e.g., [1, Thm 2]).

As a simple application of Lemma 6.2, let us see that the nearest point problem to a variety Y has zero-duality-gap for any $\bar{\theta} \in Y$ (in fact, the associated multiplier is $\bar{\lambda} = 0$).

Proposition 6.3 (Nearest point problem). *Consider the problem:*

$$\min_{y \in Y} \|y - \theta\|^2, \quad \text{where} \quad Y := \{y \in \mathbb{R}^n : \exists z' \in \mathbb{R}^{k-1} \text{ s.t. } f_i(z', y) = 0, 1 \leq i \leq m\}$$

where f_i are quadratic polynomials, and z' is a set of auxiliary variables. There is zero-duality-gap for any $\bar{\theta} \in Y$.

Proof. In order to apply Lemma 6.2, we first need to consider the homogeneous version of the problem:

$$\min_{z \in \mathbb{R}^k, y \in \mathbb{R}^n} \|y - \theta z_0\|^2 \quad \text{s.t.} \quad z_0^2 = 1, \quad h_i(z, y) = 0, \quad 1 \leq i \leq m,$$

where $z = (z_0, z')$, and h_i is the homogenization of f_i with respect to z_0 :

$$h_i(z, y) = h_i(z_0, z', y) := z_0^2 f_i(z'/z_0, y/z_0).$$

Let $\bar{x} = (\bar{z}, \bar{\theta})$ be the optimal solution, and let $g_{\bar{\theta}}(z, y) := \|y - \theta z_0\|^2$ be the objective function. Notice that $\bar{\lambda} = 0$ is a Lagrange multiplier at \bar{x} since $\nabla g_{\bar{\theta}}(\bar{x}) = 0$. Moreover, $\bar{\lambda} = 0$ is dual feasible since $g_{\bar{\theta}}$ is convex and thus $\mathcal{Q}_{\bar{\theta}}(\bar{\lambda}) = G_{\bar{\theta}} \succeq 0$. Then there is zero-duality-gap by Lemma 6.2. \square

6.3 Continuity of Lagrange multipliers

In Lemma 6.2 we identified necessary and sufficient conditions for zero-duality-gap at a fixed parameter $\bar{\theta}$. In this section we will study conditions under which we continue to get zero-duality-gap for parameters θ close to $\bar{\theta}$.

Throughout this section we let $\bar{\theta}$ be a zero-duality gap parameter, and \bar{x} be the minimizer of $(P_{\bar{\theta}})$, which we assume is *unique* (up to sign). Observe that $\bar{x} \neq 0$ since we assumed that

the constant term b_i of some $h_\theta^i(x)$ is nonzero. Consider the *Lagrange multiplier mapping*:

$$\begin{aligned}\mathfrak{L} : \Theta &\rightrightarrows \mathbb{R}^N \times \mathbb{R}^m, & \theta &\mapsto \{(x_\theta, \lambda_\theta) : x_\theta \text{ primal feasible, } \lambda_\theta \in \Lambda_\theta(x_\theta)\} \\ & & &= \{(x_\theta, \lambda_\theta) : h_\theta(x_\theta) = 0, \mathcal{Q}_\theta(\lambda_\theta)x_\theta = 0\}.\end{aligned}\tag{6.5}$$

As we will see, *continuity* properties of \mathfrak{L} play a crucial role in our analysis.

6.3.1 A first stability result

Our first stability result relies on the existence of a dual optimal solution λ satisfying the following two properties (recall that \bar{x} is fixed).

Assumption C1H (corank-one Hessian). Let λ be dual optimal at $\bar{\theta}$. The matrix $\mathcal{Q}_{\bar{\theta}}(\lambda)$ has corank-one, or equivalently, strict complementarity holds for the dual pair of SDP's.

Assumption WC (weak continuity of multipliers). Let $\bar{\ell} = (\bar{x}, \lambda) \in \mathfrak{L}(\bar{\theta})$ be a Lagrange multiplier pair. There exists $\ell_\theta \in \mathfrak{L}(\theta)$ such that $\ell_\theta \rightarrow \bar{\ell}$ as $\theta \rightarrow \bar{\theta}$.

Proposition 6.4 (C1H + WC \implies stability). *Let (\bar{x}, λ) be primal/dual optimal at $\bar{\theta}$, such that Assumptions C1H and WC hold. Then for any θ sufficiently close to $\bar{\theta}$ there is zero-duality-gap and (P_θ^*) recovers the minimizer.*

Proof. By Assumption WC, there exist $(x_\theta, \lambda_\theta)$ with: x_θ feasible for (P_θ) , $\lambda_\theta \in \Lambda_\theta(x_\theta)$, and $(x_\theta, \lambda_\theta) \rightarrow (\bar{x}, \lambda)$ as $\theta \rightarrow \bar{\theta}$. It follows that $\mathcal{Q}_\theta(\lambda_\theta) \rightarrow \mathcal{Q}_{\bar{\theta}}(\lambda)$, since g_θ and h_θ^i depend continuously on θ . Notice that $\mathcal{Q}_\theta(\lambda_\theta)$ has a 0-eigenvalue since $\mathcal{Q}_\theta(\lambda_\theta)x_\theta = 0$. Let us see that, as $\theta \rightarrow \bar{\theta}$, the remaining eigenvalues of $\mathcal{Q}_\theta(\lambda_\theta)$ are positive. This would conclude the proof because of Lemma 6.2. By assumption, $\mathcal{Q}_{\bar{\theta}}(\lambda)$ has $N - 1$ positive eigenvalues. Since $\mathcal{Q}_\theta(\lambda_\theta) \rightarrow \mathcal{Q}_{\bar{\theta}}(\lambda)$, and by the continuity of the eigenvalues, we conclude that $\mathcal{Q}_\theta(\lambda_\theta)$ also has $N - 1$ positive eigenvalues when $\theta \rightarrow \bar{\theta}$, as wanted. \square

Proposition 6.4 establishes conditions to guarantee zero-duality-gap nearby $\bar{\theta}$. We will see a few more stability results later in this chapter, but all of them rely implicitly on Proposition 6.4. This is illustrated in Figure 6-3.

While it is easy to check whether a given matrix has corank-one, it is typically not easy to find a dual variable λ that satisfies Assumption C1H. This is similar to the second order

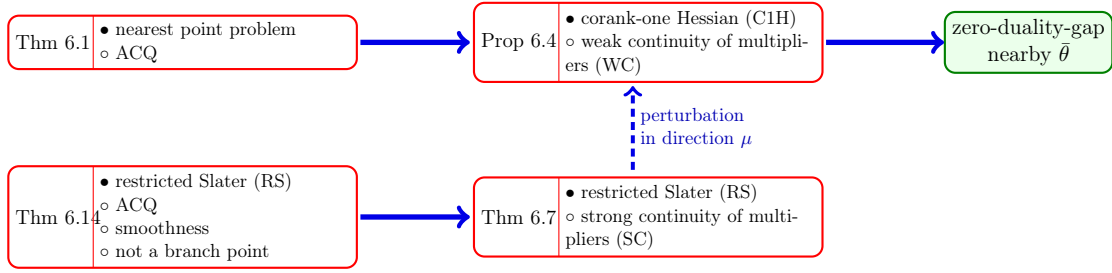


Figure 6-3: Conditions that imply zero-duality-gap nearby $\bar{\theta}$. The main assumption in each box is denoted with \bullet , and regularity assumptions are marked with \circ .

optimality conditions in non-linear programming which is also stated in terms of the existence of a Lagrange multiplier that satisfies certain conditions, without any procedure to find such a multiplier

In practice there is often a natural choice $\bar{\lambda}$ for the dual variables, such as $\bar{\lambda} = 0$ in nearest point problems (Proposition 6.3). If Assumptions C1H and WC hold for such a nominal $\bar{\lambda}$, we will be certain that there is zero-duality-gap. Unfortunately, it is often the case that the Hessian matrix $\mathcal{Q}_{\bar{\theta}}(\bar{\lambda})$ has rank less than $N - 1$, violating Assumption C1H. This situation arises, for instance, in nearest point problems to non-quadratic varieties. In the remainder of this section we will analyze how to establish stability in these cases.

6.3.2 Restricted Slater

Assume we are given a dual variable $\bar{\lambda}$ for which Assumption C1H (corank-one Hessian) is not satisfied. The following “Slater-type” condition allows us to find a dual variable λ' for which Assumption C1H holds.

Assumption RS (restricted Slater). Let $(\bar{x}, \bar{\lambda})$ be primal/dual optimal at $\bar{\theta}$, and consider the subspace

$$V := \{v \in \mathbb{R}^N : \mathcal{Q}_{\bar{\theta}}(\bar{\lambda})v = 0, \bar{x}^T v = 0\}.$$

There exists $\mu \in \mathbb{R}^m$ such that the quadratic function $\Psi_{\mu}(x) := \sum_i \mu_i h_{\bar{\theta}}^i(x)$ satisfies: $\nabla \Psi_{\mu}(\bar{x}) = 0$, and Ψ_{μ} is strictly convex on V .

Importantly, Assumption RS can be efficiently checked. Indeed, by restating the strict convexity of Ψ_{μ} in terms of its Hessian, Assumption RS corresponds to the strict feasibility of an

SDP (find μ s.t. $\sum_i \mu_i H_{\bar{\theta}}^i \bar{x} = 0$, $(\sum_i \mu_i H_{\bar{\theta}}^i)|_V \succ 0$). We will elaborate more on Assumption RS in Section 6.5, illustrating concrete examples that satisfy it.

As seen next, the vector μ from Assumption RS gives us a *direction* along which we can perturb the problem to go back to the corank-one case.

Lemma 6.5 (RS \implies “nearby” C1H). *Let $(\bar{x}, \bar{\lambda})$ be primal/dual optimal at $\bar{\theta}$, and let μ be as in Assumption RS. Then there is an $\epsilon > 0$ such that $\lambda_t := \bar{\lambda} + t\mu$ is dual optimal and $\text{corank } \mathcal{Q}_{\bar{\theta}}(\lambda_t) = 1$ for any $0 < t < \epsilon$.*

The proof of Lemma 6.5 relies on the following well-known lemma.

Lemma 6.6 (Finsler [61]). *Let $A, B \in \mathcal{S}^n$, $A \succeq 0$ be such that $v^T B v > 0$ for every $v \neq 0$ with $Av = 0$. Then there is some $\epsilon > 0$ such that $A + tB \succ 0$ for any $0 < t < \epsilon$.*

Proof of Lemma 6.5. Since $(\bar{x}, \bar{\lambda})$ is primal/dual optimal, it satisfies conditions 6.2(i-iii). We need to show that (\bar{x}, λ_t) also satisfies 6.2(i-iii), and that $\text{corank } \mathcal{Q}_{\bar{\theta}}(\lambda_t) = 1$. It is easy to see that $\lambda_t \in \Lambda_{\bar{\theta}}(\bar{x})$, so it remains to show that $\mathcal{Q}_{\bar{\theta}}(\lambda_t) \succeq 0$ and has corank-one. Let $A := \mathcal{Q}_{\bar{\theta}}(\bar{\lambda}) \succeq 0$ and $B := \sum_i \mu_i H_{\bar{\theta}}^i$. Since $\bar{\lambda} \in \Lambda_{\bar{\theta}}(\bar{x})$ then $A\bar{x} = 0$. Similarly, since $\sum_i \mu_i H_{\bar{\theta}}^i \bar{x} = 0$ then $B\bar{x} = 0$. We may assume WLOG that $\bar{x} = (1, 0^{N-1})$. Then $A = \begin{pmatrix} 0 & 0 \\ 0 & A' \end{pmatrix}$, $B = \begin{pmatrix} 0 & 0 \\ 0 & B' \end{pmatrix}$, where $A', B' \in \mathcal{S}^{N-1}$, $A' \succeq 0$. Note that the strict convexity condition in Assumption RS means that $v^T B' v > 0$ for every nonzero $v \in \mathbb{R}^{N-1}$ with $A'v = 0$. From Lemma 6.6 we know that $A' + tB' \succ 0$ for all $0 < t < \epsilon$. Therefore, $A + tB \succeq 0$ and has corank-one for $0 < t < \epsilon$, as wanted. \square

6.3.3 Stability under Assumption RS

Lemma 6.5 allows us to find some new dual variables λ' for which Assumption C1H is satisfied. In order to use Proposition 6.4 and conclude stability, it remains to see that λ' satisfies Assumption WC (weak continuity). This requires a *stronger* continuity condition on the original pair $(\bar{x}, \bar{\lambda})$. Before stating this assumption, we first recall a well-studied notion of continuity for set-valued-mappings. We refer to [4, 108] for a detailed introduction to set-valued-mappings.

Definition 6.1 (Painlevé-Kuratowski continuity). Let $\mathfrak{F} : \Theta \rightrightarrows \mathbb{R}^n$ be a set-valued mapping, and assume that each $\mathfrak{F}(\theta) \subseteq \mathbb{R}^n$ is nonempty. A *selection* of \mathfrak{F} is an assignment $y_\theta \in \mathfrak{F}(\theta)$ for

each $\theta \in \Theta$. The *inner limit* of \mathfrak{F} at $\bar{\theta}$ consists of all limits of selections $\{y_\theta\}_\theta$, i.e.,

$$\liminf_{\theta \rightarrow \bar{\theta}} \mathfrak{F}(\theta) := \{y \in \mathbb{R}^n : \exists y_\theta \in \mathfrak{F}(\theta) \text{ s.t. } y_\theta \xrightarrow{\theta \rightarrow \bar{\theta}} y\},$$

The *outer limit* of \mathfrak{F} at $\bar{\theta}$ consists of all cluster points of selections $\{y_\theta\}_\theta$, i.e.,

$$\limsup_{\theta \rightarrow \bar{\theta}} \mathfrak{F}(\theta) := \{y \in \mathbb{R}^n : \exists \theta_i \xrightarrow{i \rightarrow \infty} \bar{\theta}, \exists y_i \in \mathfrak{F}(\theta_i) \text{ s.t. } y_i \xrightarrow{i \rightarrow \infty} y\}.$$

The inner and outer limits are always closed sets that sandwich the closure of $\mathfrak{F}(\bar{\theta})$:

$$\liminf_{\theta \rightarrow \bar{\theta}} \mathfrak{F}(\theta) \subseteq \text{cl}(\mathfrak{F}(\bar{\theta})) \subseteq \limsup_{\theta \rightarrow \bar{\theta}} \mathfrak{F}(\theta).$$

\mathfrak{F} is (Painlevé-Kuratowski) *continuous*¹ at $\bar{\theta}$ if $\mathfrak{F}(\bar{\theta}) = \liminf_{\theta \rightarrow \bar{\theta}} \mathfrak{F}(\theta) = \limsup_{\theta \rightarrow \bar{\theta}} \mathfrak{F}(\theta)$.

Remark 6.3. When \mathfrak{F} is defined by continuous equations, such as \mathfrak{L} , then the equation $\mathfrak{F}(\bar{\theta}) = \limsup_{\theta \rightarrow \bar{\theta}} \mathfrak{F}(\theta)$ always holds [108, Ex 5.8]. Consequently, in this chapter we will focus our attention only on the inner limit.

Remark 6.4. Note that Assumption WC is simply that $\bar{\ell} \in \liminf_{\theta \rightarrow \bar{\theta}} \mathfrak{L}(\theta)$.

Example 6.3. Consider the mapping

$$\mathfrak{F} : \mathbb{R} \rightrightarrows \mathbb{R}, \quad \theta \mapsto \begin{cases} \{0\}, & \text{if } \theta < 0 \\ [-1, 1], & \text{if } \theta \geq 0 \end{cases}$$

This mapping is continuous at any $\theta \neq 0$. Observe that $\liminf_{\theta \rightarrow 0} \mathfrak{F}(\theta) = \{0\}$ and $\limsup_{\theta \rightarrow 0} \mathfrak{F}(\theta) = [-1, 1]$. Thus \mathfrak{F} is not continuous at 0.

Assumption SC (strong continuity of multipliers). Let $\bar{\ell} \in \mathfrak{L}(\bar{\theta})$ be a Lagrange multiplier pair. There exists a closed neighborhood $U \ni \bar{\ell}$ such that $\mathfrak{L}(\bar{\theta}) \cap U \subseteq \liminf_{\theta \rightarrow \bar{\theta}} \mathfrak{L}(\theta)$, or equivalently, such that the mapping $\theta \mapsto \mathfrak{L}(\theta) \cap U$ is continuous at $\bar{\theta}$.

Theorem 6.7 (RS + SC \implies stability). *Let $(\bar{x}, \bar{\lambda})$ be primal/dual optimal at $\bar{\theta}$, such that*

¹ Although other notions of (set-valued-mapping) continuity exist, they agree for the case of compact valued mappings [108]. Since the analysis done in this chapter is local, we may always restrict the range to some closed ball. Hence, we may ignore this distinction in this chapter.

Assumptions RS and SC hold. Then for any θ sufficiently close to $\bar{\theta}$ there is zero-duality-gap and (P_θ^) recovers the minimizer.*

Proof. Let $U \ni \bar{\ell}$ be as in Assumption SC. By Lemma 6.5, there is a dual optimal λ' with $\mathcal{Q}_{\bar{\theta}}(\lambda')$ of corank-one, and moreover, λ' can be arbitrarily close to $\bar{\lambda}$. Thus, we may assume that $\ell' := (\bar{x}, \lambda') \in U$. Since ℓ' also belongs to $\mathfrak{L}(\bar{\theta})$, it in fact belongs to $\mathfrak{L}(\bar{\theta}) \cap U \subseteq \liminf_{\theta \rightarrow \bar{\theta}} \mathfrak{L}(\theta)$. Then ℓ' satisfies Assumptions C1H and WC, and the theorem follows from Proposition 6.4. \square

Although Proposition 6.4 and theorem 6.7 are easy to interpret (particularly Proposition 6.4), verifying the continuity assumptions on \mathfrak{L} might be complicated. In the following sections we will find simpler regularity conditions that ensure these continuity properties (see Figure 6-3). In particular, in Section 6.4 we will see that, for the restricted case of Theorem 6.1, a simple constraint qualification (ACQ) suffices. For the general case (Theorem 6.14) we will need two additional regularity assumptions.

6.4 An easy first case

The purpose of this section is to prove a generalized version of Theorem 6.1, and to obtain bounds on the magnitude of the perturbations we can tolerate. Before presenting the theorem, we recall a well-studied constraint qualification (also known as quasiregularity [19]), that guarantees the existence of Lagrange multipliers (see e.g., [15, §5.1]).

Notation. *This section works with problems in homogeneous and non-homogeneous (affine) form. We will distinguish them by using different notation. In particular, y denotes variables in affine coordinates, and x in homogeneous coordinates.*

Definition 6.2. Given $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, let $Y := \{y \in \mathbb{R}^n : f(y) = 0\}$. The *Abadie constraint qualification* (ACQ) holds at $\bar{y} \in Y$, denoted $\text{ACQ}_Y(\bar{y})$, if Y is a smooth manifold nearby \bar{y} , and $\text{rk}(\nabla f(\bar{y})) = n - \dim_{\bar{y}} Y$. Here $\dim_{\bar{y}} Y$ denotes the local dimension of Y at \bar{y} .

Remark 6.5. Under ACQ, the Lagrange multiplier space has dimension $m - (n - \dim_{\bar{y}} Y)$.

We will prove the following theorem.

Theorem 6.8. *Consider the problem*

$$\min_{y \in Y} q_\theta(y), \quad \text{with} \quad Y := \{y \in \mathbb{R}^n : f_1(y) = \cdots = f_m(y) = 0\}, \quad (6.6)$$

where q_θ, f_i are quadratic, and the dependence on θ is continuous. Let $\bar{\theta}$ be such that $q_{\bar{\theta}}$ is strictly convex, and its minimizer \bar{y} satisfies $\nabla q_{\bar{\theta}}(\bar{y}) = 0$, or equivalently, \bar{y} is the unconstrained minimizer of $q_{\bar{\theta}}(y)$. If $\text{ACQ}_Y(\bar{y})$ holds, then there is zero-duality-gap whenever θ is close enough to $\bar{\theta}$. Moreover, (P_θ^*) recovers the minimizer of (6.6).

Remark 6.6. The nearest point problem to a quadratic variety (Theorem 6.1) corresponds to the case $q_\theta(y) := \|y - \theta\|^2$. Indeed, this objective is strictly convex, the minimizer is $\bar{y} = \bar{\theta}$ (since $\bar{\theta} \in Y$), and thus $\nabla q_{\bar{\theta}}(\bar{y}) = 0$. Theorem 6.1 generalizes the main result of [1], as will be discussed in Example 6.11.

6.4.1 Preparing the problem

The proof of Theorem 6.8 will rely on the tools we developed in Section 6.3; more precisely, on Proposition 6.4. We will now prepare problem (6.6) for applying these tools.

Since the equations q_θ, f_i in (6.6) may involve linear terms, we need to consider its homogenized form:

$$\min_{x \in X} g_\theta(x), \quad X := \{(z_0, y) \in \mathbb{R}^{n+1} : z_0^2 = 1, h_1(z_0, y) = \cdots = h_m(z_0, y) = 0\} \quad (6.7)$$

where g_θ, h_i are the homogenizations of q_θ, f_i . The Lagrange multiplier spaces of (6.6) and (6.7) are isomorphic. Concretely, for any feasible $x_\theta = (1, y_\theta)$ it can be shown that:

$$\Lambda_\theta(y_\theta) = \{\mu \in \mathbb{R}^m : \mu^T \nabla f(y_\theta) = -\nabla q_\theta(y_\theta)\}, \quad (6.8)$$

$$\Lambda_\theta(x_\theta) = \{\lambda = (\lambda_0, \mu) \in \mathbb{R}^{m+1} : \lambda_0 = -g_\theta(x_\theta), \mu \in \Lambda_\theta(y_\theta)\}. \quad (6.9)$$

Also consider the Hessians of the Lagrangian functions of (6.6) and (6.7):

$$\mathcal{C}_\theta(\mu) := \nabla^2 q_\theta + \sum_i \mu_i \nabla^2 f_i, \quad \mathcal{Q}_\theta(\lambda_0, \mu) := \nabla^2 g_\theta + \lambda_0 \nabla^2 (z_0^2 - 1) + \sum_i \mu_i \nabla^2 h_i. \quad (6.10)$$

Observe that $\mathcal{Q}_\theta(\lambda_0, \mu) \in \mathcal{S}^{n+1}$ contains $\mathcal{C}_\theta(\mu) \in \mathcal{S}^n$ as a submatrix.

We can now specialize the conditions from Proposition 6.4 to problem (6.6).

Lemma 6.9. *Let $\bar{x} = (1, \bar{y})$ and $\bar{\lambda} = (-g_{\bar{\theta}}(\bar{x}), 0)$. Then $(\bar{x}, \bar{\lambda})$ is primal/dual optimal, $\bar{\lambda}$ satisfies Assumption C1H, and*

$$\text{Assumption WC holds} \iff \exists y_\theta \in Y, \mu_\theta \in \Lambda_\theta(y_\theta) \text{ s.t. } (y_\theta, \mu_\theta) \xrightarrow{\theta \rightarrow \bar{\theta}} (\bar{y}, 0). \quad (6.11)$$

Proof. The equivalence in (6.11) follows from the isomorphism $\Lambda_\theta(y_\theta) \cong \Lambda_\theta(x_\theta)$. Recall that $(\bar{x}, \bar{\lambda})$ is optimal if and only if conditions 6.2(i-iii) are satisfied. Since $\nabla q_{\bar{\theta}}(\bar{y}) = 0$ then $0 \in \Lambda_{\bar{\theta}}(\bar{y})$, and thus $\bar{\lambda} \in \Lambda_{\bar{\theta}}(\bar{x})$. It remains to show that $\mathcal{Q}_{\bar{\theta}}(\bar{\lambda}) \succeq 0$ and has corank-one. Observe that $\mathcal{C}_{\bar{\theta}}(0) = \nabla^2 q_{\bar{\theta}} \succ 0$ because $q_{\bar{\theta}}$ is strictly convex. Note that $\mathcal{Q}_{\bar{\theta}}(\bar{\lambda})$ contains $\mathcal{C}_{\bar{\theta}}(0)$ as a submatrix, and the extra row/column is such that $\mathcal{Q}_{\bar{\theta}}(\bar{\lambda})\bar{x} = 0$. It follows that $\mathcal{Q}_{\bar{\theta}}(\bar{\lambda}) \succeq 0$ and has corank-one. \square

6.4.2 Small multipliers

From Lemma 6.9, what we need now is to show the existence of some (y_θ, μ_θ) such that y_θ approaches \bar{y} and μ_θ approaches 0.

Lemma 6.10. *For each θ , let y_θ be an optimal solution of (6.6). Then $y_\theta \rightarrow \bar{y}$ as $\theta \rightarrow \bar{\theta}$.*

Proof. See Section 6.8. \square

It remains to find some small multipliers μ_θ associated to y_θ . The ACQ property allows us to do so.

Lemma 6.11. *Let y_θ be an optimal solution of (6.6). Let σ_θ be the s -th largest singular value of $\nabla f(y_\theta)$, where $s := \text{codim}_{y_\theta} Y$, and assume that $\sigma_\theta > 0$ (i.e., $ACQ_Y(y_\theta)$ holds). Then there exists $\mu_\theta \in \Lambda_\theta(y_\theta)$ with $\|\mu_\theta\| \leq \frac{1}{\sigma_\theta} \|\nabla q_\theta(y_\theta)\|$.*

Proof. Recall that $\Lambda_\theta(y_\theta)$ is given by the linear equation in (6.8). This equation has a solution, since ACQ guarantees the existence of multipliers. Then $\mu_\theta^T := -\nabla q_\theta(y_\theta)J^\dagger$ is one such solution, where $J := \nabla f(y_\theta)$ is the Jacobian and † denotes the pseudo-inverse. The lemma follows by noticing that $\|J^\dagger\| = 1/\sigma_\theta$, since σ_θ is the smallest nonzero singular value of J . \square

Proof of Theorem 6.8. By Proposition 6.4 and lemma 6.9, it is enough to find y_θ, μ_θ as in (6.11). Let y_θ be an optimal solution of (6.6). By Lemma 6.10 we know that $y_\theta \rightarrow \bar{y}$ as $\theta \rightarrow \bar{\theta}$. Since ACQ is an open condition, it holds in a neighborhood of \bar{y} . By Lemma 6.11 there exists μ_θ with $\|\mu_\theta\| \leq \frac{1}{\sigma_\theta} \|\nabla q_\theta(y_\theta)\|$. By assumption $\nabla q_{\bar{\theta}}(\bar{y}) = 0$, and by ACQ we also have $\sigma_{\bar{\theta}} > 0$. It follows that $\mu_\theta \rightarrow 0$ as $\theta \rightarrow \bar{\theta}$, as wanted. \square

6.4.3 Guaranteed region of zero-duality-gap

We proceed to estimate bounds on the magnitude of the perturbations we can tolerate. Consider the set of zero-duality-gap parameters:

$$\bar{\Theta} := \{\theta \in \Theta : \text{val}(P_\theta) = \text{val}(D_\theta), \text{ and } (P_\theta^*) \text{ recovers the minimizer}\}.$$

Our goal is to find a neighborhood of $\bar{\theta}$ that is entirely contained in $\bar{\Theta}$.

Example 6.4. Consider one more time the twisted cubic from Example 6.1. Figure 6-4 illustrates the region of zero-duality-gap guaranteed by the results of this section.

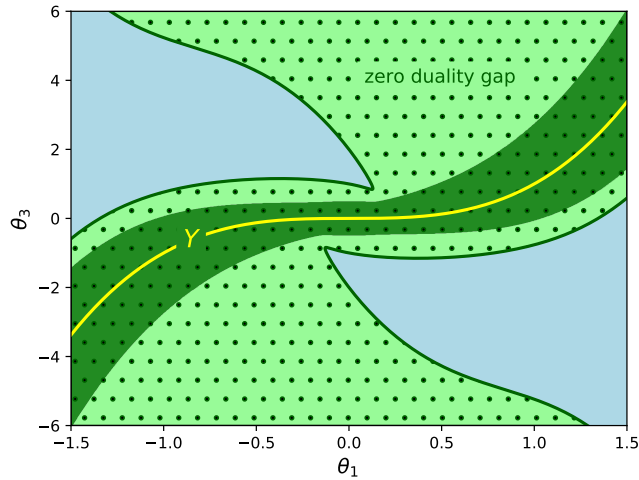


Figure 6-4: Region of zero-duality-gap from Figure 6-1. The darker region is the guaranteed region of zero-duality-gap (Corollary 6.13).

In order to find explicit bounds we need to quantify each of the assumptions we made in Proposition 6.4, and to impose some additional Lipschitz properties. We make the following assumptions:

6.12(i) (corank-one Hessian) The second smallest eigenvalue of $\mathcal{Q}_{\bar{\theta}}(\bar{\lambda})$, denoted $\nu_2(\bar{Q})$, is strictly positive.

6.12(ii) (weak continuity of \mathfrak{L}) There is a constant $K \geq 0$ such that

$$\forall \theta \in \Theta \exists (x_\theta, \lambda_\theta) \in \mathfrak{L}(\theta) \text{ s.t. } x_\theta \xrightarrow{\theta \rightarrow \bar{\theta}} \bar{x}, \quad \|\lambda_\theta - \bar{\lambda}\| \leq K\|\theta - \bar{\theta}\|.$$

6.12(iii) (dependence on θ) There is a constant $L \geq 0$ such that

$$\|\mathcal{Q}_\theta(\bar{\lambda}) - \mathcal{Q}_{\bar{\theta}}(\bar{\lambda})\|_F \leq L\|\theta - \bar{\theta}\|$$

We will need one last assumption, which holds, in particular, when Θ is bounded.

6.12(iv) Consider the linear map $\mathcal{H}_\theta : \mathbb{R}^m \rightarrow \mathcal{S}^N$ that $\mu \mapsto \frac{1}{2} \sum_i \mu_i H_\theta^i$. There is some $M \geq 0$ that bounds the operator norm $\|\mathcal{H}_\theta\| \leq M$ for all $\theta \in \Theta$.

The following theorem is the quantitative version of Proposition 6.4.

Theorem 6.12. *Under Assumptions 6.12(i-iv),*

$$\left\{ \theta \in \Theta : \|\theta - \bar{\theta}\| < \frac{\nu_2(\bar{Q})}{KM + L} \right\} \subseteq \bar{\Theta}.$$

Proof. Let θ be such that $\|\theta - \bar{\theta}\| < \frac{\nu_2(\bar{Q})}{KM + L}$. By Lemma 6.2, it is enough to show the existence of Lagrange multipliers λ_θ such that the second smallest eigenvalue of $\mathcal{Q}_\theta(\lambda_\theta)$ is positive (the first one is zero). By assumption there exists λ_θ such that $\|\lambda_\theta - \bar{\lambda}\| < K\|\theta - \bar{\theta}\|$. Note that

$$\|\mathcal{Q}_\theta(\lambda_\theta) - \mathcal{Q}_\theta(\bar{\lambda})\|_F = \|\mathcal{H}_\theta(\lambda_\theta - \bar{\lambda})\|_F \leq \|\mathcal{H}_\theta\| \|\lambda_\theta - \bar{\lambda}\| \leq KM\|\theta - \bar{\theta}\|,$$

and thus

$$\begin{aligned} \|\mathcal{Q}_\theta(\lambda_\theta) - \mathcal{Q}_{\bar{\theta}}(\bar{\lambda})\|_F &\leq \|\mathcal{Q}_\theta(\lambda_\theta) - \mathcal{Q}_\theta(\bar{\lambda})\|_F + \|\mathcal{Q}_\theta(\bar{\lambda}) - \mathcal{Q}_{\bar{\theta}}(\bar{\lambda})\|_F \\ &\leq (KM + L)\|\theta - \bar{\theta}\| < \nu_2(\bar{Q}). \end{aligned}$$

By Weyl's inequality [52, Thm 5.1], we have

$$|\nu_2(\mathcal{Q}_\theta(\lambda_\theta)) - \nu_2(\mathcal{Q}_{\bar{\theta}}(\bar{\lambda}))| \leq \|\mathcal{Q}_\theta(\lambda_\theta) - \mathcal{Q}_{\bar{\theta}}(\bar{\lambda})\| \leq \|\mathcal{Q}_\theta(\lambda_\theta) - \mathcal{Q}_{\bar{\theta}}(\bar{\lambda})\|_F$$

and thus

$$\nu_2(\mathcal{Q}_\theta(\lambda_\theta)) \geq \nu_2(\mathcal{Q}_{\bar{\theta}}(\bar{\lambda})) - \|\mathcal{Q}_\theta(\lambda_\theta) - \mathcal{Q}_{\bar{\theta}}(\bar{\lambda})\|_F > \nu_2(\bar{Q}) - \nu_2(\bar{Q}) = 0,$$

as wanted. \square

In the special case of the nearest point problem from Theorem 6.1 the above bounds can be made more explicit. The perturbation tolerance region shown in Figure 6-4 uses the bound in the following corollary.

Corollary 6.13. *Consider the (affine) setting of Theorem 6.1. For $\bar{y} \in Y$, let $\Theta(\bar{y})$ consist of all θ for which \bar{y} is the nearest point in Y , i.e., $\|\theta - \bar{y}\| = \text{dist}(\theta, Y)$. Then*

$$\left\{ \theta \in \Theta(\bar{y}) : \|\theta - \bar{\theta}\| < \frac{\sigma_s}{2M} \right\} \subseteq \bar{\Theta}$$

where:

- σ_s is the s -th largest singular value of $\nabla f(\bar{y})$, where $s := \text{codim}_{\bar{y}} Y$.
- M is the operator norm of the linear map $\mathcal{H} : \mathbb{R}^m \rightarrow \mathcal{S}^n$ that $\mu \mapsto \frac{1}{2} \sum_i \mu_i \nabla^2 f_i$.

Proof. Follows by noticing that $\nu_2(\bar{Q}) = 1$, $K = \frac{2}{\sigma_s}$, $L = 0$. See Section 6.8. \square

6.5 The general case

In Section 6.4 we observed that for nearest point problems to quadratic varieties a simple regularity condition (ACQ) guaranteed zero-duality-gap nearby $\bar{\theta}$. The purpose of this section is to identify regularity conditions that work for arbitrary QCQP's. Throughout this section we consider problem (P_θ) and use the following notation:

- the parameter space Θ is an open set of \mathbb{R}^d .
- $\bar{\theta} \in \Theta$ is a zero-duality-gap parameter, i.e., $\text{val}(P_{\bar{\theta}}) = \text{val}(D_{\bar{\theta}})$.

- $\bar{x} \in \mathbb{R}^N$ is optimal for $(P_{\bar{\theta}})$, and $\bar{\lambda} \in \mathbb{R}^m$ is optimal for $(D_{\bar{\theta}})$.
- $\bar{Q} := \mathcal{Q}_{\bar{\theta}}(\bar{\lambda}) \in \mathcal{S}^N$ is the Hessian of the Lagrangian at $\bar{\theta}$. Note that $\bar{Q} \succeq 0$.
- $X_{\theta} := \{x \in \mathbb{R}^N : h_{\theta}(x) = 0\}$ is the (primal) feasible set, and $\bar{X} := X_{\bar{\theta}}$.

We proceed to describe Theorem 6.14, our most general result. As illustrated in Figure 6-3, the theorem has four assumptions. The first of them is the restricted Slater assumption, which we restate for the convenience of the reader. The remaining three are regularity conditions that will be explained later in this section.

Theorem 6.14 (Main result). *Assume that:*

RS (restricted Slater) There exists $\mu \in \mathbb{R}^m$ such that $\mu^T \nabla h_{\bar{\theta}}(\bar{x}) = 0$ and $(\sum_i \mu_i H_{\bar{\theta}}^i)|_V \succ 0$, where $V := \{v \in \mathbb{R}^N : \bar{Q}v = 0, \bar{x}^T v = 0\}$.

R1 (constraint qualification) $\text{ACQ}_{\bar{X}}(\bar{x})$ holds.

R2 (smoothness) $\mathcal{W} := \{(\theta, x) : h_{\theta}(x) = 0\}$ is a smooth manifold nearby $\bar{w} := (\bar{\theta}, \bar{x})$, and $\dim_{\bar{w}} \mathcal{W} = \dim \Theta + \dim_{\bar{x}} \bar{X}$.

R3 (not a branch point) \bar{x} is not a branch point of \bar{X} with respect to $v \mapsto \bar{Q}v$.

Then $\text{val}(P_{\theta}) = \text{val}(D_{\theta})$ whenever θ is close enough to $\bar{\theta}$. Moreover, (P_{θ}) has a unique optimal solution x_{θ} , and (P_{θ}^) has as unique optimal solution $x_{\theta} x_{\theta}^T$.*

Remark 6.7 (Strictly convex Lagrangian). If \bar{Q} has corank-one then conditions RS and R3 always hold, so it suffices to check R1 and R2.

We proceed to discuss each of the assumptions of Theorem 6.14.

RS. Restricted Slater

In order to illustrate Assumption RS in an example, we will first express it in a slightly different form. Let $n := \text{rk } \bar{Q}$, $k := N - n$, and consider a coordinate system $x = (z, u)$, $z \in \mathbb{R}^k$, $u \in \mathbb{R}^n$ such that the Hessian has the form $\bar{Q} = \begin{pmatrix} 0 & 0 \\ 0 & \bar{C} \end{pmatrix}$ where $\bar{C} \in \mathcal{S}^n$ is positive definite. In these coordinates we have $V = (\bar{z})^{\perp} \times \{0^n\}$, where $\bar{x} = (\bar{z}, \bar{u})$. Then

$$(\sum_i \mu_i H_{\bar{\theta}}^i)|_V = \mathcal{A}(\mu)|_{(\bar{z})^{\perp}}, \quad \text{where} \quad \mathcal{A}(\mu) := \sum_i \mu_i \nabla_{zz}^2 h_{\bar{\theta}}^i \in \mathcal{S}^k, \quad (6.12)$$

and Assumption RS becomes: $\exists \mu \in \mathbb{R}^m$ s.t. $\mu^T \nabla h_{\bar{\theta}}(\bar{x}) = 0$, $\mathcal{A}(\mu)|_{(\bar{z})^\perp} \succ 0$.

Example 6.5. Consider a nearest point problem as in Proposition 6.3. After the change of coordinates $u := y - \theta z_0$ the problem becomes

$$\min_{z \in \mathbb{R}^k, u \in \mathbb{R}^n} \|u\|^2 \quad \text{s.t.} \quad h_i(z, u + \theta z_0) = 0$$

Since $\bar{\lambda} = 0$ then $\bar{Q} = \nabla^2(\|u\|^2) = \begin{pmatrix} 0 & 0 \\ 0 & \text{id}_n \end{pmatrix}$ has the desired form (in coordinates z, u).

Let us see that Assumption RS is satisfied in Example 6.2.

Example 6.6. Consider the nearest point problem to the curve $y_2^2 = y_1^3$. Homogenizing the equations in (6.3) with respect to z_0 we get

$$h_0 := z_0^2 - 1, \quad h_1 := y_2 z_0 - y_1 z_1, \quad h_2 := y_1 z_0 - z_1^2, \quad h_3 := y_2 z_1 - y_1^2.$$

Let $\theta \in Y$ be a parameter on the curve, which means $\theta = (t^2, t^3)$ for some $t \in \mathbb{R}$. The minimizer of the homogenized problem is $\bar{x} = (\bar{z}, \theta)$, where $\bar{z} = (1, t)$. Let us see that for any $\theta \neq 0$ the vector $\mu := (0, t, -t^2, -1)$ satisfies Assumption RS. Observe that

$$\nabla h(\bar{x}) = \begin{pmatrix} 2 & 0 & 0 & 0 \\ \theta_2 & -\theta_1 & -z_1 & 1 \\ \theta_1 & -2t & 1 & 0 \\ 0 & \theta_2 & -2\theta_1 & t \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ t^3 & -t^2 & -t & 1 \\ t^2 & -2t & 1 & 0 \\ 0 & t^3 & -2t^2 & t \end{pmatrix},$$

and thus $\mu^T \nabla h(\bar{x}) = 0$. It remains to check the positivity condition. In order to get the matrix $\mathcal{A}(\mu)$ we consider the change of coordinates $u = y - \theta z_0$, as explained in Example 6.5. Denoting $h'_i(z, u) := h_i(z, u + \theta z_0)$, then

$$\mathcal{A}(\mu) = \sum_i \mu_i \nabla_{zz}^2 h'_i = \begin{pmatrix} 2\mu_0 + 2\mu_1\theta_2 + 2\mu_2\theta_1 - 2\mu_3\theta_1^2 & -\mu_1\theta_1 + \mu_3\theta_2 \\ -\mu_1\theta_1 + \mu_3\theta_2 & -2\mu_2 \end{pmatrix} = \begin{pmatrix} t^4 & -t^3 \\ -t^3 & t^2 \end{pmatrix}.$$

Note that the orthogonal complement of $\bar{z} = (1, t)$ is spanned by $\zeta := (t, -1)$. Since $\zeta^T \mathcal{A}(\lambda) \zeta = t^2(t^2 + 1)^2$ is strictly positive, then $\mathcal{A}(\lambda)|_{(\bar{z})^\perp} \succ 0$. We conclude that Assumption RS holds for all $\theta \in Y \setminus \{0\}$.

Let us show now that Assumption RS is nontrivial. The following problem violates it, and indeed has positive-duality-gap for most values of θ .

Example 6.7 (Non-informative dual). Consider the following (homogenized) formulation for the nearest point problem to the twisted cubic:

$$\min_{z \in \mathbb{R}^3, y \in \mathbb{R}^3} \|y - \theta\|^2 \quad \text{s.t.} \quad z_0^2 - 1 = z_1^2 + z_2^2 - 1 = 0, \quad \begin{pmatrix} z_1 & z_2 \end{pmatrix} \begin{pmatrix} z_0 & y_1 & y_2 \\ y_1 & y_2 & y_3 \end{pmatrix} = 0$$

Let us see that $\text{val}(D_\theta) = 0$ for any θ , and thus there is positive-duality-gap for most values of θ . Observe that the diagonal entries of the Lagrangian Hessian are:

$$\text{diag}(\mathcal{Q}_\theta(\lambda)) = \frac{1}{2} \sum_i \lambda_i \text{diag}(\nabla_{xx}^2 h_i) = (\lambda_0, \lambda_1, \lambda_1, 0, 0, 0).$$

Then the cost function of any dual feasible λ satisfies $d(\lambda) := -\lambda_0 - \lambda_1 \leq 0$. Thus $\lambda = 0$ is dual optimal, as we claimed. Let us see now that the conditions from Theorem 6.14 are not met. Assuming ACQ holds, then $\dim \Lambda_{\bar{\theta}}(\bar{x}) = 5 - (6 - 1) = 0$ (see Remark 6.5) and thus $\Lambda_{\bar{\theta}}(\bar{x}) = \{0\}$. Since the μ from Assumption RS must belong $\Lambda_{\bar{\theta}}(\bar{x})$, then the only choice is $\mu = 0$. It follows that Assumption RS fails.

Remark 6.8. Examples 6.1 and 6.7 illustrate that different QCQP formulations of the same problem might have different stability properties. In general, an optimal QCQP formulation of a polynomial optimization problem can be derived by including all quadratic constraints that are valid on the variety. This is equivalent to working modulo the coordinate ring (see e.g., [42])

R1. Constraint qualification

The most basic regularity assumption we make is ACQ, i.e., that \bar{X} is smooth nearby \bar{x} , and that the tangent space of \bar{X} at \bar{x} is spanned by the gradients of the constraints. Note that ACQ is needed simply to guarantee the existence of Lagrange multipliers at \bar{x} .

R2. Smoothness

Another natural condition to make is that the dependence of the feasible set X_θ as a function of θ is smooth (continuously differentiable). More precisely, we require that $\mathcal{W} := \{(\theta, x) : h_\theta(x) = 0\}$ is a smooth manifold nearby $\bar{w} := (\bar{\theta}, \bar{x})$, and that its local dimension is precisely $\dim \Theta + \dim_{\bar{x}} \bar{X}$.

Remark 6.9. For nearest point problems the feasible set X_θ is independent of θ . Therefore, $\mathcal{W} = \Theta \times \bar{X}$ and condition R2 is satisfied.

R3. Not a branch point

The ACQ property guarantees regularity of \bar{X} nearby \bar{x} . When the Lagrangian function is not strictly convex, we also require that \bar{x} remains regular after projecting onto the range of \bar{Q} . The following example motivates this.

Example 6.8. Consider the nearest point problem to the curve Y defined by $y_2^2 = y_1^3$. In Example 6.2 we introduced an auxiliary variable z_1 to phrase the problem as a QCQP. The lifted curve in \mathbb{R}^3 is the twisted cubic, which is nonsingular everywhere; see Figure 6-2. But curve Y has a singularity at $(0, 0)$. This singularity is problematic, since it means that the nearest point map is not uniquely defined locally.

As in the above example, when the objective function is not strictly convex (e.g., due to auxiliary variables) we need that \bar{x} is regular after a suitable projection. By regularity we mean that \bar{x} is not a branch point, as formalized next.

Definition 6.3 (Branch point). Let $\pi : \mathbb{R}^N \rightarrow \mathbb{R}^n$ be a linear map. Let $\bar{X} \subseteq \mathbb{R}^N$ be the zero set of \bar{h} , and let $T_x \bar{X} := \ker \nabla \bar{h}(x)$ denote the tangent space of \bar{X} at x . We say that x is a *branch point* of \bar{X} with respect to π if there is a nonzero vector $v \in T_x \bar{X}$ with $\pi(v) = 0$.

Example 6.9. Let $\pi : (z, y_1, y_2) \mapsto (y_1, y_2)$, and consider the projection of the twisted cubic from Figure 6-2. Notice that the tangent line at the point $(0, 0, 0)$ is precisely the z -axis, and thus $(0, 0, 0)$ is a branch point.

The last regularity assumption is that \bar{x} is not a branch point with respect to the map $v \mapsto \bar{Q}v$, or equivalently, with respect to the projection $\pi_{\bar{Q}}$ onto the range of \bar{Q} .

Example 6.10. Consider a nearest point problem as in Proposition 6.3. Since $\bar{\lambda} = 0$ then $\bar{Q} = \nabla^2(\|y - \theta z_0\|^2)$, and its range is $\{(z_0, z', y) \in \mathbb{R}^{n+k} : z' = 0, z_0 = -y^T \bar{\theta}\}$ of dimension n . In particular, when $\bar{\theta} = 0$ then $\pi_{\bar{Q}}$ is simply $(z, y) \mapsto y$. Moreover, Assumption R3 holds if and only if $\nabla_{z'} f(\bar{z}', \bar{y})$ is injective.

6.6 Proof of main theorem

Recall the Lagrange multiplier mapping $\mathfrak{L} : \Theta \rightrightarrows \mathbb{R}^N \times \mathbb{R}^m$ from (6.5). In Theorem 6.7 we showed that local continuity of this mapping (Assumption SC), together with Assumption RS, guarantee zero-duality-gap nearby $\bar{\theta}$. Thus, our goal now is to see that Assumptions (R1-R3) imply local continuity of \mathfrak{L} . In order to do so, we consider the following property of set-valued mappings [54, 108].

Definition 6.4 (Aubin property). Let $\mathfrak{F} : \mathbb{R}^d \rightrightarrows \mathbb{R}^n$ be a set-valued mapping. \mathfrak{F} has the *Aubin property* at $\bar{p} \in \mathbb{R}^d$ for $\bar{y} \in \mathbb{R}^n$ if $\bar{y} \in \mathfrak{F}(\bar{p})$ and there is a constant $\kappa \geq 0$ and neighborhoods $U \ni \bar{y}, V \ni \bar{p}$ such that

$$\mathfrak{F}(p') \cap U \subseteq \mathfrak{F}(p) + \kappa|p' - p|\mathcal{B} \quad \text{for all } p', p \in V,$$

where $\mathcal{B} \subseteq \mathbb{R}^n$ denotes the unit ball.

The Aubin property implies local continuity, as stated next.

Lemma 6.15. *Let $\mathfrak{F} : \mathbb{R}^d \rightrightarrows \mathbb{R}^n$ be a mapping with closed graph. Assume that \mathfrak{F} has the Aubin property at \bar{p} for \bar{y} . Then there exists a closed neighborhood $U_0 \ni \bar{y}$ such that $p \mapsto \mathfrak{F}(p) \cap U_0$ is continuous at \bar{p} .*

Proof. See Section 6.8. □

Because of the above lemma, it is enough for us to show that the Lagrange multiplier mapping \mathfrak{L} has the Aubin property. In particular, the proof Theorem 6.14 can be reduced to the following proposition.

Proposition 6.16. *Let \bar{x} be optimal to $(P_{\bar{\theta}})$ and $\bar{\lambda} \in \Lambda_{\bar{\theta}}(\bar{x})$ be dual feasible. Under Assumptions (R1-R3) the mapping \mathfrak{L} has the Aubin property at $\bar{\theta}$ for $(\bar{x}, \bar{\lambda})$.*

Proof of Theorem 6.14 (assuming Proposition 6.16). Since \mathfrak{L} has the Aubin property at $\bar{\theta}$ for $(\bar{x}, \bar{\lambda})$, then by Lemma 6.15 there is a neighborhood $U \ni (\bar{x}, \bar{\lambda})$ such that the mapping $\theta \mapsto \mathfrak{L}(\theta) \cap U$ is continuous at $\bar{\theta}$. Then Assumption SC holds, and the result follows from Theorem 6.7. □

In the remaining of this section we will prove Proposition 6.16, thus concluding the proof of Theorem 6.14.

6.6.1 The implicit function theorem

The main technical tool we use for Proposition 6.16 is the implicit function theorem, which can be phrased in terms of the Aubin property (see [54, Ex. 4D.3]).

Theorem 6.17 (Implicit function). *Given $f : \mathbb{R}^d \times \mathbb{R}^n \rightarrow \mathbb{R}^m$ continuously differentiable, consider the solution mapping*

$$\mathfrak{F} : \mathbb{R}^d \rightrightarrows \mathbb{R}^n, \quad p \mapsto \{y \in \mathbb{R}^n : f(p, y) = 0\}.$$

Let \bar{p}, \bar{y} be such that $\bar{y} \in \mathfrak{F}(\bar{p})$. If $\nabla_y f(\bar{p}, \bar{y})$ is surjective, then \mathfrak{F} satisfies the Aubin property at \bar{p} for \bar{y} .

Note that Proposition 6.16 would be immediate if \mathfrak{L} satisfied the hypothesis from Theorem 6.17. Unfortunately this is not true, since the defining equations of \mathfrak{L} may have linearly dependent gradients. In order to fix this problem, we consider a maximal subset of the equations $h' \subseteq h$ such that $\{\nabla_x h_{\bar{\theta}}^i(\bar{x})\}_{h^i \in h'}$ are linearly independent. Equivalently, $h' \subseteq h$ is such that $\nabla_x h'_{\bar{\theta}}(\bar{x})$ is full rank, and has the same rank as $\nabla_x h_{\bar{\theta}}(\bar{x})$. Consider the modified solution mapping

$$\mathfrak{L}' : \theta \mapsto \{(x, \lambda) : h'_{\theta}(x) = 0, \mathcal{Q}_{\theta}(\lambda)x = 0\}.$$

This new mapping satisfies the assumptions of Theorem 6.17, as seen next.

Lemma 6.18. *Under Assumption R3, \mathfrak{L}' has the Aubin property at $\bar{\theta}$ for $(\bar{x}, \bar{\lambda})$.*

Proof. Let us see that the surjectivity condition from Theorem 6.17 is satisfied. To simplify the notation we will ignore the dependence on θ , since the only parameter we consider in this proof is $\bar{\theta}$. Let $J' := \nabla_x h'(\bar{x})$, which is a submatrix of $J := \nabla_x h(\bar{x})$. Note that by construction the rows of J' are independent and $\ker J' = \ker J$. Let $f(x, \lambda) := (h'(x), \mathcal{Q}_{\bar{\theta}}(\lambda)x) \in \mathbb{R}^{m+N}$.

We need to show that the rows of $\nabla f(\bar{x}, \bar{\lambda})$ are independent. Denoting $\bar{Q} := \mathcal{Q}_{\bar{\theta}}(\bar{\lambda})$, then

$$\nabla_{\lambda, x} f(\bar{x}, \bar{\lambda}) = \begin{pmatrix} 0 & \nabla h'(\bar{x}) \\ \nabla h(\bar{x})^T & \mathcal{Q}_{\bar{\theta}}(\bar{\lambda}) \end{pmatrix} = \begin{pmatrix} 0 & J' \\ J^T & \bar{Q} \end{pmatrix}.$$

Let (u, v) be a vector in the left kernel of $\nabla f(\bar{x}, \bar{\lambda})$, i.e., $v^T J^T = 0$, $u^T J' + v^T \bar{Q} = 0$. We need to show that $(u, v) = 0$. Since $v \in \ker J = \ker J'$ then $0 = (u^T J' + v^T \bar{Q})v = v^T \bar{Q}v$, and thus $\bar{Q}v = 0$. As $v \in \ker J$ and $\bar{Q}v = 0$, then $v = 0$ by Assumption R3. Therefore $0 = u^T J' + v^T \bar{Q} = u^T J'$, and thus $u = 0$ since the rows of J' are independent. \square

In order to prove Proposition 6.16 it remains to see that the modified mapping \mathfrak{L}' agrees with \mathfrak{L} , at least locally. This follows from the following lemma.

Lemma 6.19. *Let $X_\theta \subseteq X'_\theta \subseteq \mathbb{R}^N$ be the zero sets of h_θ, h'_θ . Under Assumptions R1 and R2, there are neighborhoods $V_0 \ni \bar{\theta}$ and $U_0 \ni \bar{x}$ such that $X_\theta \cap U_0 = X'_\theta \cap U_0$ for all $\theta \in V_0$.*

The proof of Lemma 6.19 requires an auxiliary lemma.

Lemma 6.20. *Let $\mathcal{W} := \{w \in \mathbb{R}^K : h(w) = 0\}$, where $h = (h_1, \dots, h_m)$, and assume that \mathcal{W} is a smooth D -dimensional manifold nearby \bar{w} . Let $h' = (h_1, \dots, h_{K-D}) \subseteq h$ be such that their gradients at \bar{w} are linearly independent. Then there is a neighborhood $U \subseteq \mathbb{R}^K$ of \bar{w} such that $\mathcal{W} \cap U = \mathcal{W}' \cap U$, where $\mathcal{W}' := \{w : h'(w) = 0\}$.*

Proof. By the implicit function theorem \mathcal{W}' is a D -dimensional manifold nearby \bar{w} . Thus, there is a neighborhood $U \subseteq \mathbb{R}^K$ of \bar{w} such that $\mathcal{W} \cap U$ is a submanifold of $\mathcal{W}' \cap U$. Since they have the same dimension, $\mathcal{W} \cap U$ must be an open set of $\mathcal{W}' \cap U$. \square

Proof of Lemma 6.20. Let $\mathcal{W} := \{(\theta, x) : h_\theta(x) = 0\}$ and $\mathcal{W}' := \{(\theta, x) : h'_\theta(x) = 0\}$. We will use Lemma 6.20 to show the existence of a neighborhood $U \ni \bar{w}$, such that $\mathcal{W} \cap U = \mathcal{W}' \cap U$. Note that this would conclude the proof. By Assumption R2 we know that \mathcal{W} is a smooth manifold nearby $\bar{w} := (\bar{x}, \bar{\theta})$ of dimension $D := \dim \Theta + \dim_{\bar{x}} \bar{X}$. Recall that by construction of h' the gradients $\{\nabla h_{\bar{\theta}}^i(\bar{x})\}_{h^i \in h'}$ are linearly independent, and the number of equations is $|h'| = \text{rk } \nabla h_{\bar{\theta}}(\bar{x})$. Since $\text{ACQ}_{\bar{X}}(\bar{x})$ holds, then

$$|h'| = \text{rk } \nabla h_{\bar{\theta}}(\bar{x}) = N - \dim_{\bar{x}} \bar{X} = (\dim \Theta + N) - D.$$

Thus the assumptions of Lemma 6.20 are satisfied, as wanted. \square

We are finally ready to prove Proposition 6.16.

Proof of Proposition 6.16. The Aubin property is a local condition. Since $\mathfrak{L}, \mathfrak{L}'$ agree nearby $\bar{\theta}, \bar{x}$ (Lemma 6.19), and since \mathfrak{L}' has the Aubin property (Lemma 6.18), then the same holds for \mathfrak{L} . \square

6.7 Applications

6.7.1 Estimation problems with strictly convex objective

Let us show some immediate consequences of Theorems 6.1 and 6.8. We will use the following well-known property (see e.g., [57, §16.6]).

Lemma 6.21. *Let $h \subseteq \mathbb{R}[x]$ be a polynomial system and let X be its variety. If the ideal $\langle h \rangle$ is radical then ACQ holds for each smooth point of X . In particular, ACQ holds generically on X (on a dense open subset).*

We first consider two nearest point problems. We will apply Theorem 6.1, and thus we will need to check the ACQ property.

Example 6.11 (Triangulation). Given ℓ projective cameras $P_j : \mathbb{P}^3 \rightarrow \mathbb{P}^2$ and noisy images $\hat{u}_j \in \mathbb{R}^2$ of an unknown point $z \in \mathbb{P}^3$, the triangulation problem is

$$\min_{u \in U} \sum_{j=1}^{\ell} \|u_j - \hat{u}_j\|^2, \quad U := \{u \in (\mathbb{R}^2)^\ell : \exists z \in \mathbb{P}^3 \text{ s.t. } u_j = \Pi P_j z \text{ for } 1 \leq j \leq \ell\},$$

where $\Pi : \mathbb{P}^2 \rightarrow \mathbb{R}^2$ is the dehomogenization $(y_1 : y_2 : y_3) \mapsto (y_1/y_3, y_2/y_3)$. Notice that this problem is parameterized by $\theta = (\hat{u}_1, \dots, \hat{u}_\ell)$. U is known as the *multiview variety*. Assume that either $\ell = 2$, or $\ell \geq 4$ and the camera centers are not coplanar. Then the variety U can be described as

$$U = \{u \in (\mathbb{R}^2)^\ell : f_{ij}(u_i, u_j) = 0, \ 1 \leq i < j \leq \ell\},$$

where f_{ij} are some quadratic equations known as the epipolar constraints [71]. By using this description of U we obtain a QCQP. Moreover, these epipolar equations define a radical

ideal [71], and thus ACQ holds generically (Lemma 6.21). It follows from Theorem 6.1 that the SDP relaxation of this QCQP is (generically) tight under small noise.

Remark 6.10. The above SDP relaxation was considered in [1], where they also showed tightness under low noise.

Example 6.12 (Rank one approximation). Consider the problem of finding the nearest rank one tensor. Let $\mathbb{R}^{n_1 \times \dots \times n_\ell}$ be the set of tensors of dimensions (n_1, \dots, n_ℓ) and let $X \subseteq \mathbb{R}^{n_1 \times \dots \times n_\ell}$ be the space of rank one tensors. X is known as the *Segre variety*, and it is also defined by quadratic equations (the 2×2 minors of the tensor flattenings). Therefore, this nearest point problem is a QCQP, and we can consider its SDP relaxation. The Segre variety is smooth at any point other than the origin. Since the ideal is radical, then ACQ holds at all these points. Thus, under low noise assumptions this problem is solved exactly by the SDP relaxation. This result also extends to the case of symmetric tensors, given that the *Veronese variety* is also defined by quadrics.

Remark 6.11. An essentially equivalent SDP relaxation for the nearest rank one tensor was proposed in [102]. No tightness results were known.

We will now see some applications of Theorem 6.8. Note that the theorem has three assumptions: the objective is strictly convex, ACQ holds, and the minimizer of the noiseless case is also the global minimum. Since in the problems below the objective function is a squared loss function, and since in the noiseless case the objective value is zero, then the last condition is always satisfied. Thus, we will only check strict convexity and ACQ.

Example 6.13 ($SO(d)$ synchronization). Consider the problem of determining the absolute rotations of $n+1$ objects given (noisy) relative rotations among some pairs. Let $SO(d)$ denote the *special orthogonal group*. Given a graph $G = (V, E)$, where $V = \{0, \dots, n\}$, and matrices $\hat{R}_{ij} \in \mathbb{R}^{d \times d}$ for each $ij \in E$, the problem is

$$\min_{R_1, \dots, R_n \in SO(d)} \sum_{ij \in E} \|R_j - \hat{R}_{ij} R_i\|_F^2, \quad SO(d) := \{R \in \mathbb{R}^{d \times d} : R^T R = \text{id}_d, \text{Det}(R) = 1\}, \quad (6.13)$$

where $R_0 := \text{id}_n$ is the reference point, This problem is parametrized by $\theta = (\hat{R}_{ij})_{ij \in E}$. To

obtain a QCQP, we can replace $SO(d)$ by the *orthogonal group* $O(d)$:

$$\min_{R_1, \dots, R_n \in O(d)} \sum_{ij \in E} \|R_j - \hat{R}_{ij} R_i\|_F^2, \quad O(d) := \{R \in \mathbb{R}^{d \times d} : R^T R = \text{id}_d\}. \quad (6.14)$$

Observe that (6.13) and (6.14) have the same minimizer in the low noise regime, given that $SO(d)^n$ is a connected component of $O(d)^n$. Consider the SDP relaxation of the QCQP (6.14). Note that the objective function is strictly convex (Lemma 6.22), and that ACQ is satisfied everywhere since the variety is smooth and the ideal is radical (Lemma 6.21). Thus, under low noise, the SDP relaxation finds the true minimizer of (6.14), which is the same of (6.13).

Lemma 6.22. *Let $G = (V, E)$ be a connected graph, let $x_0 \in \mathbb{R}^k$, and let $L_{ij} : \mathbb{R}^k \rightarrow \mathbb{R}^k$ be invertible linear maps for $ij \in E$. Then the function $f(x) := \sum_{ij \in E} \|x_j - L_{ij} x_i\|^2$, where $x = (x_1, \dots, x_n) \in (\mathbb{R}^k)^n$, is strictly convex.*

Proof. We may assume that the reference point $x_0 = 0$ (otherwise, simply apply an affine transformation). Since $f(x)$ is convex and homogeneous, it suffices to see that $f(x) = 0$ implies $x = 0$. If $f(x) = 0$ then $x_j = L_{ij} x_i$ for each $ij \in E$. Since $x_0 = 0$ and G is connected it is clear that each x_i must be zero. \square

Remark 6.12. An alternative QCQP formulation for the $SO(3)$ synchronization problem can be obtained by representing rotations with quaternions [63]. The same analysis as above shows that the corresponding SDP relaxation is tight in the low noise regime, as was observed experimentally in [63].

Example 6.14 ($SE(d)$ synchronization). A natural extension of the above problem is to replace rotation matrices by elements of the *special Euclidean group* $SE(d)$. Given a graph $G = (V, E)$, and $\hat{R}_{ij} \in \mathbb{R}^{d \times d}$, $\hat{u}_{ij} \in \mathbb{R}^d$ for $ij \in E$, the problem is

$$\min_{R_i \in SO(d), u_i \in \mathbb{R}^d} \sum_{ij \in E} \|R_j - \hat{R}_{ij} R_i\|_F^2 + \|u_j - u_i - R_i \hat{u}_{ij}\|^2, \quad (6.15)$$

where $R_0 := \text{id}_d$, $u_0 := 0$. As before, we can replace $SO(d)$ with $O(d)$ to obtain a QCQP, and consider its SDP relaxation. An argument similar to Lemma 6.22 shows that the objective function is strictly convex, and thus the SDP recovers the minimizer of (6.15) under low noise.

Remark 6.13. SDP relaxations for $SE(d)$ synchronization have received considerable attention in past years and similar tightness results have been derived [112, 146].

Example 6.15 (Orthogonal Procrustes). Given $n, k, m_1, m_2 \in \mathbb{N}$ and matrices $A \in \mathbb{R}^{m_1 \times n}$, $B \in \mathbb{R}^{m_1 \times m_2}$, $C \in \mathbb{R}^{k \times m_2}$, the weighted orthogonal Procrustes problem, also known as Penrose regression problem, is

$$\min_{X \in \mathbb{R}^{n \times k}} \|AXC - B\|_F^2, \quad \text{s.t.} \quad X^T X = \text{id}_k. \quad (6.16)$$

Note that the above is a QCQP parametrized by $\theta = (A, B, C)$. ACQ holds everywhere since the variety (the Stiefel manifold) is smooth and the ideal is radical. The objective function is strictly convex as long as the linear map $X \mapsto AXC$ is injective. In such cases the SDP relaxation (D_θ) will be tight under low noise.

Remark 6.14. Recall that in Section 5.7.2 we also studied this problem, and we introduced a “better” SDP relaxation (stronger and more efficient) than (D_θ) . Naturally, such relaxation will also be tight under low noise.

6.7.2 Stability of unconstrained SOS

Let $\mathbb{R}[z]_{2d}$ be the vector space of multivariate polynomials of degree at most $2d$ in variables $z = (z_1, \dots, z_n)$. Consider the parametric family of polynomial optimization problems:

$$\min_{z \in \mathbb{R}^n} p_\theta(z), \quad \text{where } p_\theta \in \mathbb{R}[z]_{2d} \text{ depends continuously on } \theta. \quad (POP_\theta)$$

The SOS relaxation of (POP_θ) is

$$\max_{\gamma \in \mathbb{R}} \gamma \quad \text{s.t.} \quad p_\theta(z) - \gamma \in \Sigma_{n,2d}, \quad (SOS_\theta)$$

where $\Sigma_{n,2d}$ is the SOS cone:

$$\Sigma_{n,2d} := \{f \in \mathbb{R}[z]_{2d} : f(z) = \sum_i f_i(z)^2 \text{ for some } f_i \in \mathbb{R}[z]_d\}.$$

Assume now that for a fixed value of $\bar{\theta}$ we know that the relaxation is tight. As before, we investigate the behavior of the SOS relaxation as $\theta \rightarrow \bar{\theta}$.

Example 6.16. For the polynomial $p_\theta(z) := z_1^4 z_2^2 + z_1^2 z_2^4 + \theta z_1^2 z_2^2 \in \mathbb{R}[z]_6$ we have:

$$\begin{aligned} \theta \geq 0 &\implies \text{val}(POP_\theta) = \text{val}(SOS_\theta) = 0, \\ \theta < 0 &\implies \text{val}(POP_\theta) = \frac{1}{27}\theta^3 \text{ and } (SOS_\theta) \text{ is infeasible.} \end{aligned}$$

Hence the relaxation is not stable nearby $\bar{\theta} = 0$.

The following theorem shows stability under a certain interiority condition.

Theorem 6.23. *Let $\bar{\theta}$ be such that $\bar{\gamma} := \text{val}(POP_{\bar{\theta}}) = \text{val}(SOS_{\bar{\theta}})$ and there is a unique minimizer \bar{z} . Consider the face $K_{\bar{z}}$ of the cone $\Sigma_{n,2d}$ given by the vanishing at \bar{z} :*

$$K_{\bar{z}} := \Sigma_{n,2d} \cap L_{\bar{z}}, \quad L_{\bar{z}} := \{f \in \mathbb{R}[z]_{2d} : f(\bar{z}) = 0, \nabla f(\bar{z}) = 0\}.$$

Note that $p_{\bar{\theta}} - \bar{\gamma} \in K_{\bar{z}}$. If $p_{\bar{\theta}} - \bar{\gamma}$ lies in the relative interior of $K_{\bar{z}}$, then the relaxation (SOS_θ) is tight and recovers the minimizer whenever θ is close enough to $\bar{\theta}$.

We proceed to prove Theorem 6.23. We may assume WLOG that $\bar{\gamma} = 0$, and thus $p_{\bar{\theta}} \in \Sigma_{n,2d}$. In order to use our methods, we need to rephrase (POP_θ) as a QCQP. Let

$$x := (z^\alpha)_{\alpha \in J} \in (\mathbb{R}[z]_d)^N, \quad \text{where } J := \{\alpha \in \mathbb{N}^n : \sum_i \alpha_i \leq d\}, \quad N := \binom{n+d}{d},$$

be the vector with all N monomials of degree at most d . Notice that any $f \in \mathbb{R}[z]_{2d}$ can be written in the form $f(z) = x^T Q x$ for some $Q \in \mathcal{S}^N$. We say that such a Q is a *Gram matrix* of f . Moreover, f is SOS if and only if it has a positive semidefinite Gram matrix. We need two properties of these Gram matrices.

Lemma 6.24. *Assume that $\bar{p} \in \text{int } K_{\bar{z}}$. Then \bar{p} has a Gram matrix $\bar{Q} \succeq 0$ of corank-one.*

Proof. Consider the linear map

$$\phi : \mathcal{S}^N \rightarrow \mathbb{R}[z]_{2d}, \quad A \mapsto x^T A x. \tag{6.17}$$

Let $\bar{x} \in \mathbb{R}^N$ be given by evaluating each of the monomials in x at \bar{z} . Let

$$S := \{Q \in \mathcal{S}^N : Q \succeq 0, Q\bar{x} = 0\}$$

and observe that $K_{\bar{z}} = \phi(S)$. Since linear maps preserve relative interiors of convex sets, then $\text{int } K_{\bar{z}} = \phi(\text{int } S)$. It follows that \bar{p} has a Gram matrix $\bar{Q} \succeq 0$ such that $\bar{Q}\bar{x} = 0$, and $\text{corank } \bar{Q} = 1$. \square

Lemma 6.25. *Let \bar{Q} be a Gram matrix of $p_{\bar{\theta}}$. Then $Q_{\theta} := \phi^{\dagger}(p_{\theta} - p_{\bar{\theta}}) + \bar{Q}$ is a Gram matrix of p_{θ} , where ϕ^{\dagger} is the pseudo-inverse of the linear map in (6.17).*

Proof. Follows by noticing that Q is a Gram matrix of f if and only if $\phi(Q) = f$. \square

By the above lemmas, we know that there exist Gram matrices $Q_{\theta} \in \mathcal{S}^N$ for each p_{θ} such that: $Q_{\bar{\theta}} \succeq 0$ and has corank-one, and the dependence on θ is continuous. Thus the parametric optimization problem (POP_{θ}) can be phrased as

$$\min_{x \in X} x^T Q_{\theta} x, \quad \text{where} \quad X := \{(z^{\alpha})_{\alpha \in J} : z \in \mathbb{R}^n\} \subseteq \mathbb{R}^N.$$

The above is indeed a QCQP since X , the Veronese variety, is defined by quadratic equations

$$X := \{x \in \mathbb{R}^N : x_0 = 1, x_{\alpha_1} x_{\alpha_2} = x_{\beta_1} x_{\beta_2} \ \forall \alpha_1, \alpha_2, \beta_1, \beta_2 \in J \text{ s.t. } \alpha_1 + \alpha_2 = \beta_1 + \beta_2\}.$$

Since we have a QCQP formulation, its Lagrangian dual gives an SDP relaxation of (POP_{θ}) . Moreover, this Lagrangian dual coincides with (SOS_{θ}) . The proof of Theorem 6.23 now follows from Theorem 6.8.

Proof of Theorem 6.23. Consider the above QCQP. Since the first coordinate of x is always one, let $x = (1, y)$ with $y \in \mathbb{R}^{N-1}$. Similarly, let $\bar{x} = (1, \bar{y})$ where $\bar{y} = (\bar{z}^{\alpha})_{\alpha \in J \setminus \{0\}}$. Let $q_{\theta}(y) := x^T Q_{\theta} x$. By construction we know that $Q_{\bar{\theta}} \succeq 0$, $Q_{\bar{\theta}}\bar{x} = 0$, $\text{corank } Q_{\bar{\theta}} = 1$. It follows that $\min_y q_{\bar{\theta}}(y) = 0$, and it is attained at \bar{y} . The corank-one assumption means that $q_{\bar{\theta}}$ is strictly convex. In order to apply Theorem 6.8 it remains to see that the ACQ assumption is satisfied. Note that the variety X is smooth since the unique singularity of the Veronese

variety is the origin, but we are fixing the first coordinates to be one. Since the ideal is radical, Lemma 6.21 implies that ACQ holds everywhere. \square

6.7.3 Noisy Euclidean distance matrix completion

We now show a simple application of Theorem 6.14. Consider the problem of determining the location of $n + 1$ objects given the (noisy) pairwise distances among some of them. Formally, given $p \in \mathbb{N}$, a graph $G = (V, E)$, and positive numbers θ_{ij} for each $ij \in E$, the goal is to find $t_i \in \mathbb{R}^p$ such that

$$\|t_i - t_j\|^2 \approx \theta_{ij} \quad \text{for } ij \in E.$$

The problem can be modelled as finding the nearest point to the variety of *Euclidean distance matrices*:

$$\min_{d \in D} \sum_{ij \in E} (d_{ij} - \theta_{ij})^2, \quad D := \{d \in \mathbb{R}^{\binom{V}{2}} : \exists t_i \in \mathbb{R}^p \text{ s.t. } d_{ij} = \|t_i - t_j\|^2 \text{ for } ij \in \binom{V}{2}\}. \quad (6.18)$$

We point out that given a valid Euclidean distance matrix $d \in D$ recovering the locations t_i amounts to an eigenvalue decomposition.

Remark 6.15. It follows from [116] that the above problem is NP-hard even if $p = 1$.

Remark 6.16. The special case in which all pairs are observed, i.e., $E = \binom{V}{2}$, is nontrivial, and it is known as *multidimensional scaling* (see e.g., [33]).

We focus here on the one-dimensional case ($p = 1$). The variety of 1D Euclidean distance matrices is defined by some quadratics known as the *Cayley-Menger determinants*:

$$D = \{d \in \mathbb{R}^{\binom{V}{2}} : h_{ijk}(d) = 0 \text{ for } ijk \in \binom{V}{3}\}, \quad h_{ijk}(d) := \text{Det} \begin{pmatrix} 0 & d_{ij} & d_{ik} & 1 \\ d_{ij} & 0 & d_{jk} & 1 \\ d_{ik} & d_{jk} & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}.$$

In particular, problem (6.18) is a QCQP (when $p = 1$). Notice that if all pairs are observed, i.e., $E = \binom{V}{2}$, then Theorem 6.1 tells us that its SDP relaxation is tight under low noise. In case that there are missing pairs this argument does not apply. The next example illustrates that Theorem 6.14 might be used to show tightness in such cases.

Example 6.17. Consider the graph

$$G = (V, E), \quad V = \{0, 1, 2, 3\}, \quad E = \{02, 03, 12, 13, 23\},$$

whose only missing edge is 01. Let $\bar{\theta} \in \mathbb{R}^E$ be a zero duality gap parameter, i.e., there is some $t \in \mathbb{R}^n$ such that $\bar{\theta}_{ij} = (t_i - t_j)^2$ for all $ij \in E$. We assume that $\bar{\theta}$ is generic (in particular, $t_i \neq t_j$). We will use Theorem 6.14 to show zero-duality-gap nearby $\bar{\theta}$. We denote $t_{ij} := t_j - t_i$ to simplify the notation. Let us split the vector $d = (z_{01}, y)$, where $z_{01} \in \mathbb{R}$ and $y \in \mathbb{R}^E$. The minimizer of (6.18) is $\bar{d} = (\bar{z}_{01}, \bar{\theta})$, where $\bar{z}_{01} := t_{01}^2$, and its Jacobian is

$$\nabla h(\bar{d}) = \nabla_{z,y} h(\bar{d}) = 4 \begin{pmatrix} -t_{12}t_{02} & t_{12}t_{01} & 0 & -t_{01}t_{02} & 0 & 0 \\ -t_{13}t_{03} & 0 & t_{13}t_{01} & 0 & -t_{01}t_{03} & 0 \\ 0 & -t_{23}t_{03} & t_{23}t_{02} & 0 & 0 & -t_{02}t_{03} \\ 0 & 0 & 0 & -t_{23}t_{13} & t_{23}t_{12} & -t_{12}t_{13} \end{pmatrix}.$$

Conditions (R1-R3) from Theorem 6.14 are easy to check:

R1. Note that $\text{rk } \nabla h(\bar{d}) \geq 3$ (generically), and that $\dim D = 3$. Thus, ACQ holds.

R2. The feasible set is independent of θ , so $\mathcal{W} = \Theta \times \bar{X}$.

R3. By Example 6.10, it is enough to check that $\nabla_{z_{01}} h(\bar{d})$ is injective. This follows by noticing that the first row of $\nabla h(\bar{d})$ is nonzero.

For the remaining condition, Assumption RS, we consider the vector $\mu \in \mathbb{R}^{\binom{V}{3}}$ with entries

$$\mu_{012} = -t_{03}t_{13}t_{23}, \quad \mu_{013} = t_{02}t_{12}t_{23}, \quad \mu_{023} = -t_{01}t_{12}t_{13}, \quad \mu_{123} = t_{01}t_{02}t_{03}. \quad (6.19)$$

We will see that either μ or $-\mu$ satisfy Assumption RS. It is easy to see that $\mu^T \nabla h(\bar{d}) = 0$, so it remains to verify the positivity condition. As in Example 6.5, we consider the change of coordinates $u = y - \theta z_0$, and let $h'_{ijk}(z_0, z_{01}, u) := h_{ijk}(z_{01}, u + \theta z_0)$. It can be checked that $\nabla_{zz}^2 h'_{ijk} = 0$ if $ij \neq 01$, and the matrix \mathcal{A} from (6.12) is:

$$\begin{aligned} \mathcal{A}(\mu) &= \sum_{ijk} \mu_{ijk} \nabla_{zz}^2 h'_{ijk} = \mu_{012} \begin{pmatrix} t_{01}^2(t_{01}+2t_{12})^2 & -t_{02}^2-t_{12}^2 \\ -t_{02}^2-t_{12}^2 & 1 \end{pmatrix} + \mu_{013} \begin{pmatrix} t_{01}^2(t_{01}+2t_{13})^2 & -t_{03}^2-t_{13}^2 \\ -t_{03}^2-t_{13}^2 & 1 \end{pmatrix} \\ &= t_{23}^2(t_0 + t_1 - t_2 - t_3) \begin{pmatrix} t_{01}^4 & -t_{01}^2 \\ -t_{01}^2 & 1 \end{pmatrix}. \end{aligned}$$

Notice that $\mathcal{A}(\mu) \neq 0$ and that $\mathcal{A}(\mu)\bar{z} = 0$ where $\bar{z} := (1, t_{01}^2)$. It follows that either

$\mathcal{A}(\mu)|_{(\bar{z})^\perp} \succ 0$ or $\mathcal{A}(-\mu)|_{(\bar{z})^\perp} \succ 0$, and thus Assumption RS holds.

Remark 6.17. The argument from above can be readily adapted to the case $V = \{0, \dots, n\}$, $E = \binom{V}{2} \setminus \{01\}$. Conditions (R1-R3) are easy. Consider the vector $\mu \in \mathbb{R}^{\binom{V}{3}}$ whose only nonzero entries are the ones in (6.19). The matrix $\mathcal{A}(\mu)$ is then the same as before, and thus Assumption RS holds.

6.8 Additional proofs

Proof of Lemma 6.10. Let $\gamma_\theta := q_\theta(y_\theta)$ be the optimal value. Let us first show that $\gamma_\theta \rightarrow \gamma_{\bar{\theta}}$ as $\theta \rightarrow \bar{\theta}$. Since $\gamma_\theta = q_\theta(y_\theta) \leq q_\theta(\bar{y})$ then

$$\limsup_{\theta \rightarrow \bar{\theta}} \gamma_\theta \leq \lim_{\theta \rightarrow \bar{\theta}} q_\theta(\bar{y}) = q_{\bar{\theta}}(\bar{y}) = \gamma_{\bar{\theta}}$$

Let $\rho_\theta := \min_{y \in \mathbb{R}^n} q_\theta(y)$ be the unconstrained minimum of q_θ . Clearly $\rho_\theta \leq \gamma_\theta$. Since q_θ is convex quadratic, there is an explicit formula for ρ_θ , and it can be checked that $\rho_\theta \rightarrow \rho_{\bar{\theta}}$. Therefore,

$$\gamma_{\bar{\theta}} = \rho_{\bar{\theta}} = \lim_{\theta \rightarrow \bar{\theta}} \rho_\theta \leq \liminf_{\theta \rightarrow \bar{\theta}} \gamma_\theta.$$

It follows that $\lim_{\theta \rightarrow \bar{\theta}} \gamma_\theta = \gamma_{\bar{\theta}}$, as we claimed.

Let us now show that $y_\theta \rightarrow \bar{y}$. Since $g_{\bar{\theta}}$ is strictly convex and \bar{y} is the minimizer, it is sufficient to see that $g_{\bar{\theta}}(y_\theta) \rightarrow g_{\bar{\theta}}(\bar{y})$. Let $t > \gamma_{\bar{\theta}}$ be arbitrary and let $S_\theta := \{y \in \mathbb{R}^n : g_\theta(y) \leq t\}$. Since $g_{\bar{\theta}}$ is strictly convex and g_θ depends continuously on θ , there is a compact set S such that $S_\theta \subseteq S$ for all θ sufficiently close to $\bar{\theta}$. Since $\gamma_\theta \rightarrow \gamma_{\bar{\theta}}$, then $g_\theta(y_\theta) < t$ when θ is close to $\bar{\theta}$. Therefore, we may assume that $y_\theta \in S$ for all θ . Denoting $\|\cdot\|_S$ the infinity norm on the compact set S , then

$$|g_{\bar{\theta}}(y_\theta) - g_{\bar{\theta}}(\bar{y})| \leq |g_{\bar{\theta}}(y_\theta) - g_\theta(y_\theta)| + |g_\theta(y_\theta) - g_{\bar{\theta}}(\bar{y})| \leq \|g_{\bar{\theta}} - g_\theta\|_S + |\gamma_\theta - \gamma_{\bar{\theta}}| \xrightarrow{\theta \rightarrow \bar{\theta}} 0$$

as wanted. \square

Proof of Corollary 6.13. We will use a simple variation of Theorem 6.12. Recall the definitions of $\mathcal{C}_\theta(\mu)$, $\mathcal{Q}_\theta(\lambda)$ from (6.10). Since $\mathcal{C}_\theta(\mu)$ is a submatrix of $\mathcal{Q}_\theta(\lambda)$, where $\lambda = (\lambda_0, \mu)$, then

their eigenvalues satisfy $\nu_1(\mathcal{C}_\theta(\mu)) \leq \nu_2(\mathcal{Q}_\theta(\lambda))$. The proof of Theorem 6.12 relied on lower bounding $\nu_2(\mathcal{Q}_\theta(\lambda))$, but we can instead bound $\nu_1(\mathcal{C}_\theta(\mu))$. Consequently, Theorem 6.12 can be modified by replacing $\mathcal{Q}_\theta(\lambda)$ with $\mathcal{C}_\theta(\mu)$, and $\nu_2(\bar{\mathcal{Q}})$ with $\nu_1(\mathcal{C}_{\bar{\theta}}(\bar{\mu}))$. It only remains to compute the constants from Assumptions 6.12(i-iii):

- (i) $\nu_1(\mathcal{C}_{\bar{\theta}}(\bar{\mu})) = 1$ since $\bar{\mu} = 0$ and thus $\mathcal{C}_{\bar{\theta}}(\bar{\mu}) = \nabla^2(\|y - \bar{\theta}\|^2) = \text{id}_n$.
- (ii) $K = \frac{2}{\sigma_s}$ since $\|\mu_\theta\| \leq \frac{2}{\sigma_s}\|\bar{y} - \theta\|$ for any $\theta \in \Theta(\bar{y})$ by Lemma 6.11.
- (iii) $L = 0$ since $\mathcal{C}_\theta(\bar{\mu}) = \text{id}_n$ is independent of θ .

□

Proof of Lemma 6.15. From the definition of the Aubin property it is clear that there exists a neighborhood $U_0 \ni \bar{y}$ such that \mathfrak{F} has the Aubin property at \bar{p} for y , for any $y \in U_0 \cap \mathfrak{F}(\bar{p})$. We may assume that U_0 is closed. Let $\mathfrak{F}_0 : p \mapsto \mathfrak{F}(p) \cap U_0$, and note that it has closed graph since \mathfrak{F} does. Thus, \mathfrak{F}_0 is outer semicontinuous [108, Thm 5.7]. The lemma follows from [108, Thm 9.38].

□

Chapter 7

Summary

This thesis focused on the use of graphical and geometrical structure in problems involving polynomial equations, with applications in sciences and engineering. The first part of this thesis (Chapters 3 and 4) investigated graphical structure for solving polynomial systems, as well as for computing permanents and related functions. The second part (Chapters 5 and 6) explored semidefinite relaxations of polynomial optimization problems, obtaining more efficient relaxations and better guarantees.

In Chapter 3 we introduced a new data structure to represent polynomial systems, chordal networks, aimed at exploiting graphical structure. We showed that several interesting families of polynomial systems have small chordal network representations, even though the number of components is exponentially large. We also saw that chordal networks can be efficiently used to compute several properties of the underlying variety (e.g., cardinality, dimension). Chordal networks showed orders of magnitude reduction over existing methods in cases from algebraic statistics and vector addition systems.

In Chapter 4 we obtained a novel algorithm to compute permanents of matrices with structured sparsity. The complexity of our method is $\tilde{O}(n2^\omega)$, where ω is the treewidth of the bipartite adjacency graph. We also derived efficient algorithms to compute mixed discriminants and hyperdeterminants of structured arrays. Finally, we showed that computing mixed volumes and solving generic polynomial systems is $\#P$ -hard, even when the treewidth is bounded.

In Chapter 5 we proposed a novel SOS relaxation for optimization problems over algebraic

varieties. Our method represents the variety with a generic set of complex samples, as opposed to previous techniques which rely on an algebraic description. Importantly, we can reduce the size of the relaxation by taking advantage of the geometric structure of the variety. Our methods can be directly used for several varieties of interest, such as $SO(n)$, Grassmannians, and rank k tensors. They also allow the use, for the first time, of tools from numerical algebraic geometry to SOS optimization.

In Chapter 6 we studied SDP relaxations of parametrized optimization problems over varieties. We assumed that the relaxation is tight (or exact) for a nominal value of the parameters, and identified sufficient conditions (and quantitative bounds) for the relaxation to remain tight after small perturbations of the parameters. We showed several applications to estimation problems (e.g., the triangulation problem, rotation synchronization, rank-one tensor approximation), proving that SDP relaxations solve these problems exactly under low noise.

Bibliography

- [1] C. Aholt, S. Agarwal, and R. Thomas. A QCQP approach to triangulation. In *ECCV (1)*, volume 7572 of *Lecture Notes in Computer Science*, pages 654–667. Springer, 2012.
- [2] S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, 100(6):509–516, 1978.
- [3] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k -tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.
- [4] J.-P. Aubin and H. Frankowska. *Set-valued analysis*. Springer Science & Business Media, 2009.
- [5] P. Aubry, D. Lazard, and M. M. Maza. On the theories of triangular sets. *Journal of Symbolic Computation*, 28(1):105–124, 1999.
- [6] S. V. Avgustinovich. Multidimensional permanents in enumeration problems. *Journal of Applied and Industrial Mathematics*, 4(1):19–20, 2010.
- [7] N. Balaji and S. Datta. Bounded treewidth and space-efficient linear algebra. In *Theory and Applications of Models of Computation*, volume 9076 of *Lecture Notes in Computer Science*, pages 297–308. Springer International Publishing, 2015.
- [8] R. B. Bapat. Mixed discriminants of positive semidefinite matrices. *Linear Algebra and its Applications*, 126:107–124, 1989.
- [9] A. Barvinok. New algorithms for linear k -matroid intersection and matroid k -parity problems. *Mathematical Programming*, 69(1-3):449–470, 1995.
- [10] A. Barvinok. Two algorithmic results for the traveling salesman problem. *Mathematics of Operations Research*, 21(1):65–84, 1996.
- [11] A. Barvinok. Computing mixed discriminants, mixed volumes, and permanents. *Discrete & Computational Geometry*, 18(2):205–237, 1997.
- [12] A. Barvinok and A. Samorodnitsky. Computing the partition function for perfect matchings in a hypergraph. *Combinatorics, Probability and Computing*, 20(06):815–835, 2011.
- [13] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. Bertini: Software for numerical algebraic geometry. Available at www.nd.edu/~sommese/bertini, 2006.

- [14] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler. *Numerically solving polynomial systems with Bertini*, volume 25. SIAM, 2013.
- [15] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms*. John Wiley & Sons, 2013.
- [16] A. Beck and Y. C. Eldar. Strong duality in nonconvex quadratic optimization with two quadratic constraints. *SIAM Journal on Optimization*, 17(3):844–860, 2006.
- [17] 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*, 2005.
- [18] D. N. Bernstein. The number of roots of a system of equations. *Functional Analysis Appl.*, 9(3):1–4, 1975.
- [19] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, Belmont (Mass.), 1999.
- [20] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets Möbius: fast subset convolution. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2007.
- [21] J. R. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.
- [22] G. Blekherman, P. Parrilo, and R. Thomas. *Semidefinite optimization and convex algebraic geometry*, volume 13. MOS-SIAM Series on Optimization, 2013.
- [23] G. Blekherman, G. Smith, and M. Velasco. Sums of squares and varieties of minimal degree. *Journal of the American Mathematical Society*, 29(3):893–913, 2016.
- [24] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 226–234. ACM, 1993.
- [25] H. L. Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theoretical computer science*, 209(1):1–45, 1998.
- [26] H. L. Bodlaender and A. M. Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.
- [27] H. L. Bodlaender and A. M. Koster. Treewidth computations I. Upper bounds. *Information and Computation*, 208(3):259 – 275, 2010.
- [28] J. F. Bonnans and A. Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.
- [29] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys (CSUR)*, 24(3):293–318, 1992.

- [30] B. Buchberger. *An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal (in German)*. PhD thesis, University of Innsbruck, Austria, 1965.
- [31] P. Bürgisser. The computational complexity of immanants. *SIAM Journal on Computing*, 30(3):1023–1040, 2000.
- [32] A. Cayley. Mémoire sur les hyperdéterminants. *Journal für die reine und angewandte Mathematik*, 30:1–37, 1846.
- [33] L. Cayton and S. Dasgupta. Robust Euclidean embedding. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 169–176. ACM, 2006.
- [34] M. Chardin. Une majoration de la fonction de Hilbert et ses conséquences pour l’interpolation algébrique. *Bulletin de la Société Mathématique de France*, 117(3):305–318, 1989.
- [35] M. T. Chu and N. T. Trendafilov. The orthogonally constrained regression revisited. *Journal of Computational and Graphical Statistics*, 10(4):746–771, 2001.
- [36] D. Cifuentes. Exploiting chordal structure in systems of polynomial equations. Master’s thesis, Massachusetts Institute of Technology, 2014.
- [37] D. Cifuentes, S. Agarwal, P. A. Parrilo, and R. R. Thomas. On the local stability of semidefinite relaxations. *arXiv preprint arXiv:1710.04287*, 2017.
- [38] D. Cifuentes and P. A. Parrilo. An efficient tree decomposition method for permanents and mixed discriminants. *Linear Algebra and its Applications*, 493:45 – 81, 2016.
- [39] D. Cifuentes and P. A. Parrilo. Exploiting chordal structure in polynomial ideals: a Gröbner bases approach. *SIAM Journal on Discrete Mathematics*, 30(3):1534–1570, 2016.
- [40] D. Cifuentes and P. A. Parrilo. Chordal — a Macaulay2 package. Available at <http://www.mit.edu/~diegcif>, 2017.
- [41] D. Cifuentes and P. A. Parrilo. Chordal networks of polynomial ideals. *SIAM Journal on Applied Algebra and Geometry*, 1(1):73–110, 2017.
- [42] D. Cifuentes and P. A. Parrilo. Sampling algebraic varieties for sum of squares programs. *SIAM Journal on Optimization (to appear)*, 2017.
- [43] C. Ciliberto. Geometric aspects of polynomial interpolation in more variables and of Waring’s problem. In *European Congress of Mathematics*, volume 201 of *Progress in Mathematics*, pages 289–316. Springer, 2001.
- [44] B. Codenotti, V. Crespi, and G. Resta. On the permanent of certain $(0, 1)$ Toeplitz matrices. *Linear Algebra and its Applications*, 267:65–100, 1997.
- [45] B. Courcelle, J. A. Makowsky, and U. Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1):23–52, 2001.

- [46] D. A. Cox, J. Little, and D. O’Shea. *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. Springer, 2007.
- [47] D. A. Cox, J. B. Little, and D. O’Shea. *Using algebraic geometry*, volume 185. Springer, 2005.
- [48] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank- (r_1, r_2, \dots, r_n) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.
- [49] J. A. De Loera, J. Lee, P. N. Malkin, and S. Margulies. Hilbert’s Nullstellensatz and an algorithm for proving combinatorial infeasibility. In *Proceedings of the Twenty-first International Symposium on Symbolic and Algebraic Computation*, ISSAC’08, pages 197–206, New York, 2008. ACM.
- [50] R. Dechter. *Constraint processing*. Morgan Kaufmann, 2003.
- [51] W. Decker, G. Greuel, G. Pfister, and H. Schönemann. SINGULAR 3-1-6 — A computer algebra system for polynomial computations. <http://www.singular.uni-kl.de>, 2012.
- [52] J. W. Demmel. *Applied Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.
- [53] P. Diaconis, D. Eisenbud, and B. Sturmfels. *Mathematical Essays in honor of Gian-Carlo Rota*, chapter Lattice Walks and Primary Decomposition, pages 173–193. Birkhäuser, Boston, MA, 1998.
- [54] A. L. Dontchev and R. T. Rockafellar. *Implicit functions and solution mappings: a view from variational analysis*. Springer Monographs on Mathematics. Springer, 2009.
- [55] S. J. Dow and P. M. Gibson. Permanents of d -dimensional matrices. *Linear Algebra and its Applications*, 90:133–145, 1987.
- [56] M. Dyer, P. Gritzmann, and A. Hufnagel. On the complexity of computing mixed volumes. *SIAM Journal on Computing*, 27(2):356–400, 1998.
- [57] D. Eisenbud. *Commutative algebra: with a view toward algebraic geometry*, volume 150. Springer Science & Business Media, 2013.
- [58] S. N. Evans, B. Sturmfels, and C. Uhler. Commuting birth-and-death processes. *The Annals of Applied Probability*, pages 238–266, 2010.
- [59] J.-C. Faugère. Finding all the solutions of Cyclic 9 using Gröbner basis techniques. In *Computer Mathematics - Proceedings of the Fifth Asian Symposium (ASCM 2001)*, volume 9, pages 1–12. World Scientific, 2001.
- [60] A. V. Fiacco and Y. Ishizuka. Sensitivity and stability analysis for nonlinear programming. *Annals of Operations Research*, 27(1):215–235, 1990.
- [61] P. Finsler. Über das Vorkommen definiter und semidefiniter Formen in Scharen quadratischer Formen. *Commentarii Mathematici Helvetici*, 9(1):188–192, 1936.

- [62] U. Flarup, P. Koiran, and L. Lyaudet. On the expressive power of planar perfect matching and permanents of bounded treewidth matrices. In *Algorithms and Computation*, volume 4835 of *Lecture Notes in Computer Science*, pages 124–136. Springer, 2007.
- [63] J. Fredriksson and C. Olsson. Simultaneous multiple rotation averaging using Lagrangian duality. In *Asian Conference on Computer Vision*, pages 245–258. Springer, 2012.
- [64] S. Gao. Counting zeros over finite fields using Gröbner bases. Master’s thesis, Carnegie Mellon University, 2009.
- [65] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.
- [66] J. Gouveia, P. Parrilo, and R. Thomas. Theta bodies for polynomial ideals. *SIAM J. Optim.*, 20(4):2097–2118, 2010.
- [67] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at <http://www.math.uiuc.edu/Macaulay2/>.
- [68] L. Gurvits. On the complexity of mixed discriminants and related problems. In *Mathematical Foundations of Computer Science*, pages 447–458. Springer, 2005.
- [69] A. G. Hadigheh, O. Romanko, and T. Terlaky. Sensitivity analysis in convex quadratic optimization: simultaneous perturbation of the objective and right-hand-side vectors. *Algorithmic Operations Research*, 2(2):94, 2007.
- [70] J. Herzog, T. Hibi, F. Hreinsdóttir, T. Kahle, and J. Rauh. Binomial edge ideals and conditional independence statements. *Advances in Applied Mathematics*, 45(3):317 – 333, 2010.
- [71] A. Heyden and K. Åström. Algebraic properties of multilinear constraints. *Mathematical Methods in the Applied Sciences*, 20(13):1135–1162, 1997.
- [72] C. J. Hillar and L.-H. Lim. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):1–45, 2013.
- [73] E. Hubert. *Notes on triangular sets and triangulation-decomposition algorithms I: Polynomial systems*. Springer, 2003.
- [74] J. C. Jantzen. Nilpotent orbits in representation theory. In *Lie theory*, pages 1–211. Springer, 2004.
- [75] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697, 2004.
- [76] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.
- [77] T. Kahle. Decompositions of binomial ideals. *Annals of the institute of statistical mathematics*, 62(4):727–745, 2010.

- [78] M. Kalkbrenner. A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. *Journal of Symbolic Computation*, 15(2):143–167, 2 1993.
- [79] E. Kaltofen. Polynomial factorization 1987–1991. In I. Simon, editor, *Proceedings of the 1st Latin American Symposium on Theoretical Informatics*, pages 294–313, Berlin, Heidelberg, April 1992. Springer.
- [80] P. W. Kasteleyn. Graph theory and crystal physics. *Graph theory and theoretical physics*, 1:43–110, 1967.
- [81] S. Kim and M. Kojima. Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations. *Computational Optimization and Applications*, 26(2):143–154, 2003.
- [82] D. E. Knuth. *Combinatorial Algorithms, Part 1*, volume 4A of *The Art of Computer Programming*. Addison-Wesley Professional, 2011.
- [83] S. R. Kosaraju. Decidability of reachability in vector addition systems. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC ’82, pages 267–281, New York, NY, USA, 1982. ACM.
- [84] J. M. Landsberg and J. Weyman. On the ideals and singularities of secant varieties of Segre varieties. *Bulletin of the London Mathematical Society*, 2007.
- [85] J. B. Lasserre. *Moments, positive polynomials and their applications*, volume 1 of *Series on Optimization and Its Applications*. World Scientific, 2009.
- [86] M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging applications of algebraic geometry*, pages 157–270. Springer, 2009.
- [87] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.
- [88] D. Lazard. Solving zero-dimensional algebraic systems. *Journal of symbolic computation*, 13(2):117–131, 1992.
- [89] F. Lemaire, M. M. Maza, and Y. Xie. The RegularChains library. In *Maple conference*, volume 5, pages 355–368, 2005.
- [90] A. B. Levy, R. A. Poliquin, and R. T. Rockafellar. Stability of locally optimal solutions. *SIAM Journal on Optimization*, 10(2):580–604, 2000.
- [91] Z. Liu and L. Vandenberghe. Low-rank structure in semidefinite programs derived from the KYP lemma. In *46th IEEE Conference on Decision and Control*, pages 5652–5659, 2007.
- [92] J. Löfberg and P. A. Parrilo. From coefficients to samples: a new approach to SOS optimization. In *43rd IEEE Conference on Decision and Control*, volume 3, pages 3154–3159, 2004.

- [93] L. Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.
- [94] J.-G. Luque and J.-Y. Thibon. Hankel hyperdeterminants and Selberg integrals. *Journal of Physics A: Mathematical and General*, 36(19):5267–5292, 2003.
- [95] M. M. Maza. On triangular decompositions of algebraic varieties. Presented at the MEGA-2000 Conference, NAG, UK, 2000.
- [96] K. Meer. An extended tree-width notion for directed graphs related to the computation of permanents. In *Computer Science Theory and Applications*, pages 247–260. Springer, 2011.
- [97] M. Mevissen and M. Kojima. SDP relaxations for quadratic optimization problems derived from polynomial optimization problems. *Asia-Pacific Journal of Operational Research*, 27(01):15–38, 2010.
- [98] H. Minc. Permanent compounds and permanents of $(0, 1)$ -circulants. *Linear Algebra and its Applications*, 86:11–42, 1987.
- [99] M. Monagan and R. Pearce. Sparse polynomial division using a heap. *Journal of Symbolic Computation*, 46(7):807–822, 2011.
- [100] B. S. Mordukhovich, R. T. Rockafellar, and M. E. Sarabi. Characterizations of full stability in constrained optimization. *SIAM Journal on Optimization*, 23(3):1810–1849, 2013.
- [101] J. Nie, J. Demmel, and B. Sturmfels. Minimizing polynomials via sum of squares over the gradient ideal. *Mathematical programming*, 106(3):587–606, 2006.
- [102] J. Nie and L. Wang. Semidefinite relaxations for best rank-1 tensor approximations. *SIAM Journal on Matrix Analysis and Applications*, 35(3):1155–1179, 2014.
- [103] S. Onn and U. G. Rothblum. Convex combinatorial optimization. *Discrete & Computational Geometry*, 32(4):549–566, 2004.
- [104] P. A. Parrilo. Exploiting structure in sum of squares programs. In *42nd IEEE Conference on Decision and Control*, volume 5, pages 4664–4669, 2003.
- [105] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320, 2003.
- [106] F. Permenter and P. A. Parrilo. Selecting a monomial basis for sums of squares programming over a quotient ring. In *IEEE 51st Annual Conference on Decision and Control*, pages 1871–1876, 2012.
- [107] I. Pólik and T. Terlaky. A survey of the S-lemma. *SIAM review*, 49(3):371–418, 2007.
- [108] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

- [109] T. Roh, B. Dumitrescu, and L. Vandenberghe. Multidimensional FIR filter design via trigonometric sum-of-squares optimization. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):641–650, 2007.
- [110] T. Roh and L. Vandenberghe. Discrete transforms, semidefinite programming, and sum-of-squares representations of nonnegative polynomials. *SIAM Journal on Optimization*, 16(4):939–964, 2006.
- [111] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on computing*, 5(2):266–283, 1976.
- [112] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard. A certifiably correct algorithm for synchronization over the special Euclidean group. In *Intl. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, San Francisco, CA, Dec. 2016.
- [113] F. Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5):433–461, 1999.
- [114] H. J. Ryser. *Combinatorial mathematics*. Wiley, New York, 1963.
- [115] T. Sauer. Polynomial interpolation in several variables: lattices, differences, and ideals. *Studies in Computational Mathematics*, 12:191–230, 2006.
- [116] J. B. Saxe. Embeddability of weighted graphs in k -space is strongly NP-hard. In *Proceedings of the 17th Allerton Conference on Communications, Control, and Computing*, pages 480–489, 1979.
- [117] N. Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79, 2009.
- [118] C. Scheiderer. Positivity and sums of squares: a guide to recent results. In *Emerging applications of algebraic geometry*, pages 271–324. Springer, 2009.
- [119] K. Schmüdgen. The k -moment problem for compact semi-algebraic sets. *Mathematische Annalen*, 289(1):203–206, 1991.
- [120] R. Schneider. *Convex bodies: the Brunn-Minkowski theory*, volume 44 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1993.
- [121] M. Schwartz. Efficiently computing the permanent and Hafnian of some banded Toeplitz matrices. *Linear Algebra and its Applications*, 430(4):1364–1374, 2009.
- [122] M. Schweighofer. Global optimization of polynomials using gradient tentacles and sums of squares. *SIAM Journal on Optimization*, 17(3):920–942, 2006.
- [123] N. Z. Shor. *Nondifferentiable optimization and polynomial problems*, volume 24. Springer Science & Business Media, 2013.
- [124] A. J. Sommese and C. W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*, volume 99. World Scientific, 2005.

- [125] W. Stein et al. Sage: Open source mathematical software, 2008.
- [126] R. J. Stern and H. Wolkowicz. Indefinite trust region subproblems and nonsymmetric eigenvalue perturbations. *SIAM Journal on Optimization*, 5(2):286–313, 1995.
- [127] B. Sturmfels. *Gröbner bases and convex polytopes*, volume 8 of *University Lecture Series*. American Mathematical Soc., Providence, RI, 1996.
- [128] B. Sturmfels. *Solving systems of polynomial equations*, volume 97 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Soc., Providence, RI, 2002.
- [129] K. Temme and P. Wocjan. Efficient computation of the permanent of block factorizable matrices. *arXiv preprint arXiv:1208.6589*, 2012.
- [130] H. Temperley and M. E. Fisher. Dimer problem in statistical mechanics—an exact result. *Philosophical Magazine*, 6(68):1061–1063, 1961.
- [131] T. Theobald and T. De Wolff. Approximating amoebas and coamoebas by sums of squares. *Mathematics of Computation*, 84(291):455–473, 2015.
- [132] M. C. Tichy. Sampling of partially distinguishable bosons and the relation to the multidimensional permanent. *Physical Review A*, 91(2):022316, 2015.
- [133] R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical programming*, 95(2):189–217, 2003.
- [134] L. G. Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.
- [135] J. M. van Rooij, H. L. Bodlaender, and P. Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *Algorithms-ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009.
- [136] L. Vandenberghe and M. S. Andersen. Chordal graphs and semidefinite optimization. *Foundations and Trends in Optimization*, 1(4):241–433, 2014.
- [137] J. Verschelde. Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation. *ACM Transactions on Mathematical Software (TOMS)*, 25(2):251–276, 1999.
- [138] T. Viklands. *Algorithms for the weighted orthogonal Procrustes problem and other least squares problems*. PhD thesis, Umea University, Sweden, 2006.
- [139] R. Villarreal. *Monomial algebras*, volume 8. CRC Press, 2015.
- [140] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 2013.
- [141] P. O. Vontobel. The Bethe permanent of a nonnegative matrix. *Information Theory, IEEE Transactions on*, 59(3):1866–1901, 2013.

- [142] D. Wang. Computing triangular systems and regular systems. *Journal of Symbolic Computation*, 30(2):221 – 236, 2000.
- [143] D. Wang. *Elimination methods*. Springer Science & Business Media, 2001.
- [144] D. Wang. *Elimination Practice: Software Tools and Applications*. Imperial College Press, London, 2003.
- [145] H. Wang, S. Yan, D. Xu, X. Tang, and T. Huang. Trace ratio vs. ratio trace for dimensionality reduction. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.
- [146] L. Wang and A. Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference*, page iat005, 2013.
- [147] Y. Watanabe and M. Chertkov. Belief propagation and loop calculus for the permanent of a non-negative matrix. *Journal of Physics A: Mathematical and Theoretical*, 43(24):242002, 2010.
- [148] I. Wegener. *Branching programs and binary decision diagrams: theory and applications*, volume 4 of *Discrete Mathematics and Applications*. SIAM, 2000.
- [149] W.-T. Wu. On the decision problem and the mechanization of theorem-proving in elementary geometry. *Scientia Sinica*, 21(2):159–172, 1978.
- [150] Y. Ye and S. Zhang. New results on quadratic minimization. *SIAM Journal on Optimization*, 14(1):245–267, 2003.
- [151] L. H. Zhang and R. C. Li. Maximization of the sum of the trace ratio on the Stiefel manifold, I: Theory. *Science China Mathematics*, 57:2495–2508, 2014.
- [152] L. H. Zhang and R. C. Li. Maximization of the sum of the trace ratio on the Stiefel manifold, II: Computation. *Science China Mathematics*, 57:1–18, 2014.
- [153] S. Zhang. Quadratic maximization and semidefinite relaxation. *Mathematical Programming*, 87(3):453–465, 2000.