This is a working draft. Feedback is welcome.

Direct comments to apapalia@umich.edu or leave an issue on Github:
https://github.com/alanpapalia/Intro-Certifiable-Robotics

An Introduction to Certifiably Correct Optimization in Robotics

Alan Papalia*, Connor Holmes*, Frederike Dümbgen*,
Corbinian Schlosser, Tobia Marcucci

May 20, 2025

Contents

1 Introduction

Optimization is a workhorse of modern robotics. State-of-the-art approaches in fields as diverse as in motion planning [55, 92, 107, 78, 84, 65, 66], control [17, 10, 79, 12], perception [9, 32], and state estimation [10, 7, 79] rely on optimization. While the use of optimization is ubiquitous in robotics, the majority of problems exhibit some form of non-convexity. Because of the need for real-time performance in robotics applications, most solutions resort to local optimization techniques without any guarantees of global optimality. This has two major implications: (i) the solutions may be suboptimal and (ii) when they are, this usually happens without notice. In low-stakes applications (e.g., path-planning for warehouse logistics) relying on suboptimal solutions may result in reduced efficiency. In high-stakes applications (e.g., state-estimation for autonomous driving) a suboptimal solution may represent a highly inaccurate estimate of the state of the world and lead to serious, irrecoverable failures.

Certifiably correct optimization is a growing area of research that aims to address these limitations by providing guarantees of global optimality for many non-convex optimization problems. Particularly, a certifiably correct algorithm will both (i) under certain conditions (e.g., sufficiently low levels of sensor noise) find a globally optimal solution to the given problem and (ii) provide a *a posteriori* certificate of optimality [4]. These properties make certifiably correct optimization an exciting direction for the future of robotics, mitigating the issues of local optimization techniques and imbuing robotics with increased safety and reliability.

At the core of the techniques presented in this paper is the concept of convex relaxations. At a high level, the philosophy is that we replace a difficult-to-solve problem with an easier problem, where the latter provides at least an approximate solution to the original problem. As we will see, many problems in robotics admit relatively *tight* relaxations, meaning that the solution of the relaxation can be used to extract the solution to the original, non-convex problem, or to at least certify candidate solutions. A visualization of this idea is given in Fig. 1.

More specifically, we focus in this paper on convex relaxations of *polynomial optimization problems (POPs)*, which take the form of semidefinite programs [88, 59, 74]. This class of problems has many applications in robotics, as the following two concrete examples illustrate.

1.1 Motivational Examples

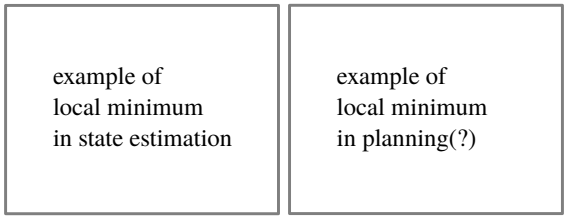


Figure 1: Visualization of the core principle of the algorithms presented in this tutorial paper. Top row: an example polynomial optimization problem (see **??**), its local and global minima. Bottom row: examples of local minima that standard solvers used in robotics are likely to converge to, for an example state estimation problem (rotation synchronization, left) and an example planning problem (TBD, right).

First, consider simultaneous localization and mapping (SLAM), a ubiquitous problem with a long history in both computer vision and robotics communities [25]: determining the trajectory of a moving device (a robot or camera, for example), along with a representation of the environment. Using a probabilistic viewpoint [94, 7], finding the maximum likelihood estimate boils down to solving a non-convex optimization problem. When using a state-of-the-art local solver, convergence to poor local minima, as shown in Figure 1, is likely. This is potentially dangerous: relying on such estimates for downstream tasks can clearly lead to unsafe behavior.

Second, consider motion planning – another classic problem in robotics: solving for an optimal robot motion under constraints such as joint limits, actuation constraints, or obstacle avoidance. Even for very simple settings, such as the one shown in Figure 1 (a planar robot aiming to find the shortest path from given start to end points while avoiding obstacles). The problem is non-convex and the quality of different feasible solutions (corresponding to different homotoy classes) may vary a lot. Using a local minimum here can lead to inefficient operations.

Besides these two examples, many other use cases of convex relaxations, and SDPs in general, exist in the robotics literature. A non-exhaustive list includes, for example, robust estimation [102], inverse kinematics [41], optimal planning including rigid-body dynamics [93], planning through contact [44]. The described techniques also have a long history in the control community for topics such as automatic discovery of Lyapunov functions or determination of invariant sets such as regions of attraction. An overview of these applications can be found in [27].

1.2 Goal and structure for this paper

While certifiably correct optimization is a promising paradigm that admits useful guarantees for large classes of problems, the tools of certifiably correct optimization remains inaccessible to many roboticists as the necessary mathematical background (semidefinite programming and polynomial optimization) is relatively exotic compared to common robotics knowledge and requires piecing together information across several areas of applied mathematics. This tutorial paper aims to lower the barrier of entry for roboticists to the field of certifiably correct optimization. While the techniques in certifiably correct optimization have found great success in a variety of robotics applications, there is a lack of accessible resources for practitioners to understand (i) the common framework underlying these methods, (ii) the landscape of existing methodologies, and (iii) how to apply these methods to their own problems. This paper aims to fill this gap by providing a structured introduction to the fundamentals of this field.

1.2.1 Intended audience

This document is intended for practitioners who are familiar with the basics of optimization and linear algebra and are interested in using certifiably correct algorithms in their work. The reader should be familiar with posing problems in the language of optimization. This document is aimed towards the robotics community, in which certifiably correct algorithms have seen growing interest in recent years. However, the foundations and techniques discussed here are broadly applicable to a wide range of optimization problems.

1.2.2 Scope

The techniques presented in this paper – convex (in particular, semidefinite) relaxations – are by no means the only techniques for global optimization. Indeed, while they often provide global solutions *a posteriori*, they are not always guaranteed to do so. If *always* finding a global solution is of utmost importance, other techniques can be used that are designed to *always* find the global solution. For example, in polynomial optimization, the optimality conditions are simply a set of polynomial equations of which the roots can be found using symbolic [91] computational tools. Because such methods are known to have numerical issues, numeric [89] techniques have also attracted considerable attention; as well as hybrid techniques such as homotopy continuation methods [90, 67]. While this is still an active field of research, there remain scalability issues. Furthermore, these methods inherently "waste" significant efforts for finding *all* local minima rather than only the global minimum. Moving away from polynomial optimization, techniques such as branch and bound, typically applied to problems with combinatorial nature, are guaranteed to find

the global minimum, although worse-case in exponential time [62]. Note that concepts of this paper, such as finding tight relaxations, are often used within branch and bound frameworks for more effective pruning. Other optimization methods such as genetic algorithms [49] or Bayesian optimization [85] are very general as they only require function evaluations (also known as black-box methods), but while they empirically work well for exploring the entire feasible domain in an efficient way, they generally do not come with optimality guarantees. Methods using semidefinite relaxations lie at an interesting sweet spot between the methods that are always optimal, such as branch and bound or polynomial root finding methods, and local methods. For the particular problems encountered in robotics, experimental evidence suggests that the relaxations are often tight enough to yield tangible benefits, while being relatively cheap to solve thanks to exploiting problem structure. Because of these positive aspects, they are the main focus of this paper.

1.2.3 The structure of this paper

In Fig. 2 we represent the subsequent structure of this paper as a flowchart, which practitioners can follow to develop certifiably correct algorithms for their own problems. Each section of the paper is designed to be relatively self-contained, so that the reader can go to the section that is most relevant to their needs.

In Section 2 the paper will introduce fundamental concepts (e.g., Lagrangian duality theory, convex relaxations, and semidefinite programming) which represent the minimal mathematical background necessary to pose optimization problems in a form suitable for certifiably correct optimization with off-the-shelf tools. Importantly, modern approaches to obtain certificates of optimality depend on being able to produce a *tight* convex relaxation of the original optimization problem (i.e., a relaxation which shares an optimal solution with the original problem). Section 3 discusses different approaches to tighten the types of relaxations we work with in practice. Finally, Section 4 discusses different ways in which specific structure in the problem can be exploited to improve computational performance (decreased runtime and memory requirements) when solving certifiably correct optimization problems.

Each section ends with a set of practical considerations relevant to the topic of that section. These considerations attempt to best capture the lessons learned from the authors' experiences in applying certifiably correct optimization to real-world problems, focusing on the little details that can make a major difference in practical success.

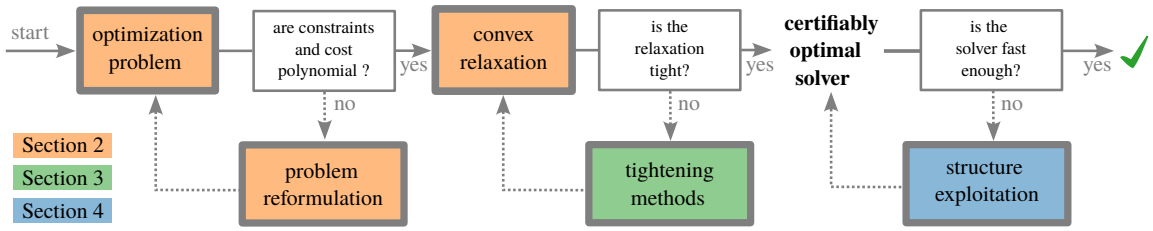


Figure 2: **Overview of the paper.** The flowchart summarizes the structure of the paper along with the core ideas of each section. In Section 2 (indicated in the figure with ■), we discuss the fundamentals of certifiably correct optimization in robotics, including how to formulate problems as a polynomial optimization problem (POP) and its convex (semidefinite) relaxation. In Section 3 (indicated in the figure with ■), we discuss various approaches to tighten the convex relaxation of the original problem. This is important, as tightness is a prerequisite for certifying optimality. Finally, in Section 4 (indicated in the figure with ■), we discuss how to exploit specific structure in different problems to improve the computational performance of certifiably correct optimization (i.e., reduce runtime and memory requirements).

2 Primer on SDP Relaxations

As discussed earlier, this paper focuses on a specific technique for certifiable optimization, namely using convex relaxations. In particular, the focus is on SDP relaxations which arise naturally in many problems of interest, such as in the two example problems introduced earlier. In this section, we discuss how to derive SDP relaxations and introduce the concept of relaxation tightness, providing optimization theory background as needed. We also provide some commonly used tools from linear algebra that can help the practitioner to formulate a relaxation.

More specifically, we are concerned with convex relaxations of POPs. POPs are defined as optimization problems with cost and constraints that can be expressed as *multivariate polynomial functions*. The most general form of a POP is as follows:

$$\begin{aligned} \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \quad & f(\boldsymbol{\theta}) \\ \text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{\theta}) = 0, \\ & \boldsymbol{g}(\boldsymbol{\theta}) \geq 0, \end{aligned} \tag{POP}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^q$, and $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^p$ are polynomial functions. As we will see, a surprising number of problems in robotics can be formulated in this way. Particularly in the field of state estimation, these problems can often be formulated without

inequality constraints, which can simplify their convex relaxations.

When considering the application of the methods described herein, the reader must first ask whether their particular problem can be expressed as a POP. The answer to this question is not always obvious: for example, cost functions that are *rational* can often be re-expressed as a polynomial by introducing a slack variable. Moreover, even if the cost and constraints are not *exactly* polynomial, they may be *well-approximated* by polynomials in a particular region of interest.

As it is currently expressed, it is not clear how we can relax (POP) to a convex program. This section aims to demonstrate one way such a relaxation can be obtained, which is also known under the name of Shor's relaxation [88]. This is not the only approach, and more recently, powerful generalizations of this approach, using the theory of moments [59] or the sums-of-squares approach [74], have been introduced. We chose to focus mostly on Shor's relaxation here because we deem it more approachable for the robotics community.

2.1 Standard Formulations for Certifiable Methods

Once it has been established that a problem is a POP, we can transform it into a *standard form* that admits a standard convex relaxation. A key realization at this point is the fact that *any* problem of the form shown in (POP) can be converted into an equivalent problem consisting only of second-degree (i.e., quadratic) polynomials by introducing additional variables and constraints. To see why this is true, consider the fact that the polynomial function $f(\theta) = \theta^3$, for example, can be re-expressed as a quadratic function in two variables, $f(\theta, \alpha) = \theta\alpha$, as long as we introduce a constraint that enforces the quadratic equality $\alpha = \theta^2$.

Polynomial Example: In general, the reformulation of (POP) to (QCQP-1) is not unique and can often be a daunting task for beginners. As such, we now provide a simplified example throughout this section that seeks to minimize a sixth-order, univariate polynomial:

$$\min_{\theta \in \mathbb{R}} \quad p(\theta) = \sum_{i=0}^{6} \alpha_i \theta^i. \tag{1}$$

In general, this idea can be applied recursively to convert any POP into a standard quadratically constrained quadratic program (QCQP):

$$\begin{aligned} f^* = \min_{\boldsymbol{x} \in \mathbb{R}^n} \quad & \boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{x}^T \boldsymbol{A}_i \boldsymbol{x} = b_i, \quad \forall i \in [p], \\ & \boldsymbol{x}^T \boldsymbol{B}_j \boldsymbol{x} \geq d_j, \quad \forall j \in [q], \end{aligned} \tag{QCQP-1}$$

where the objective and constraint matrices are symmetric, $\boldsymbol{C}, \boldsymbol{A}_i, \boldsymbol{B}_j \in \mathrm{Sym}^n$, and $b_i \in \mathbb{R}$.

In some cases, in particular when our original variables are matrix-valued, it is computationally advantageous to avoid the vectorized form as in (QCQP-1) and to use a matrix-valued form instead. For example, in the rotation synchronization (RS) example, it is advantageous to collect all $r \times r$ stacked rotation matrices in $\boldsymbol{Y} \in \mathbb{R}^{n \times r}$ rather than vectorizing them. Introducing the variable $\boldsymbol{Y} \in \mathbb{R}^{n \times r}$ (rather than $\boldsymbol{x} \in \mathbb{R}^{nr}$), where $n > r$, we get the form

$$\begin{aligned} f^* = \min_{\boldsymbol{Y} \in \mathbb{R}^{n \times r}} \quad & \mathrm{tr}(\boldsymbol{Y}^T \boldsymbol{C} \boldsymbol{Y}) \\ \text{s.t.} \quad & \mathrm{tr}(\boldsymbol{Y}^T \boldsymbol{A}_i \boldsymbol{Y}) = b_i, \quad \forall i \in [p], \\ & \mathrm{tr}(\boldsymbol{Y}^T \boldsymbol{B}_j \boldsymbol{Y}) \geq d_j, \quad \forall j \in [q], \end{aligned} \tag{QCQP-r}$$

we have used the trace operator $\mathrm{tr}(\cdot)$. This latter formulation constitutes the most general form of QCQP that we will treat in this paper, though it is often common to see the vector formulation, (QCQP-1). Note that (QCQP-1) is a special case of (QCQP-r) and, in general, it is possible to convert between them by using vectorization techniques (see Appendix A.1). It is worth noting that generally both (POP) and (QCQP-r) are nonconvex programs.[1] As such, finding their solutions is typically NP-Hard *unless* we can find a suitable convex relaxation.

A key technique in transforming (POP) into (QCQP-r) is the introduction of new *substitution variables and constraints* that render all the cost and constraints quadratic. As with any optimization problem, there is never a *unique* choice of $\boldsymbol{x}$ or $\boldsymbol{Y}$, but they generally contain powers of $\boldsymbol{\theta}$ and possibly additional substitution variables. For a multivariate polynomial of degree d, one choice is to add all monomials of $\boldsymbol{\theta}$ up to degree $\lceil d/2 \rceil$. However, as we explore further in Section 3.2, adding more degrees than strictly necessary may help in increasing the tightness of the relaxation. Once we have defined all of our variables, we can collect them into a single vector, $\boldsymbol{x}$.

Polynomial Example: To reformulate our POP as a QCQP we start by defining a new variable, $\boldsymbol{x} \in \mathbb{R}^4$, that can adequately represent this polynomial,

$$\boldsymbol{x}^\top = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \end{bmatrix} = \begin{bmatrix} 1 & \theta & \theta^2 & \theta^3 \end{bmatrix}.$$

In this example, x_2, and x_3 are substitution variables that represent the powers θ. We can enforce the relationships between these variables using the following quadratic

[1] Notable exceptions are, for example, QCQPs with convex cost ($\boldsymbol{C} \succeq 0$), affine equality constraints, and convex inequality constraints, or convex quadratic programs (QPs), which also belong to this broader class of problems.

substitution constraints:

$$x_1^2 - 1 = 0, \quad x_1^2 - x_2 * x_0 = 0, \quad x_2 * x_1 - x_3 x_0 = 0$$

To formulate the matrices in (QCQP-1) it is useful to consider an *outer-product variable matrix* that contains all possible *quadratic products* of our variables:

$$\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T.$$

: In our running example, the outer product matrix takes the following form:

$$\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^\top = \begin{bmatrix} x_0x_0 & x_0x_1 & x_0x_2 & x_0x_3 \\ x_1x_0 & x_1x_1 & x_1x_2 & x_1x_3 \\ x_2x_0 & x_2x_1 & x_2x_2 & x_2x_3 \\ x_3x_0 & x_3x_1 & x_3x_2 & x_3x_3 \end{bmatrix} = \begin{bmatrix} 1 & \theta & \theta^2 & \theta^3 \\ \theta & \theta^2 & \theta^3 & \theta^4 \\ \theta^2 & \theta^3 & \theta^4 & \theta^5 \\ \theta^3 & \theta^4 & \theta^5 & \theta^6 \end{bmatrix}.$$

If we have selected our variables appropriately, we can then express a polynomial, $\ell(\boldsymbol{x})$, of our problem in terms of a *Frobenius inner product* between a *coefficient* matrix and this *variable* matrix:

$$\ell(\boldsymbol{x}) = \sum_{i,j=1}^{n} \boldsymbol{L}_{ij}\boldsymbol{X}_{ij} = \langle \boldsymbol{L}, \boldsymbol{X} \rangle = \operatorname{tr}(\boldsymbol{L}\boldsymbol{x}\boldsymbol{x}^\top) = \operatorname{tr}(\boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{L}\boldsymbol{x}. \tag{2}$$

We can construct the coefficient matrix, $\boldsymbol{L}$, by simply matching its elements to the coefficients of $\ell(\boldsymbol{x})$.

Polynomial Example: We can construct the cost matrix, $\boldsymbol{C}$, by simply matching its elements to the coefficients of $p(\theta)$. One possible choice for this coefficient matrix is as follows:

$$\boldsymbol{C} = \begin{bmatrix} \alpha_0 & \frac{1}{2}\alpha_1 & \frac{1}{3}\alpha_2 & \frac{1}{4}\alpha_3 \\ \frac{1}{2}\alpha_1 & \frac{1}{3}\alpha_2 & \frac{1}{4}\alpha_3 & \frac{1}{3}\alpha_4 \\ \frac{1}{3}\alpha_2 & \frac{1}{4}\alpha_3 & \frac{1}{3}\alpha_4 & \frac{1}{2}\alpha_5 \\ \frac{1}{4}\alpha_3 & \frac{1}{3}\alpha_4 & \frac{1}{2}\alpha_5 & \alpha_6 \end{bmatrix}.$$

Similarly, we can represent the constraints as:

$$\boldsymbol{x}^T \boldsymbol{A}_i \boldsymbol{x} = b_i \tag{3}$$

where

$$\boldsymbol{A}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{A}_1 = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 \\ 0 & -1 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{A}_2 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix},$$

$$b_0 = 1,\ b_1 = 0,\ b_2 = 0.$$

Solving (QCQP-1) with the parameters defined above will yield the solution to our polynomial optimization in (1).

Additionally, the Appendices A.1 and A.2 provide an introduction to additional tools – namely, vectorization and homogenization – that can also be useful in this process.

2.1.1 Rotation Synchronization (RS)

We now introduce the problem of RS, and its reformulation as a QCQP. This problem is a core element of many modern SLAM backends, including the example from Figure 1. Indeed, at the heart of many SLAM backends is *pose graph optimization (PGO)* [82]. Though PGO involves solving for optimal robot *poses*, it is often the case that the translation variable is *unconstrained* and can be re-expressed as a closed-form function of the rotation variables. As a result, PGO can be solved by first solving a problem involving only the robot orientation, which is referred to as *RS*.[2] Along with PGO, certifiably correct RS has been well studied by both the robotics [82, 19, 33] and vision [39, 21] communities.

The objective of this optimization problem is to solve for an optimal set of N_R rotation matrices, $\boldsymbol{R}_i \in \mathrm{SO}(3)$ – representing robot or camera orientations – with respect to a set of relative rotation measurements, $\tilde{\boldsymbol{R}}_{ij}$. Generally, these measurements are arranged into a graph structure, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, \ldots, N_R\}$ and edges, $\{i, j\} \in \mathcal{E}$ exist only if there is a measurement, $\tilde{\boldsymbol{R}}_{ij}$, between $\boldsymbol{R}_i$ and $\boldsymbol{R}_j$. Assuming isotropic measurement noise, the maximum-likelihood orientations can be estimated by solving

$$\min_{\boldsymbol{R}_i \in \mathrm{SO}(3)} \quad \sum_{\{i,j\} \in \mathcal{E}} \sigma_{ij} \|\boldsymbol{R}_i - \tilde{\boldsymbol{R}}_{ij} \boldsymbol{R}_j\|_F^2, \tag{4}$$

where σ_{ij} denotes a *weight* assigned to measurement, $\tilde{\boldsymbol{R}}_{ij}$. The convex relaxation of this problem is arguably one of the most well-studied among the certifiable methods for

[2] Particularly in the vision community, this problem is often referred to as *rotation averaging*.

robotics and machine vision [82, 39, 21]. We immediately note that the objective function is already quadratic in the $\boldsymbol{R}_i$. It is not immediately obvious that the $\mathrm{SO}(3)$ constraints are also polynomial in the elements of $\boldsymbol{R}_i$:

$$\mathrm{SO}(3) = \{\boldsymbol{R}_i \in \mathbb{R}^{3\times 3} | \boldsymbol{R}_i \boldsymbol{R}_i^T = \boldsymbol{I},\ \det \boldsymbol{R}_i = 1\}, \tag{5}$$

where $\det$ computes the determinant of a given matrix. We note that the orthogonality constraint, $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$, is quadratic, while the determinant constraint, $\det \boldsymbol{R} = 1$,[3] is cubic in the elements of $\boldsymbol{R}$, meaning that this problem is a POP. At least in the case of rotation synchronization with *isotropic noise*, it has been shown that the determinant constraint can be omitted without any practical repercussion [82, 39].

Since the remaining constraints and cost are *quadratic* in $\boldsymbol{R}$, we are left with a QCQP; what remains is to put the problem in one of our standard forms. In Appendix A.3, we show that this problem can be rewritten in the form of (QCQP-r):

$$\begin{aligned} f^* = \min_{\boldsymbol{R}\in\mathbb{R}^{n\times d}} \quad & \mathrm{tr}(\boldsymbol{R}^T \boldsymbol{C} \boldsymbol{R}) \\ \text{s.t.} \quad & \mathrm{tr}(\boldsymbol{R}^T \boldsymbol{A}_i \boldsymbol{R}) = b_i, \quad \forall i \in [p], \end{aligned} \tag{RS-QCQP}$$

As we will see in Section 4, this particular formulation corresponds exactly to a particular manifold, which can be exploited to yield a highly efficient, globally optimal algorithm for solving the RS problem.

Remark 1 (Enforcing the Determinant Constraint)**.** *Recent works have shown that it is important to include the determinant constraints when dealing with* anisotropic noise *in rotation synchronization [70] and related problems [51, 41]. It turns out that the determinant constraints can actually be enforced by adding a "handedness" constraint that restricts each rotation to correspond to a right-handed frame. Letting $\boldsymbol{r}_i$ be the i^{th} column of $\boldsymbol{R} \in \mathrm{O}(3)$, this constraint takes the following form:*

$$\boldsymbol{r}_i \times \boldsymbol{r}_j = \boldsymbol{r}_k. \tag{6}$$

It is worth noting that this equation has both a quadratic and a linear term. As discussed in Appendix A.2, a technique known as homogenization allows us to re-express the linear terms as quadratic terms and thus include these constraints in the QCQP.

2.2 Shor's Relaxation

Once we have done the work to convert a POP to a QCQP, it is relatively straightforward to obtain a particular convex relaxation, which is often referred to as *Shor's Relaxation* [88].

[3] In general, the determinant of a matrix is a polynomial equation with degree equal to the dimension of the matrix.

First, we note that the cyclic property of the trace implies that $\operatorname{tr}(\boldsymbol{Y}^\top \boldsymbol{C}\boldsymbol{Y}) = \operatorname{tr}(\boldsymbol{C}\boldsymbol{Y}\boldsymbol{Y}^\top)$. Rewriting the product as a single matrix, $\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{Y}^\top$, we have the following property:[4]

$$\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{Y}^T,\ \boldsymbol{Y} \in \mathbb{R}^{n\times r} \iff \boldsymbol{X} \in \mathcal{S}^n_+,\ \operatorname{rank} \boldsymbol{X} = r. \tag{7}$$

Using this property, we can express (QCQP-r) only in terms of $\boldsymbol{X}$,

$$\begin{aligned} f^* = \min_{\boldsymbol{X}\in\mathbb{R}^{n\times n}} \quad & \operatorname{tr}(\boldsymbol{C}\boldsymbol{X}) \\ \text{s.t.} \quad & \operatorname{tr}(\boldsymbol{A}_i\boldsymbol{X}) = b_i, \quad \forall i \in [p], \\ & \operatorname{tr}(\boldsymbol{B}_j\boldsymbol{X}) \geq d_j, \quad \forall j \in [q], \\ & \boldsymbol{X} \in \mathcal{S}^n_+, \\ & \operatorname{rank} \boldsymbol{X} = r. \end{aligned} \tag{8}$$

Note that the objective and constraints have become linear and affine, respectively, and $\mathcal{S}^n_+$ is a convex cone. In fact, all the *nonconvexity* of the problem has been relegated to the constraint on the rank of $\boldsymbol{X}$. It follows that we can remove the constraint on rank to obtain the *primal SDP relaxation* of (8):[5]

$$\begin{aligned} p^* = \min_{\boldsymbol{X}\in\mathbb{R}^{n\times n}} \quad & \operatorname{tr}(\boldsymbol{C}\boldsymbol{X}) \\ \text{s.t.} \quad & \operatorname{tr}(\boldsymbol{A}_i\boldsymbol{X}) = b_i, \quad \forall i \in [p], \\ & \operatorname{tr}(\boldsymbol{B}_j\boldsymbol{X}) \geq d_j, \quad \forall j \in [q], \\ & \boldsymbol{X} \in \mathcal{S}^n_+. \end{aligned} \tag{SDP-Primal}$$

Owing to symmetry, the total dimension of the SDP variable, $\boldsymbol{X}$, is $\frac{n(n+1)}{2}$, whereas the original problem had nr variables. Due to this dimension increase, we often say that the SDP variable is a 'lifted form' of the original variable. This increase also constitutes one of the chief computational difficulties when applying SDPs to large-scale problems in robotics.

2.3 Lagrangian Duality

It turns out that both (QCQP-r) and (SDP-Primal) share the same *Lagrangian dual problem*.[6] Indeed, another means of deriving (SDP-Primal) is to find the dual of the dual formulation of (QCQP-r). The dual problem serves an important role when certifying optimality, which we will explore later in Section **??**. To derive the dual problem, we first

[4] This property can be easily proven by taking the singular value decomposition (SVD) of $\boldsymbol{X}$.

[5] This problem is also a relaxation to (QCQP-1): by noting that $\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x} = \operatorname{tr}(\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x})$, the preceding development is identical.

[6] For a detailed treatment of duality theory, see [69, Chapter 12] or [18, Chapter 5].

form the Lagrangian function of (QCQP-r) as follows:

$$\mathcal{L}(\boldsymbol{Y}, \boldsymbol{\lambda}) = \operatorname{tr}(\boldsymbol{Y}^T(\boldsymbol{C} + \sum_{i=1}^{p} \boldsymbol{A}_i\lambda_i - \sum_{j=1}^{q} \boldsymbol{B}_j\mu_j)\boldsymbol{Y}) - \boldsymbol{b}^T\boldsymbol{\lambda} + \boldsymbol{d}^T\boldsymbol{\mu}, \tag{9}$$

where $\boldsymbol{b} = [b_i]_i$, $\boldsymbol{d} = [d_j]_j$, and $\boldsymbol{\lambda} = [\lambda_i]_i$ and $\boldsymbol{\mu} = [\mu_j]_j$ are the Lagrange multipliers associated with the equality constraints and inequality constraints, respectively. The dual variables associated with inequalities are also constrained to be positive, $\mu_j \geq 0$.

The *Lagrangian dual function*, $g(\boldsymbol{\lambda}, \boldsymbol{\mu})$, is found by taking the infimum of the Lagrangian with respect to the (unconstrained) primal variable, $\boldsymbol{Y}$:

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) &= \inf_{\boldsymbol{Y}} \mathcal{L}(\boldsymbol{Y}, \boldsymbol{\lambda}, \boldsymbol{\mu}) && (10)\\ &= \inf_{\boldsymbol{Y}} \operatorname{tr}(\boldsymbol{H}\boldsymbol{Y}\boldsymbol{Y}^T) - \boldsymbol{b}^T\boldsymbol{\lambda} + \boldsymbol{d}^T\boldsymbol{\mu} && (11)\\ &= \begin{cases} -\boldsymbol{b}^T\boldsymbol{\lambda} + \boldsymbol{d}^T\boldsymbol{\mu} & \boldsymbol{H} \in \mathcal{S}_+^n, \\ -\infty & \text{o.w.}, \end{cases} && (12) \end{aligned}$$

where, for convenience, we have defined $\boldsymbol{H} = \boldsymbol{C} + \sum_{i=1}^{p} \boldsymbol{A}_i\lambda_i - \sum_{j=1}^{q} \boldsymbol{B}_j\mu_j$. We arrived at the expression on the right by noting that the infimum will be unbounded below when $\boldsymbol{H}$ has any negative eigenvalues (since we can always select a column of $\boldsymbol{Y}$ to be in the direction of the corresponding eigenvector). On the other hand, if $\boldsymbol{H} \in \mathcal{S}_+^n$, then the infimum of $\operatorname{tr}(\boldsymbol{H}\boldsymbol{Y}\boldsymbol{Y}^T)$ must be equal to zero, since $\boldsymbol{Y}\boldsymbol{Y}^T$ cannot have negative eigenvalues. We arrive at the dual problem by *maximizing* the dual function with respect to the dual variables:

$$\begin{aligned} d^* = \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{H}} \quad & -\boldsymbol{b}^T\boldsymbol{\lambda} + \boldsymbol{d}^T\boldsymbol{\mu} \\ \text{s.t.} \quad & \mu_j \geq 0, \ \forall j \in [q] \\ & \boldsymbol{H} = \boldsymbol{C} + \sum_{i=1}^{p} \boldsymbol{A}_i\lambda_i - \sum_{j=1}^{q} \boldsymbol{B}_j\mu_j \\ & \boldsymbol{H} \in \mathcal{S}_+^n. \end{aligned} \tag{SDP-Dual}$$

The dual matrix $\boldsymbol{H}$ plays a central role in many certifiable algorithms and is often referred to as the *certificate matrix*.

2.4 Karush-Kuhn-Tucker Conditions

The Karush-Kuhn-Tucker (KKT) conditions are a set of optimality conditions that play a central role in optimization. In general, these conditions are necessary for any *local* optimum, but for convex programs, they are often both sufficient and necessary for the *global*

optimum. The general form of these conditions is well-studied in classic optimization texts (see [69, 18]), but we provide the KKT conditions for SDP relaxation below:

$$\begin{array}{ll} \text{stationarity:} & \boldsymbol{H} = \boldsymbol{C} + \sum_{i=1}^{m} \boldsymbol{A}_i \lambda_i, \\ \text{primal feasibility:} & \operatorname{tr}(\boldsymbol{A}_i \boldsymbol{X}) = b_i, \ \forall i \in [p], \quad \operatorname{tr}(\boldsymbol{B}_j \boldsymbol{X}) \geq d_j, \ \forall j \in [q], \quad \boldsymbol{X} \in \mathcal{S}_+^n, \\ \text{dual feasility:} & \boldsymbol{H} \in \mathcal{S}_+^n, \quad \mu_j \geq 0, \ \forall j \in [q], \\ \text{complementarity:} & \operatorname{tr}(\boldsymbol{H}\boldsymbol{X}) = 0. \end{array} \tag{KKT}$$

Combining the conditions that $\boldsymbol{H}$ and $\boldsymbol{X}$ are positive semidefinite (PSD), the complementarity condition is equivalent to $\boldsymbol{H}\boldsymbol{X} = \boldsymbol{0}$.[7] This condition implies that $\boldsymbol{H}$ and $\boldsymbol{X}$ share a common set of eigenvectors, since they commute. The complementarity condition can also be expressed as

$$\boldsymbol{y} \in \ker(\boldsymbol{H}), \quad \forall \boldsymbol{y} \in \operatorname{Im}(\boldsymbol{Y}), \tag{13}$$

where $\boldsymbol{Y}$ is a factorization of the primal solution, $\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{Y}^T$. We will see in Section 4 that these conditions are of particular interest to us, since they can provide us with a means to *certify* global optimality. In particular, given a candidate solution, $\hat{\boldsymbol{Y}}$, to (QCQP-r), showing that $\hat{\boldsymbol{X}} = \hat{\boldsymbol{Y}}\hat{\boldsymbol{Y}}^T$ satisfies (KKT) is equivalent to certifying global optimality of $\hat{\boldsymbol{Y}}$.

2.5 Duality, Tightness, and Suboptimality Bounds

It is worth noting that convex programs usually exhibit the interesting property – known as *strong duality* – that their primal and dual formulations yield the same optimal value.[8] For our convex relaxation, strong duality implies that

$$-\boldsymbol{b}^T \boldsymbol{\lambda}^* + \boldsymbol{d}^T \boldsymbol{\mu}^* = \operatorname{tr}(\boldsymbol{C}\boldsymbol{X}^*) \tag{14}$$

at the optimal primal-dual solution $(\boldsymbol{X}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$.

On the other hand, *weak duality* is a key concept from optimization theory that guarantees that the optimal value of a given problem is *always lower-bounded* by the optimal value of its Lagrangian dual. This property holds regardless of whether the optimization problem is convex or nonconvex. In our context, weak duality of the nonconvex problem and strong duality of its relaxation give us the following property:

$$d^* = p^* \leq f^* \leq f(\boldsymbol{Y}), \tag{15}$$

[7] See Section 1 of [2] for an explanation.

[8] Strong duality for convex programs is guarantied when certain *regularity conditions* (also called *constraint qualifications*) hold [18]. A common regularity condition for SDPs is Slater's Condition, which holds for most problems of interest to us. In cases where it does not hold, *facial reduction* techniques can be applied to transform the problem into an equivalent one where Slater's Condition holds [36, 77].

where d^*, p^* and f^* are the optima of (SDP-Dual), (SDP-Dual), and (QCQP-r), respectively, and $\boldsymbol{Y}$ is any feasible point of (QCQP-r). A key implication of these inequalities is that if any feasible point, $\boldsymbol{Y}$, has an objective value equal to p^*, then it *must be a global minimizer of* (QCQP-r) (i.e., we can *certify global optimality of* $\boldsymbol{Y}$) since, by weak duality, there can be no other feasible point with a lower objective value. In this case, the inequality above becomes an equality ($p^* = f^*$) and we say that the convex relaxation is *tight* or *exact*. Since (QCQP-r) and (SDP-Primal) share the same dual problem, we can also say that strong duality holds *for the nonconvex problem*, (QCQP-r).

Alternatively, if the rank of a minimizer, $\boldsymbol{X}^*$, of the relaxation (SDP-Primal) is equal to r, then it can be factorized as $\boldsymbol{X}^* = \boldsymbol{Y}\boldsymbol{Y}^T$. In this case, it is easy to show that the factor, $\boldsymbol{Y}$, is a feasible point of (QCQP-r) with objective value equal to p^* and a *global minimizer*. We can therefore recover the global optimizer directly from the SDP solution. However, it is important to note that it is still possible for the relaxation to be tight even if $\mathrm{rank}(X^*) > r$.[9]

Much of the literature is concerned with finding tight relaxations or methods by which a relaxation can be made tight. However, even if a given convex relaxation is not tight, it can still provide a useful *bound on the suboptimality* of a solution to (QCQP-r): given a feasible point, $\boldsymbol{Y}$, with objective value $\hat{f}$, weak duality guarantees that

$$\hat{f} - f^* \leq \hat{f} - p^*. \tag{16}$$

2.6 Solving the SDP: A First Pass

As convex problems, SDPs can almost always be solved in polynomial time using interior point (IP) solvers [68, 1].[10] Indeed, most state-of-the-art solvers use interior-point methods to solve SDPs. Despite complexity guarantees, it is often the case that solving SDPs directly is not a viable option for large-scale robotic problems due to both the speed and memory requirements. In general, a given SDP relaxation of a robotics problem will scale as $O(n^2)$ in terms of memory complexity and between $O(n^2)$ and $O(n^3)$ in terms of time complexity [53]. Additionally, general-purpose SDP solvers have been known to struggle with the poor conditioning and low rank of SDP relaxations [103].

Despite their limitations, general-purpose solvers are still convenient tools for initially assessing whether a particular robotics problem has a tight relaxation. Once a problem has

[9] This point is subtle and has to do with the fact that the solution to a given SDP is not necessarily unique, especially for problems with low-rank solutions (see Section 2.7.5).

[10] There are some examples of convex programs that *cannot* be solved in polynomial time, including some pathological cases of SDPs. For example, copositive programming [38] and nonnegative polynomial optimization [11, Section 3.1.1] both optimize over convex cones, but solving them is NP-Hard in many cases.

been formulated as a QCQP, we can study its tightness by plugging the objective and constraint matrices directly into a general-purpose solver and assessing either of the metrics introduced in Section 2.5. Since the tightness of a given problem can vary with problem parameters [28], it is good practice to assess tightness over a representative range of parameters and determine at which point tightness breaks down. Once, a suitably tight relaxation has been found, the solution method can be tailored to exploit the specific problem structure to make it fast enough for real-time applications.

The remainder of this tutorial is divided into two main sections, which discuss the next steps for applying certifiable methods once an initial "tightness study" has been completed:

- If the relaxation is not tight, then Section 3 introduces methods that can be used to *improve tightness*.
- If the relaxation is not fast, Section 4 introduces specialized certifiable methods that can significantly improve the speed and memory requirements compared to more general solvers.

2.7 Practical considerations

2.7.1 Reformulation and Tightness

It is important to keep in mind that the conversion from POP to QCQP is often a non-unique mapping. For example, our choice of cost and constraint coefficients for the unconstrained polynomial problem in the previous sections was not unique; the following cost matrix is equally valid:

$$\boldsymbol{C} = \begin{bmatrix} \alpha_0 & \frac{1}{2}\alpha_1 & 0 & 0 \\ \frac{1}{2}\alpha_1 & \alpha_2 & \frac{1}{2}\alpha_3 & 0 \\ 0 & \frac{1}{2}\alpha_3 & \alpha_4 & \frac{1}{2}\alpha_5 \\ 0 & 0 & \frac{1}{2}\alpha_5 & \alpha_6 \end{bmatrix}. \tag{17}$$

In general, there are infinitely many ways to convert a POP to a QCQP and, importantly, these choices can have a profound impact on the tightness and sparsity of the relaxation of the POP. Choosing a formulation that leads to a tight relaxation is still an active area of research, but a common heuristic is to reduce the number of variables and constraints that need to be introduced to reformulate the problem [99, 101].

Another key example of this impact of formulation occurs in the RS problem. We mentioned in Section 1, that the determinant constraint could be included by homogenizing the problem. One way to do this involves first converting (RS-QCQP) into the form of (QCQP-1) via vectorization (see Appendix A.1). However, this vectorized form results

in a relaxation whose dimension is $\frac{nr(nr+1)}{2}$ (as opposed to $\frac{n(n+1)}{2}$ for the non-vectorized form). The additional degrees of freedom of the vectorized form may necessitate additional constraints to tighten the resulting relaxation.

2.7.2 SDP Solvers

There are several general-purpose, conic optimization toolboxes that can be used to solve the SDPs described above. CVX (and its Python-based counterpart, CVXPy) is an open-source, convex optimization parser that is fairly user-friendly and offers access to a variety of different backend optimization solvers [45, 35]. Parsers like CVX allow the user to specify their problem in a simplified syntax, which is then automatically converted to a specific canonical form based on the desired solver. There is also a range of parsers, such as SOSTOOLS [71] and GloptiPoly [50], that can automatically formulate hierarchies of SDP relaxations based on a user-specified POP. However, we must caution the reader that these parsers do not necessarily lead to an *efficient* formulation of the problem.

Since SDPs are convex, conic optimization problems, there is a wide variety of solvers that can be applied. These solvers can be broadly divided into two categories based on overall solver technique: first-order solvers and interior point solvers. First-order method solvers typically use a so-called operator splitting technique to alternately project the solution onto the affine constraints and the PSD cone. Although these solvers can deal with very large problem instances, they typically struggle to converge to high-accuracy solutions (sublinear to linear convergence). On the other hand, IP solvers use Newton's method with a modified cost and can converge quickly to high-accuracy solutions (linear to quadratic convergence). However, as mentioned above, these solvers struggle in terms of both memory and time in large-scale SDP optimization. A non-exhaustive summary of solvers that can be used to solve SDPs is provided in Table 1.

We have found that Mosek is a good all-around option in terms of speed and reliability during initial experimentation and although it is a commercial solver, it is free for academic use. In terms of open-source alternatives, Clarabel is a good choice for small to medium size programs while SCS is a well-known for scaling to large problems with moderate precision requirements.

Solver	Solution Method	Exploits Sparsity	License	Ideal Problem Scale	Precision	GPU Support	Languages
MOSEK	IP	LS only	Commercial	Small-Medium	High	No	Python, MATLAB, C, Java
SDPT3	IP	LS (limited)	GPL-2.0	Small-Medium	High	No	MATLAB
SeDuMi	IP	LS (limited)	GPL-2.0	Small-Medium	High	No	MATLAB
SCS	ADMM	CD	Apache-2.0	Large	Low-Medium	Yes	C, Python, Julia
COSMO	ADMM	CD	GPL-3.0	Large	Low-Medium	No	Julia
CSDP	IP	LS	EPL-2.0	Small-Medium	Medium-High	No	C, MATLAB
DSDP	IP	LS	EPL-1.0	Small-Medium	Medium	No	C, MATLAB
SDPNAL+	IP + AL	LS + CD	Academic (restricted)	Large	High	No	MATLAB
CDCS	ADMM	–	GPL-3.0	Large	Low-Medium	No	MATLAB
Clarabel	IP + PM	LS + CD	Apache-2.0	Small-Medium	Medium	Planned	Rust, C, Python, Julia

Table 1: Comparison of Semidefinite Programming Solvers.
Acronyms: IP = Interior-Point, AL = Augmented Lagrangian, ADMM = Alternating Direction Method of Multipliers, PM = Proximal Methods, LS = Linear Solver, CD = Cone Decomposition.

2.7.3 Measuring Tightness

Due to issues with numerical precision in practice, a relaxation is never exactly tight. We instead use continuous metrics to quantify *how tight* a given relaxation is. The most common metric is the *relative duality gap*, which (in our context) is defined as[11]

$$s^* = \frac{f^* - p^*}{|p^*|}, \tag{18}$$

where a small s^* value indicates that a relaxation is tight. Alternatively, we can consider an *eigenvalue ratio* metric that measures how close $\boldsymbol{X}^*$ is to being rank-r,

$$e^* = \frac{\sigma_r}{\sigma_{r+1}}, \tag{19}$$

where σ_i denotes the i^{th}-largest eigenvalue of $\boldsymbol{X}^*$ and a large value for e^* indicates that a relaxation is tight. In practice, we may consider a relaxation to be tight if $s^* < 1 \times 10^{-6}$ or $e^* > 1 \times 10^6$, but these values will depend on problem parameters and conditioning.

2.7.4 Rounding Solutions

In some cases, we solve (SDP-Primal) and find that the rank of the solution, $\boldsymbol{X}^*$, is higher than the desired rank, r. Again, this can occur even when the SDP solution is tight. In these situations, the relaxed solution can still be useful for obtaining a feasible point, $\boldsymbol{Y}$, for (QCQP-r) that is not necessarily optimal, but has *bounded suboptimality*. To do so, we use a two-step procedure known as *rounding*:

Low-Rank Approximation: We first form the rank-r approximation, $\hat{\boldsymbol{X}}$, of the solution matrix by truncating the $n - r$ lowest eigenvalues of the solution matrix:[12]

$$\boldsymbol{X}^* = \sum_{i=1}^{n} \sigma_i \boldsymbol{u}_i \boldsymbol{u}_i^T \xrightarrow{truncate} \hat{\boldsymbol{X}} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{u}_i^T,$$

where σ_i and $\boldsymbol{u}_i$ are the eigenvalues and corresponding eigenvectors of $\boldsymbol{X}^*$, respectively. The rank of the truncated matrix is exactly r so it can be factored as $\hat{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{U}^T$ where $\boldsymbol{U} \in \mathbb{R}^{n \times r}$.

[11] In the literature, one can find slightly different variations on the definition or naming of this metric (c.f., [105, 87, 82]), but all involve the relative difference between the nonconvex optimum and that of its relaxation. For instance, if it is known that p^* is very close to or equal to zero, a one is added to the denominator to avoid numerical issues.

[12] Note that $\hat{\boldsymbol{X}}$ is the closest rank-r, PSD matrix to $\boldsymbol{X}$ with respect to the Frobenius inner product.

Projection onto Feasible Set: If the factor $\boldsymbol{U}$ is feasible for (QCQP-r) then the rounding procedure is complete with $\boldsymbol{Y} = \boldsymbol{U}$. More commonly, we will need to find a feasible point $\boldsymbol{Y}$ by *projecting* $\boldsymbol{U}$ onto the feasible set of (QCQP-r). This projection step is not always computationally straightforward, but there have been some cases when a closed-form projection is available. For example, in the context of RS, an SVD can be used to project variables to the nearest element of $\mathrm{SO}(3)$[82]. In SDP relaxations of combinatorial problems ($\boldsymbol{x} \in \{0,1\}^n$), the low-rank approximation can be interpreted probabilistically and a near-optimal feasible point can be found via sampling or directly rounding to zero or one [42, 63].

As mentioned above, since we have an optimal value for (SDP-Primal), we can bound the suboptimality of $\boldsymbol{Y}$. In general, we define a *suboptimality gap*, to quantify how close the rounded solution is to being optimal:

$$s(\boldsymbol{Y}) = \frac{\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{C}\boldsymbol{Y}) - p^*}{p^*}. \tag{20}$$

In general, we have that $s(\boldsymbol{Y}) \geq s^*$, with equality only when $\boldsymbol{Y}$ is the global minimizer.

2.7.5 Uniqueness of the solution

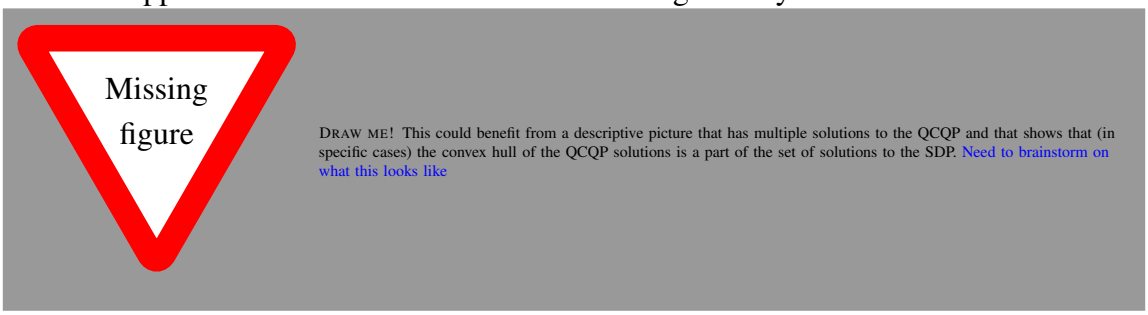
An often confusing phenomena arises when the SDP being solved has multiple primal solutions, which occurs when the SDP satisfies certain degeneracy conditions [2].[13] In general, even if an SDP has a low-rank solution (i.e., it is tight), it is not guaranteed that all possible solutions to the SDP will also have a low rank. One particular case where this occurs is when the original QCQP had multiple solutions. When the SDP relaxation is tight, each of these solutions will constitute a low-rank, globally optimal solution to the convex relaxation. It is known that any convex combination of a solution to a convex problem will also be a solution [18]. Therefore, any convex combination of these low-rank solutions will also be a solution to the SDP, and will generally be of higher rank.

The effect of obtaining a (non-unique) high-rank solution is particularly noticeable when using interior point methods to solve the SDP, since they typically provide *maximum-rank* solutions [8].[14] Intuitively, this makes sense, since these solvers generate iterates that lie in the *interior* of the SDP cone, while low-rank solutions lie at the boundary of the SDP cone [18]. Finding the lowest-rank, optimal solution to an SDP is generally NP-hard,

[13] Namely, the conditions for multiple primal solutions are *strict complementarity* and *dual degeneracy*.

[14] In general, interior point methods converge to solutions of *maximum complementarity*, which, in the context of SDPs implies that the rank of both the primal and dual solutions will be maximal [40].

but there are some good heuristic algorithms for doing so [60]. For example, the Burer-Monteiro approach discussed in Section 4.4.2 is designed to yield low-rank solutions.



DRAW ME! This could benefit from a descriptive picture that has multiple solutions to the QCQP and that shows that (in specific cases) the convex hull of the QCQP solutions is a part of the set of solutions to the SDP. Need to brainstorm on what this looks like

3 How to Tighten SDP Relaxations

As we have established, the value of SDP relaxations lies in their potential to provide a *tight* relaxation for our non-convex problems. However, the SDP relaxation is not guaranteed to be tight.

No single best way to construct a tight relaxation for a given problem exists. There are several tradeoffs to consider, and finding the ideal approach remains somewhat of an art rather than a science. A practitioner must juggle between runtime, memory requirements, and the tightness of the relaxation. In certain scenarios, an entirely different but equivalent formulation of the problem may result in a tighter SDP relaxation.[15]

This section presents various techniques to tighten SDP relaxations, which we hope can guide the practitioner in finding a good relaxation. In Section 3.1, we introduce the concept of *redundant constraints* along with different approaches to obtaining them. In Section 3.2, we discuss the moment hierarchy and how it can be used to tighten relaxations. In Section 3.3, we discuss how problem formulation and parameters of a problem can affect the tightness of a relaxation. The presented techniques are, in general, complementary, and it isn't easy to make general recommendations about which approach the practitioner should start with. We nevertheless aim to provide a comparison and general recommendations in Section 3.4.

3.1 Redundant Constraints

In this section, we introduce the definition of redundant constraints, explain their importance, and present three different approaches to finding them. Sometimes, redundant con-

[15] E.g., contrast the QCQP formulation of [73] to the equivalent QCQP used to obtain a tight relaxation in [72]. The relaxation of the latter is often tight without any redundant constraints, while the former is not.

straints are not too difficult to find simply by studying the problem structure. This is visualized with two examples in Section 3.1.2. However, this is not always possible and, in general, may require extensive experience and intuition. Thankfully, there are symbolic (Section 3.1.3) and numerical (Section 3.1.4) tools that allow us to discover these redundant constraints automatically, which we also present.

3.1.1 Definition

We use the phrase *redundant constraints* to denote constraints that are redundant for the original QCQP, i.e., they do not change its feasible set, but are non-redundant for the SDPs. In particular, they reduce the feasible set of (SDP-Primal), which can increase p^*.[16] The constraints can also be seen as enlarging the feasible set of (SDP-Dual), which increases d^*[17] The introduction of redundant constraints can also be seen as trimming away portions of the feasible set of (SDP-Primal) that can cause the solution to have a higher rank.

To ease discussion, we will occasionally use the term *primary* constraint to denote the original constraints of (POP) that shape the feasible set. If a constraint is not primary, then it is redundant. However, there is no unique assignment of which constraints are primary and which are redundant, and this labeling does not affect the underlying mathematical problem.

3.1.2 Manual Approach

We make the concept of redundant constraints more concrete by showing how to determine them manually, using the two previously introduced example problems.

Unconstrained Polynomial Optimization In our example polynomial optimization problem **??**, we alluded to the fact that solving (SDP-Primal) may not result in a rank-one solution. However, if we add the following redundant constraint:

$$\boldsymbol{A}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & -1 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}, \quad b_3 = 0, \tag{21}$$

the relaxation will always give a rank-one solution provided there is one unique global minimum. Looking at this constraint in the original QCQP, it is clear that it corresponds

[16] Since p^* is the optimal cost of a minimization, reducing its set can only worsen the optimum, thus increasing its cost.

[17] Recall that the dual problem (SDP-Dual) is a maximization, so increasing the feasible set can only improve the optimum, thus increasing its cost.

to enforcing that $\boldsymbol{x}_2^2 = \boldsymbol{x}_1\boldsymbol{x}_3$. But since the primary constraints enforce that $\boldsymbol{x}_2 = \boldsymbol{x}_1^2$ and $\boldsymbol{x}_3 = \boldsymbol{x}_2\boldsymbol{x}_1 = \boldsymbol{x}_1^3$, this constraint is implied (calling $\boldsymbol{x}_1 = \theta$, we get $\theta^2\theta^2 = \theta\theta^3$). However, this new constraint in (21) is linearly independent from the other constraints when applied to the SDP relaxation (SDP-Primal). To see this, note that it affects elements that are not touched by any of the other constraints ($\boldsymbol{X}_{23}$ and $\boldsymbol{X}_{22}$). As a result, this new constraint adds new structure to the relaxation, thereby tightening it. [18]

Rotation Synchronization Another common example of redundant constraints arises when dealing with $\mathrm{SO}(3)$ constraints, such as in the RS problem (see Section 2.1.1). For each rotation variable, $\boldsymbol{R} \in \mathrm{SO}(3)$ (here, we drop the subscripts for readability), nine primary constraints will appear in the QCQP formulation. Six of these constraints arise from enforcing $\boldsymbol{R}^\top\boldsymbol{R} = \boldsymbol{I}$ (due to symmetry of $\boldsymbol{R}^\top\boldsymbol{R}$, only the triangular upper half contributes independent constraints). Three additional constraints arise from the enforcing right-handedness: $\boldsymbol{r}_1 \times \boldsymbol{r}_2 = \boldsymbol{r}_3$, where $\boldsymbol{r}_i$ is the i-th column of $\boldsymbol{R}$ and $\times$ is the vector cross-product. Taken together, these nine constraints fully define the set of rotation matrices.

However, the orthogonality of rotation matrices can be alternatively enforced with $\boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I}$, and right-handedness can be written as $\boldsymbol{r}_2 \times \boldsymbol{r}_3 = \boldsymbol{r}_1$ or $\boldsymbol{r}_3 \times \boldsymbol{r}_1 = \boldsymbol{r}_2$. This means that we have alternative ways of algebraically expressing the set of rotations. Any of these alternative constraints can be applied as redundant constraints to tighten the SDP relaxation, as outlined in [20] (but note that only 21 out of the total 22 constraints are actually independent [37]).

3.1.3 Symbolic Approach

This section introduced a more principled approach to finding redundant constraints, sometimes more suitable than the manual approach. Formally, the problem of finding redundant constraints can be stated as finding all constraints $h_{m+i}(\boldsymbol{\theta}), i = 1, \ldots, r$ that must hold given the constraints $h_i(\boldsymbol{\theta}), i = 1 \ldots m$ in (POP).

This is a classic problem in the field of algebraic geometry and can be solved using concepts such as the Gröbner basis [22]. For the interested reader, we give a high-level introduction to this tool and its connection with redundant constraints in Section B.1. Existing open-source tools such as TS-SOS [100] provide automatic calculation of the Gröbner basis to (symbolically) discover some redundant constraints for a given problem.[19] How-

[18] Univariate polynomials are a well-studied subproblem in the POP theory and have favorable properties when it comes to their semidefinite relaxations. An overview can be found, for example, in [75, Lecture 5].

[19] Note that the Gröbner basis not only can be used to improve tightness but also reduce the problem size. This approach is described in [74], for example, and pedagogical examples can be found in the appendix of [29].

ever, it is widely recognized that such symbolic approaches have scalability issues, both in terms of computational complexity and numerical precision [46, p.29][29]. Therefore, the numerical approaches discussed in the next section may be more appropriate for the size typically encountered in robotics applications.

3.1.4 Numeric Approach

Two recent works leverage tools from numerical linear algebra to algorithmically obtain redundant constraints with improved scalability.

AUTOTIGHT and AUTOTEMPLATE [37] The first approach, from [37], finds all redundant constraints by solving a nullspace problem, a classic numerical linear algebra problem. The idea is to sample several feasible points $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_S\}$ and then find the nullspace of the matrix formed by these samples. This nullspace will contain all possible redundant constraints that are linearly independent from the primary constraints.

More formally, note that all redundant and primary constraints have the same form: $\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = 0$, with $\boldsymbol{A}$ a symmetric matrix. We can write this as $\text{vech}(\boldsymbol{x}\boldsymbol{x}^\top)^\top \boldsymbol{a}$ where vech is the half-vectorization operator, and $\boldsymbol{a} := \text{vech}(\boldsymbol{A})$. Therefore, we can find all possible constraints by finding the nullspace of the matrix

$$\begin{bmatrix} \text{vech}(\boldsymbol{x}_1\boldsymbol{x}_1^\top) \\ \vdots \\ \text{vech}(\boldsymbol{x}_S\boldsymbol{x}_S^\top) \end{bmatrix} \tag{22}$$

Under mild assumptions on the number and choice of samples, we can ensure that if the matrix in (22) is not of full column rank, the problem is ensured to cover all possible linearly independent redundant constraints. More information about this approach, particularly a more scalable algorithm, AUTOTEMPLATE, which avoids the curse of dimensionality induced by increasingly large nullspace problems as we increase problem size, can be found in [37]. This approach has been used to identify tight relaxations for many state estimation problems in robotics [52, 43, 6, 57].

Sampling Algebraic Varieties [29] The second approach was proposed by [29], and has been used in robotics, for example, to compute regions of attraction [86] or invariant funnels [34], and in computer vision for robust camera calibration [76]. Rather than deriving redundant constraints, this approach solves an SDP parametrized by feasible samples. Under some easily verifiable assumptions on the picked samples, it can be shown that using the problem parametrized by samples is equivalent to the original problem. As opposed to

AUTOTIGHT, this approach can be seen as working directly on a minimal parameterization rather than lifting the problem to high dimensions and removing superfluous degrees of freedom through redundant constraints.

3.2 Moment Hierarchy

If the relaxation is not tight even after adding all redundant constraints, we need different techniques. Maybe surprisingly, one such technique is to add higher-order degrees of $\boldsymbol{\theta}$ to the QCQP formulation. Since these are degrees that do not even appear in the cost or constraints of the original problem (POP), it may seem counterintuitive why such additional variables help with tightness. We can make sense of it by adopting the moment hierarchy perspective, which acts as a generalization of Shor's relaxation [58].[20]

In what follows, we provide a practical guide for formulating these higher-dimensional versions of (SDP-Primal) and (SDP-Dual). As we will see, although the theory of why adding higher-order terms helps may be daunting, applying the theory is relatively straightforward.

3.2.1 Adding Higher-Order Terms

As alluded to in Section 2.1, a typical choice for $\boldsymbol{x}$ in deriving the QCQP formulation is the vector of monomials up to the minimally required degree: $d_{\min} := \lceil \frac{d}{2} \rceil$, where d is the degree of the original (POP). We can construct a higher-dimensional problem with degree $d \geq d_{\min}$ by introducing, for example, all additional monomials up to degree d; we call the resulting vector $\boldsymbol{x}^d$ and its dimension m_d. We obtain new matrices $\boldsymbol{C}^d \in \mathbb{R}^{m_d}$ and $\boldsymbol{A}_i^d \in \mathbb{R}^{m_d}, i = 1, \ldots, m$ by padding their original equivalents with zeros, possibly by redistributing the terms, as discussed in the example below. Finally, additional matrices $\boldsymbol{A}_i^d, i = m+1, \ldots, m_d$ are added for each new term in $\boldsymbol{x}^d$. We may also need to complement the relaxation with additional redundant constraints. Using these newly defined matrices, we can formulate (SDP-Primal-d) and (SDP-Dual-d), the primal and dual formulations at hierarchy degree d.

We return to the simple polynomial example to visualize this process.

[20] There is a direct connection between the moment hierarchy, which we focus on here, and sums-of-squares (SOS) tools, which are more common in the control community [**parrilo˙semidefinite˙2003**]. Intuitively speaking, the SOS perspective explains why adding higher-order terms helps tighten the dual problem. We will provide the background of both perspectives in an updated version of this paper.

Unconstrained Polynomial Optimization If we use $\boldsymbol{x}^4 = \begin{bmatrix}1 & \theta & \theta^2 & \theta^3 & \theta^4\end{bmatrix}^\top$ in an attempt to increase tightness, possible choices for the matrix $\boldsymbol{C}^4$ are, for example,

$$\boldsymbol{C}^4 = \begin{bmatrix} \alpha_0 & \frac{1}{2}\alpha_1 & \frac{1}{3}\alpha_2 & \frac{1}{4}\alpha_3 & \frac{1}{5}\alpha_4 \\ \frac{1}{2}\alpha_1 & \frac{1}{3}\alpha_2 & \frac{1}{4}\alpha_3 & \frac{1}{5}\alpha_4 & \frac{1}{4}\alpha_5 \\ \frac{1}{3}\alpha_2 & \frac{1}{4}\alpha_3 & \frac{1}{5}\alpha_4 & \frac{1}{4}\alpha_5 & \frac{1}{5}\alpha_6 \\ \frac{1}{4}\alpha_3 & \frac{1}{5}\alpha_4 & \frac{1}{4}\alpha_5 & \frac{1}{3}\alpha_6 & 0 \\ \frac{1}{5}\alpha_4 & \frac{1}{4}\alpha_5 & \frac{1}{3}\alpha_6 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{C}^4 = \begin{bmatrix} \alpha_0 & \frac{1}{2}\alpha_1 & 0 & 0 & 0 \\ \frac{1}{2}\alpha_1 & \alpha_2 & \frac{1}{2}\alpha_3 & 0 & 0 \\ 0 & \frac{1}{2}\alpha_3 & \alpha_4 & \frac{1}{2}\alpha_5 & 0 \\ 0 & 0 & \frac{1}{2}\alpha_5 & \alpha_6 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{23}$$

We pad $\boldsymbol{A}_1$ to $\boldsymbol{A}_3$ with zeros, and add additional constraints:

$$\boldsymbol{A}_4^4 = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 \end{bmatrix}, \text{ redundant: } \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \cdots, \tag{24}$$

where we provide only a couple of example redundant constraints.

3.2.2 Convergence Guarantees

The power of increasing the problem size as described above lies in the fact that, under mild assumptions,[21] the suite of increasing-degree problems yields increasingly tight relaxations. More specifically, denoting by p_d^* the optimal value of (SDP-Primal-d), we have that

$$p_d^* \xrightarrow{d\to\infty} f^*. \tag{25}$$

We also have that the rank of the minimizer of (SDP-Primal-d), $\boldsymbol{X}_d^*$, approaches r as we increase the relaxation degree, although this convergence is in general slower than the convergence of the minimum.

3.3 Problem Parameters

Finally, the tightness of a problem's relaxation is intrinsically dependent on its parameters. We have already seen one instance of such a dependence: if the parameters are such that the problem admits multiple global minima, as discussed in Section 2.7.5, then we cannot hope to recover a rank-r solution. This can happen, for example, in the degenerate case where all observed landmarks in the SLAM problem are coplanar. More generally,

[21] The assumption is the Archimedianity condition which can always be satisfied when the feasible set is compact [58].

dependence of relaxation tightness on elements such as measurement noise, connectivity, or geometry, respectively, has been reported in many state estimation works [5, 82, 39, 72, 52, 6]. In [52], an explicit link between tightness and the posterior certainty, which is inversely proportional to measurement noise and measurement graph sparsity, is provided.

Generally speaking, promising avenues for improving tightness, if the application allows, can therefore be increasing measurement frequency and connectivity, denoising or outlier rejection, or improving the problem's geometry (e.g., moving landmarks closer to the camera, resolving degeneracies such as almost co-planar landmarks, using favorable trajectories). There is still a lot of work to be done to understand, from a theoretical perspective, the dependence of problem parameters on the tightness of relaxations. The remainder of this section highlights interesting advances in this direction.

3.3.1 SDP Stability

One might argue that the fact that tightness is dependent on the problem parameters makes relaxation-based approaches somewhat brittle – even if the problem was observed on example data to yield a tight relaxation, it may fail to do so on a different dataset. However, thanks to the concept of SDP stability [28], there is a certain level of generalization guarantees. SDP stability promises that, under certain assumptions, if the relaxation is tight for a given set of *nominal* parameters, then it will remain tight in a region around these parameters. This result is not only interesting on its own as it provides confidence in the generalization of tightness studies, but also has been a main driver in deriving so-called a-priori guarantees for tightness, as explained next.

3.3.2 A-Priori Tightness Guarantees

By a-priori tightness guarantees we mean statements of the form "if X holds, the semidefinite relaxation of my problem is tight". Such guarantees are relevant because they allow us to go from a *sufficient* optimality check to a *necessary* optimality check in the solve-then-certify approach. In other words, if we know a priori that the SDP relaxation of a problem is tight, we know that the failure of a solution candidate to pass the dual certificate implies that the solution is not the global minimum.

Using the SDP stability theory, one can set the nominal parameters to be noiseless measurements, and then derive the (noise) bounds for which tightness is ensured. This was first used to show that, for sufficiently small noise, the angle synchronization problem admits a tight relaxation with high probability [5]. The result was extended to a similar bound in rotation synchronization in [82]. A more explicit noise bound is proven in [39, Theorem 4.1] for the problem of rotation synchronization. The authors provide a condition on the measurement residuals after converging to a solution candidate: if the maximum

residual magnitude is smaller than a problem-dependent constant, strong duality holds. The constant depends on the graph connectivity: the higher the connectivity, the higher the admissible residuals.

As highlighted in [26], finding more general a-priori tightness guarantees that are not too conservative is an open area of research.

3.4 Practical Considerations

3.4.1 Comparison Between Different Approaches

We have seen that there are many different approaches to tightening SDP relaxations, and there is generally no single best approach. The choice of which approach(es) to use depends on the specific problem and the desired trade-off between tightness and computational cost. In what follows, we highlight some considerations.

For example, while adding redundant constraints can be a powerful way to tighten relaxations, it is important to be aware of the potential downsides:

- Adding redundant constraints breaks constraint qualifications such as linear independence constraint qualification (LICQ) and thus may prohibit using the efficient Riemannian solvers described in Section 4.
- Depending on which redundant constraints are added, sparsity may be diminished, decreasing the effectiveness of the sparsity-exploiting solvers described in Section 4.2.

Sometimes, one may choose to go up in the moment hierarchy rather than to add all redundant constraints, in an attempt to keep sparsity intact and to avoid breaking constraint qualifications. However, adding higher-order terms of course increases problem size, which has its own toll on computational cost.

One may chose to tweak the parameters to yield a tighter relaxation; but the user may not always have control over all parameters. Adding pre-processing steps to decrease noise levels complicates the algorithmic pipeline and may lead to overall less optimal results unless sufficient separation principles apply.

Finally, the practitioner may try to find a totally different formulation that admits a tight relaxation, but finding such formulations, if they exist, requires extensive domain expertise and not seldomly certain levels of intuition and creativity.

3.4.2 Sparse Moment Hierarchy

In practice, increasing the degree considered in the relaxation, as outlined in Section 3.2 may result in a large number of terms being added to the problem, which in turn results

in a big increase in the problem size. However, adding only a few additional terms may be sufficient to achieve tightness in many instances. Some symbolic tools such as TSSOS [100] will try to infer from the problem formulation which terms need to be added. In AUTOTIGHT [37], additional terms are added one by one until a desired level of tightness is attained. For some problems in the literature, it has been shown which higher-order terms are required to reach a sufficiently tight relaxation [102].

3.4.3 Recommendations for Reporting of Tightness

We end this section with a couple of recommendations for reporting the tightness of relaxations.

- Rather than reporting statements such as "the solution has rank one" or "we have trong duality", it is more constructive to report the actual numbers as defined in Section 2.7.3. Tightness is not a binary but rather a continuous property and should be treated as such.
- Because tightness depends on problem parameters, it is generally good practice to report the tightness across a range of representative parameters. Even if a relaxation is found to be tight for a certain set of parameters (for example, a certain noise level in state estimation), tightness likely deteriorates as parameters change (for example, as noise increases).

4 How to Accelerate Solvers

Up to this point, we have focused on the general foundations of semidefinite programming based certifiable perception and techniques that can be used to get to a tight SDP relaxation. What we have not yet discussed is how to solve these SDP relaxations. Fortunately, there are a number of ready-to-use solvers that can solve generic SDPs (e.g., [3, 95]). Before doing anything else, you should try one of these solvers and see if you can get solutions sufficiently fast for your application. If you're happy with the runtime and scalability that you are able to get, then you do not need to read through the rest of this section.

If you do need more performance from your solver, then this section will discuss several avenues that have been found to improve the scalability/runtime of certifiable perception algorithms. Specifically, this section will discuss specific problem structures that can often be leveraged to improve the computational performance of certifiable perception algorithms beyond that of more generic solvers. We will focus on advantageous structures that are particularly common in robotics problems.

There are three main structures that will be discussed in this section: low-rank structure, the sparsity of the problem, and how well-behaved the geometry of the constraints is.[22] We will finish the section with a discussion of various practical details that also make a major difference in the performance of your solver.

4.1 Low-Rank Solutions

In our context, the term *low-rank structure* refers to the fact that, for a given SDP, the solution $X^* \in \mathcal{S}^n_+$ has a rank that is much smaller than the dimension of the decision variable $(\operatorname{rank} X^* \ll n)$. In our case, because you have a *tight* SDP relaxation, you are guaranteed that the SDP will have a low-rank solution[23] As a result, for the purpose of this document, we will assume that we can always take advantage of this low-rank structure.

The key idea behind leveraging low-rank structure is that instead of searching for the primal solution in the high-dimensional space of $\mathcal{S}^n_+$, we can instead restrict our search to low-dimensional subspaces of $\mathcal{S}^n_+$. We can capture this low-dimensional subspace by replacing the original variable with a low-rank factorization of the form:

$$X = YY^\top, \tag{26}$$

where $Y \in \mathbb{R}^{n \times r}$ is a rank-r matrix and $r \ll n$.

This variable substitution brings substantial computational benefits, as the search space is now much smaller and any operations (e.g., matrix multiplication) will require far fewer computational operations. This idea was first explored in the seminal works of Burer and Monteiro [23, 24] and has since been explored in a number of approaches [48, 104, 98, 16, 13].

linecolor=blue,backgroundcolor=blue!25,bordercolor=blue]FIN ME: I would like to expand this discussion a bit...

4.2 Chordal Sparsity

While there are a few different notions of sparsity that can be relevant to optimization problems (see Section 4.5 for discussion on sparse linear algebra), one of the most relevant to SDP problems is that of *chordal sparsity*. Chordal sparsity captures the idea that different entries of the matrix variable X are often *sparsely* connected to each other via the constraints and cost function. This structure can be leveraged to decompose the SDP

[22] Specifically, we will study the geometry of the constraints through the lens of the linear independence constraint qualification (LICQ).

[23] Because the relaxation is typically not useful if it is not tight. See Section 3, where we describe techniques to obtain a tight relaxation.

into smaller coupled subproblems, which can lead to substantial computational improvements.[24]

We will provide a toy example displaying chordal sparsity to illustrate the concept, but readers interested in more details should refer to the excellent references by Vandenberghe and Andersen [96, Ch. 14] and Zheng, Fantuzzi, and Papachristodoulou [106]. In this example we will first demonstrate *operationally* how chordal sparsity can be exploited to reduce the size of the problem. We will then discuss the underlying *theory* that allows us to do this.

For example, consider a small 4×4 SDP of the form:

$$\min_{X \in \mathcal{S}^4_+} \langle Q, X \rangle \quad \text{subject to} \quad \langle A_i, X \rangle = b_i, \quad i = 1, \dots, m.$$

The *aggregate sparsity* of the problem refers to the union of the sparsity patterns of the cost matrix Q and the constraint matrices A_i. Formally, it is the sparsity pattern of the matrix $\text{abs}(Q) + \sum_i \text{abs}(A_i)$, where $\text{abs}(A)$ denotes the entrywise absolute value of A (i.e., $[\text{abs}(A)]jk = |Ajk|$). This captures all positions (j, k) that are nonzero in at least one of the matrices. If our problem has the following sparsity pattern (represented as an entrywise binary mask):

$$\text{nonzero pattern} \left(|Q| + \sum_i |A_i| \right) = \begin{bmatrix} \times & \times & \times & \\ \times & \times & \times & \\ \times & \times & \times & \times \\ & & \times & \times \end{bmatrix},$$

and we write the PSD matrix $X \in \mathcal{S}^4_+$ as

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{12} & x_{22} & x_{23} & x_{24} \\ x_{13} & x_{23} & x_{33} & x_{34} \\ x_{14} & x_{24} & x_{34} & x_{44} \end{bmatrix},$$

then we can see that the x_{14} and x_{24} entries of X do not affect either the cost or constraints. As a result, the only role that these entries play in the problem is to ensure that $X \succeq 0$. It turns out (for reasons we will explain) that we can ignore these variables from the problem and solve for them afterwards if needed.

To understand how we ignore x_{14} and x_{24}, it can be helpful to visualize the aggregate sparsity of the problem as a graph, where each node represents a variable and an edge

[24] The simplest way to understand the computational benefits of chordal sparsity is that it often allows us to substantially reduce the size of the PSD constraints, which cost $\mathcal{O}(n^3)$ to enforce using typical interior-point solvers [106].

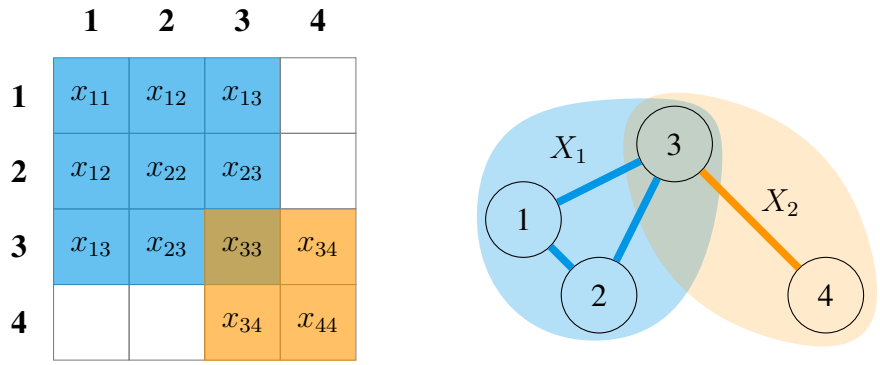


Figure 3: The aggregate sparsity of the SDP problem can be visualized as a graph, where each node represents a variable and an edge between two nodes indicates that the corresponding variables are connected by the cost or constraints. In this example, we have two cliques (i.e., fully connected subgraphs) which we have labeled X_1 and X_2. Because this graph is chordal, we can decompose the positive semidefinite constraint $X \succeq 0$ into two smaller PSD constraints, one for each clique ($X_1 \succeq 0$ and $X_2 \succeq 0$).

between two nodes indicates that the corresponding variables are connected by the cost or constraints. As we depict in Fig. 3, we can view the aggregate sparsity of our toy problem as a graph with four nodes and four edges.

We can describe this graph as having two 'cliques' (i.e., fully connected subgraphs), C_1 and C_2. We will decompose the positive semidefinite constraint $X \succeq 0$ into two smaller PSD constraints, one for each clique.

$$X_1 = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{12} & x_{22} & x_{23} \\ x_{13} & x_{23} & x_{33} \end{bmatrix} \succeq 0 \quad \text{and} \quad X_2 = \begin{bmatrix} x_{33} & x_{34} \\ x_{34} & x_{44} \end{bmatrix} \succeq 0.$$

With these variables, we can instead solve the following problem:

$$\begin{aligned} \min_{X,\, X_1,\, X_2} \quad & \langle Q, X \rangle \\ \text{subject to} \quad & \langle A_i, X \rangle = b_i, \quad i = 1, \ldots, m \\ & X_1 = X[C_1] \ \text{ and } \ X_2 = X[C_2], \\ & X_1 \succeq 0 \ \text{ and } \ X_2 \succeq 0, \end{aligned}$$

where $X_1 = X[C_1]$ and $X_2 = X[C_2]$ are the principal submatrices of X corresponding to

the cliques C_1 and C_2 of the graph.[25,26]

In this problem, the 4×4 PSD constraint has been replaced with two smaller PSD constraints (3×3 and 2×2). The cost and linear constraints remain unchanged. While in this toy example, the size reductions are not substantial, we can see that the size of PSD constraints is reduced from the original variable size to the size of the cliques. For many real-world problems, the size of the cliques is substantially smaller than the size of the original problem [106].

We have now seen an operational example of how chordal sparsity can be exploited, but we have not yet discussed the underlying theory. More precisely, we have not yet explained why we can ignore the x_{14} and x_{24} entries of X. We will now discuss the theory that allows us to do this; we attempt to keep this discussion as high-level as possible with the aim of providing a basic understanding of why this decomposition is possible.

As one would expect from the title of this subsection, we were able to perform this decomposition because our problem exhibited *chordally sparse* graph structure.

Definition 2 (Chordal Graph)**.** *A graph is* chordal *if every cycle of length greater than three has a chord (i.e., an edge that is not part of the cycle but connects two nodes in the cycle). Chordal graphs are also sometimes referred to as triangulated, as every loop in the graph is the collection of one or more triangles (collections of three nodes that are all connected to each other).*

We want to briefly emphasize that even for problems that do not originally exhibit chordal structure, they can be made chordal by inserting edges. Effectively, this means that the techniques described here can be applied to any problem. However, the important structure to take advantage of is not just that the graph is chordal, but that it is *chordally sparse* (i.e., that the chordal graph can be decomposed into cliques that are smaller than the original problem size). This sparsity is what lends computational benefits.

This chordal structure is important because of a theorem first proved by Grone et al. [47] (see Theorem 3 below). This theorem provides us a path to ensuring positive semidefiniteness for a full matrix X while only enforcing positive semidefiniteness for the principal submatrices corresponding to cliques of the graph.[27]

[25] These principal submatrices of X can be algebraically obtained by selection matrices, binary matrices which index the rows and columns of X corresponding to the cliques (see e.g., [106]).

[26] We note that there are many different ways to decompose the problem, and even often several different formulations for each decomposition. We have chosen one here to illustrate the concept, but refer to [106] for a more complete discussion.

[27] A principal submatrix is a matrix obtained by deleting rows and columns from an original matrix. For example, if you have a 3×3 matrix A, then the principal submatrix corresponding to the indices $\{1, 3\}$ would be the 2×2 matrix $A_{\{1,3\}} = \begin{bmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{bmatrix}$ obtained by deleting the second row and column of A.

Theorem 3 (Grone's Theorem [47])**.** *Let G be a chordal graph with n nodes. Then, for any symmetric matrix $X \in \mathrm{Sym}^n$ with entries specified according to the aggregate sparsity pattern given by G, the following statements are equivalent:*

1. *For any clique C of G, the corresponding principal submatrix X_C is PSD.*
2. *X has a PSD completion.*

Definition 4 (PSD Completion)**.** *Let $X \in \mathrm{Sym}^n$ be a symmetric matrix with some entries specified and others unspecified, according to a sparsity pattern defined by a graph G on n nodes. A* positive semidefinite (PSD) completion *of X is a matrix $\tilde{X} \in \mathrm{Sym}^n$ such that:*

1. *$\tilde{X}_{ij} = X_{ij}$ for all $(i,j) \in E(G)$, and*
2. *$\tilde{X} \succeq 0$ (i.e., $\tilde{X}$ is positive semidefinite).*

Effectively, what this is saying is that if we have a chordal graph G that corresponds to a matrix X, then as long as the principal submatrices induced by the cliques of G are PSD, we can find a way to fill in the remaining entries of X such that the entire matrix X is PSD.

Let's tie this back to how this helps in solving SDPs. This result means that if we have chordal structure, we can ignore the PSD constraint on the entire matrix $X \succeq 0$ and instead only apply PSD constraints to the principal submatrices corresponding to the cliques of the graph $\{X_k \succeq 0 \mid k \in \text{cliques of } G\}$.

This additionally implies that we can then ignore the entries of X that are not captured in the chordal graph used to perform the decomposition (as we know that these entries do not appear in the aggregate sparsity pattern and therefore these entries do not affect the cost or constraints). In fact, standard techniques that take advantage of chordal structure will ignore these entries entirely. In applications where the full matrix X is not needed, no more work needs to be done. However, when the full matrix X is needed, we need to find values of these previously ignored entries that provide a PSD completion of the matrix X. This is because PSD-ness of the principal submatrices only guarantees that X has a PSD completion; it is possible for X to have PSD principal submatrices but not be PSD itself. Thankfully, finding a PSD completion is a well-studied problem and can be done relatively efficiently (e.g., [54]).

4.3 The Geometry of the Constraints (the LICQ)

When the constraints of a problem are particularly well-behaved, we can obtain a number of computational benefits. More specifically, the condition we are interested in is the

linear independence constraint qualification (LICQ). When we know that the LICQ is satisfied, we can use specialized algorithms that lend major computational speedups for both performing optimization for the primal SDP as well as certifying the global optimality of the solution.

The LICQ is a condition that ensures that the gradients of the active constraints are linearly independent. If we consider constraints of the form

$$h_i(x) = x^T A_i x = b_i \quad \text{and} \quad g_i(x) = x^T B_i x \leq c_i,$$

then the LICQ is satisfied at a point x if the following set of gradients of all active constraints is linearly independent:

$$\{\nabla h_i(x) = 2A_i x \mid i = 1, \ldots, m\} \cup \{\nabla g_i(x) = 2B_i x \mid i = 1, \ldots, p \text{ and } g_i(x) \text{ is active}\}.$$

Most algorithms only require that the LICQ is satisfied at the iterates of the optimizer or at the optimal solution. However, it is typically most practical to show that the LICQ is satisfied at all feasible points.[28] In all practical instances we have found in robotics, the LICQ has been an all-or-nothing condition: either it is satisfied at all feasible points, or it is not satisfied at any feasible point. This all-or-nothing property does not have to be the case, but tends to be true in practice because typically the reason the LICQ is not satisfied is that we have overconstrained the problem (i.e., we have so many constraints that they can't form a linearly independent set).

The primary importance of the LICQ is that it guarantees that the Karush-Kuhn-Tucker (KKT) multipliers of a problem are unique [97] at the given point. Uniqueness of the KKT multipliers in turn implies that the KKT conditions (which characterize local optimality) are well-posed. With the LICQ satisfied, we can obtain the (unique) KKT multipliers by simply solving a linear system of equations.

We can show this by inspecting the KKT conditions for a general constrained optimization problem:

By looking at the stationarity condition (2.4), we can see that at a given point x, we are at a stationary point if the gradients of the active constraints can cancel out the gradient of the cost function. In fact, it is through the stationarity condition that we can easily connect the KKT multipliers to a linear system of equations.

Specifically, we can equivalently write the stationarity condition as a linear system of equations:

$$\begin{bmatrix} \nabla H(x) & \nabla G(x) \end{bmatrix} \begin{bmatrix} \lambda \\ \nu \end{bmatrix} = -\nabla f(x), \tag{27}$$

[28] This is because it is difficult to guarantee that the iterates of an optimizer will magically satisfy the LICQ and not wander to points where it is not satisfied. However, if all feasible points satisfy the LICQ, then it immediately follows that all iterates will satisfy the LICQ.

Condition	**Expression**
Stationarity	$\nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x) + \sum_{i=1}^{p} \nu_i \nabla g_i(x) = 0$
Primal Feasibility	$h_i(x) = 0, \quad i = 1, \ldots, m$
	$g_i(x) \leq 0, \quad i = 1, \ldots, p$
Complementary Slackness	$\nu_i g_i(x) = 0, \quad \nu_i \geq 0, \quad i = 1, \ldots, p$

Table 2: Karush–Kuhn–Tucker (KKT) optimality conditions.

where we have defined

$$\nabla H(x) = \begin{bmatrix} \nabla h_1(x) & \cdots & \nabla h_m(x) \end{bmatrix}, \tag{28}$$

$$\nabla G(x) = \begin{bmatrix} \nabla g_1(x) & \cdots & \nabla g_p(x) \end{bmatrix}, \tag{29}$$

$$\lambda = \begin{bmatrix} \lambda_1 & \cdots & \lambda_m \end{bmatrix}, \tag{30}$$

$$\nu = \begin{bmatrix} \nu_1 & \cdots & \nu_p \end{bmatrix}. \tag{31}$$

Due to complementary slackness (KKT), the KKT variable related to an inactive inequality constraint must be zero (i.e., $\nu_i = 0$ if $g_i(x) < 0$). This means that we can simplify the linear system of (27) to ignore the gradients from inactive constraints:

$$\begin{bmatrix} \nabla H(x) & \nabla \tilde{G}(x) \end{bmatrix} \begin{bmatrix} \lambda \\ \tilde{\nu} \end{bmatrix} = -\nabla f(x), \tag{32}$$

where $\tilde{G}(x)$ is the column-wise concatenation of gradients of *active* inequality constraints and $\tilde{\nu}$ is the vector of corresponding KKT multipliers.

From this perspective, we can see that a point is a stationary point if and only if we can find a set of KKT multipliers $(\lambda, \tilde{\nu})$ that satisfy the system of equations. Furthermore, there is a unique solution to this system if and only if the matrix $\begin{bmatrix} \nabla H(x) & \nabla \tilde{G}(x) \end{bmatrix}$ has full column rank. This is precisely an algebraic description of the LICQ.

From this, it is apparent that if the LICQ is satisfied, then we can obtain the *unique* λ and ν by solving a linear system of equations. This is important for two distinct advantages it allows: (i) extremely efficient certification of the solution and (ii) more efficient optimization algorithms to be used. The LICQ is a key enabler for what are currently the most computationally efficient methods for certifiably correct optimization: attaching a certifier to a local solver (Section 4.4.1), the Burer-Monteiro method (Section 4.4.2), and the Riemannian Staircase (Section 4.4.3).

Importance of the LICQ (Certification) First, the LICQ allows for efficient certification of QCQP formulations by computing the KKT variables and then constructing what is often referred to as the *certificate matrix*:

$$S_x = Q + \sum_{i=1}^{m} \lambda_i A_i + \sum_{i=1}^{p} \nu_i B_i, \tag{33}$$

where the λ_i and ν_i are the KKT multipliers corresponding to the active constraints at the point x. At a given point x, if the certificate matrix S_x is positive semidefinite, then we know that x is globally optimal for that given problem [81]. [29]

This means that, if the LICQ is satisfied, we can perform efficient certification by: (i) simply solving a linear system of equations ((27)) to obtain the KKT multipliers, (ii) constructing the certificate matrix S_x using (33), and (iii) checking if $S_x \succeq 0$ (which can be efficiently done by attempting to compute the Cholesky factorization of S_x) [80].

This method of certification often allows us to effectively solve SDPs by solving low-dimensional non-convex QCQPs and then attempting certification to check if the obtained QCQP (local) solution corresponds to a global solution of the SDP. This is highly valuable because it eliminates the need to solve an SDP directly in its standard (high-dimensional) form.

Importance of the LICQ (Optimization) Beyond providing an efficient certification scheme, the LICQ also allows for the use of optimization routines that provide substantial computational benefits. This is perhaps unsurprising when we consider the LICQ to be a statement on how well-behaved the constraints of the problem are. When the LICQ is satisfied, second-order optimization methods (e.g., Newton's method) can be used to solve the constrained optimization problem. This follows from the fact that many second-order methods require the Hessian of the Lagrangian to be well-defined (i.e., unique).

The Lagrangian is defined as:

$$\mathcal{L}(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{i=1}^{p} \nu_i g_i(x). \tag{34}$$

The Hessian of the Lagrangian is then given by:

$$\nabla^2 \mathcal{L}(x, \lambda, \nu) = \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 h_i(x) + \sum_{i=1}^{p} \nu_i \nabla^2 g_i(x), \tag{35}$$

[29] This property stems from the fact that obtaining a positive semidefinite certificate matrix ($S_x \succeq 0$) tells us that x represents a low-rank factorization of the optimal solution to the SDP relaxation [81]. In other words, x gives us a solution ($Z^* = xx^\top$) to the SDP relaxation of our QCQP if $S_x \succeq 0$.

which, for the QCQPs we are interested in, simplifies to

$$\nabla^2 \mathcal{L}(x, \lambda, \nu) = Q + \sum_{i=1}^{m} \lambda_i A_i + \sum_{i=1}^{p} \nu_i B_i. \tag{36}$$

As a result, we can see that the Hessian of the Lagrangian is simply the certificate matrix S_x defined in (33).

As we know that the LICQ is necessary to guarantee that the certificate matrix S_x is uniquely defined, we thus have that the LICQ is also necessary for the Hessian of the Lagrangian to be well-defined. This means that the LICQ is a natural prerequisite for the use of second-order optimization methods (which typically exhibit substantially faster convergence than first-order methods).

As an additional note, the LICQ also plays a major role in Riemannian optimization (which is a core component of the Riemannian Staircase method discussed in Section 4.4.3). Specifically, if we know that the LICQ is satisfied for all feasible points, then we know that the feasible set is a smooth manifold.[30]

is a requirement for the use of Riemannian optimization (as discussed in Section 4.4.3). This is based on the fact that the search space of the constrained optimization problem is a smooth manifold if and only if the LICQ is satisfied at all feasible points [14, Ch. 7].[31]

4.4 Common Techniques for Solving

We briefly discuss different approaches for solving SDPs based on the structures discussed above. We begin by describing techniques that require the LICQ to be satisfied: pairing a local solver with a certifier, the Burer-Monteiro method, and the Riemannian Staircase. These LICQ-based techniques are typically the most computationally efficient, and are recommended when possible.

[30] Also, for embedded submanifolds with dimension k less than the dimension of the ambient space n, we also know that at all points x on the manifold, in a neighborhood U around the point there *must be* a function $h : U \to \mathbb{R}^k$ that (locally) parameterizes the manifold (i.e., $h^{-1}(0)$ locally describes the manifold) and which has a full rank differential at that point ($\operatorname{rank} Dh(x) = k$). While this is a fairly specific set of conditions, this captures nearly all manifolds of interest in robotics (e.g., $SO(d)$). In fact, this function h is known as a defining function of the manifold [14, Def. 3.10]. The importance of this information is twofold: (i) we now immediately know that the LICQ is satisfied when our constraints are the defining functions of embedded submanifolds, and (ii) it highlights the close connections between the LICQ and the smooth structure of constraints.

[31] As we will discuss in Section 4.4.3, the LICQ being satisfied is not enough for the practical success of Riemannian optimization. Additional information on the underlying manifold (specifically, the retraction operators) is needed to have efficient numerical implementations [15, Ch. 3]

However, oftentimes our problems do not satisfy the LICQ (as in the case of introducing redundant constraints). In these instances, we can still leverage sparsity and low-rank structure to improve the computational performance of our solver; we discuss current approaches for exploiting these structures.

4.4.1 Pairing a Local Solver with a Certifier

4.4.2 The Burer-Monteiro Method

4.4.3 The Riemannian Staircase

The Riemannian Staircase is a special instance of the Burer-Monteiro method which requires additional structure on the problem's constraints. In particularly fortunate circumstances, you may find that not only does your problem satisfy the LICQ, but that the constraints of your QCQP can be expressed as manifolds. In other words, your problem can be posed as a Riemannian optimization problem. In fact, to show that the LICQ is satisfied for all feasible points, it is sufficient to demonstrate that the constraints are manifolds [14, Ch. 7].

In these specific instances, you can leverage what has been referred to as the *Riemannian Staircase* [13]. To date, this has been the most computationally efficient approach to solving SDPs in robotics applications. At this point, solving the SDP can be reduced to solving a series of unconstrained Riemannian optimization problems, something we are currently able to with substantial computational efficiency and reliability.

4.4.4 Exploiting Chordal Sparsity

4.4.5 Low-Rank Exploiting SDP Solvers

While the 'classic' approach to exploit low-rank structure is the Burer-Monteiro method, we have established that efficiently solving the BM formulation of the problem requires the LICQ (without the LICQ the KKT conditions become degenerate and second-order methods begin to fail). However, there is a growing number of approaches that attempt to solve the SDP

4.5 Practical Considerations

There are a number of practical considerations that need less theoretical background to understand, yet can make a major difference in the performance and reliability of your solver.

Sparse Linear Algebra As with many numerical algorithms, in the right circumstance, substantial performance improvements can be obtained by using sparse linear algebra operations. Most pre-existing solvers will do this for you, but may require that you provide the problem in a specific format (e.g., matrices are stored in a sparse format). Of course, a prerequisite for this is that the problem is actually sparse (i.e., the matrices that represent the constraints or objective consist of mostly zeros). In addition to improved computational performance, sparse linear algebra can also reduce the memory footprint of your problem and aid with numerical stability. The value of sparse linear algebra often arises above a certain problem size (e.g., $n > 100$), when the runtime improvements begin to outweigh the overhead of maintaining a sparse representation.

Preconditioning and Acceleration Methods Depending on the solver you are using, there are often well-known techniques that can be used to improve the convergence of the solver. For example, many second-order methods (e.g., Newton's method) can be substantially accelerated with a good preconditioner.[32] Oftentimes for problems of the form **??**, a good preconditioner is the inverse of the regularized data matrix $(Q + \lambda I)^{-1}$. Importantly, instead of directly computing this preconditioner, one should cache the Cholesky factorization $LL^\top = (Q + \lambda I)$ and apply the preconditioner via forward and backward substitution.

Similarly, many first-order methods (e.g., gradient descent) can be accelerated with what are known as acceleration methods [31]. These methods are typically based on momentum terms and can be used to accelerate the convergence of the solver. Many acceleration methods are intuitively motivated by the idea of 'averaging' the iterates of the solver to reduce the variance of the iterates and thus improve convergence. This is similar to the idea of a ball rolling down a valley, where the momentum of the ball allows for it to have a smoother trajectory to the bottom that is less affected by the roughness of local topography. Acceleration methods can be used in conjunction with preconditioning to further improve the convergence of the solver.

Computing Minimum Eigenvalues Many algorithms necessitate computing the minimum eigenvalue of a real, symmetric matrix $S \in \mathrm{Sym}^n$. In our case, this becomes useful in what is known as the saddle-escape step in Burer-Monteiro methods (including the Riemannian Staircase). It is useful in general to know that specialized iterative methods exist for computing extremal eigenvalues of symmetric matrices [61, 83]. In the specific case of the matrix that arises in the saddle-escape step (often referred to as the certificate matrix)

[32] Preconditioning is a technique that can be used to improve the convergence of iterative solvers by effectively reshaping the loss landscape of the problem to make it easier for the solver to find the solution. Preconditioners try to 'even out' the loss landscape, making it more isotropic (evenly shaped).

there is additional structure that can be leveraged [80]. Namely, an optimal preconditioner can be quickly computed for the matrix and combined with the locally optimal block preconditioned conjugate gradient (LOBPCG) [56] method to quickly compute the minimum eigenvalue.

5 Conclusion and Future Directions

TBD

Appendices

A Additional Notes for Primer

A.1 Trace and Vectorization

The trace operator appears frequently in QCQP reformulations due to its relationship to the Frobenius inner product of two matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times n}$:

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_{i,j=1}^{n} \boldsymbol{A}_{ij}\boldsymbol{B}_{ij} = \operatorname{tr}(\boldsymbol{A}^T\boldsymbol{B}) \tag{37}$$

The trace operator also has a few properties that are important to keep in mind when manipulating QCQPs, where $a,\ b \in \mathbb{R}$:

$$\begin{aligned}
&\operatorname{tr}(\boldsymbol{A}\boldsymbol{B}) = \operatorname{tr}(\boldsymbol{B}\boldsymbol{A}) && \text{(cyclic property)},\\
&\operatorname{tr}(\boldsymbol{A}) = \operatorname{tr}(\boldsymbol{A}^T) && \text{(transpose invariance)},\\
&\operatorname{tr}(a\boldsymbol{A} + b\boldsymbol{B}) = a\operatorname{tr}(\boldsymbol{A}) + b\operatorname{tr}(\boldsymbol{B}) && \text{(linearity)}.
\end{aligned}$$

It is always possible to reformulate (QCQP-r) as (QCQP-1) by stacking the columns of $\boldsymbol{Y} \in \mathbb{R}^{n\times r}$ into a vector, $\boldsymbol{x} \in \mathbb{R}^{nr}$. More formally, we define the *vectorization operator*, $\operatorname{vec}(\cdot)$, such that:

$$\boldsymbol{x} = \operatorname{vec}(\boldsymbol{Y}) = \operatorname{vec}(\begin{bmatrix}\boldsymbol{y}_1 & \dots & \boldsymbol{y}_r\end{bmatrix}) = \begin{bmatrix}\boldsymbol{y}_1^T & \dots & \boldsymbol{y}_r^T\end{bmatrix}^T. \tag{38}$$

The following identities from [64] greatly facilitate reformulating a POP into a QCQP:

$$\operatorname{tr}(\boldsymbol{A}^T\boldsymbol{B}) = \operatorname{vec}(\boldsymbol{A})^T\operatorname{vec}(\boldsymbol{B}), \quad \operatorname{vec}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}) = (\boldsymbol{C}^T \otimes \boldsymbol{A})\operatorname{vec}(\boldsymbol{B}), \tag{39}$$

where $\otimes$ represents the Kronecker product. For example, the objective may be reformulated as follows:

$$\begin{aligned}\mathrm{tr}(\boldsymbol{Y}^T\boldsymbol{C}\boldsymbol{Y}) =& \mathrm{vec}(\boldsymbol{Y})^T\mathrm{vec}(\boldsymbol{C}\boldsymbol{Y}) \\ =& \mathrm{vec}(\boldsymbol{Y})^T\mathrm{vec}(\boldsymbol{C}\boldsymbol{Y}\boldsymbol{I}) \\ =& \mathrm{vec}(\boldsymbol{Y})^T(\boldsymbol{I}^T \otimes \boldsymbol{C})\mathrm{vec}(\boldsymbol{Y}) \\ =& \boldsymbol{x}^T(\boldsymbol{I} \otimes \boldsymbol{C})\boldsymbol{x}\end{aligned}$$

The constraints can be similarly reformulated by replacing $\boldsymbol{C}$ with $\boldsymbol{A}_i$.

A.2 Homogenization

In Section 2, our main QCQP formulations involve *explicit, quadratic forms*. However, we may often want our objective or constraints to take on the following form, which includes linear or even constant terms:

$$\boldsymbol{y}^\top \boldsymbol{F}\boldsymbol{y} + \boldsymbol{f}^\top \boldsymbol{y} + f,$$

where $\boldsymbol{F} \in \mathrm{Sym}(n)$, $\boldsymbol{f} \in \mathbb{R}^n$, $f \in \mathbb{R}$. In this case, a technique known as *homogenization* can be applied, which adds a *homogenization variable*, h, to the optimization. This variable is then constrained to be equal to one, $h = 1$, allowing us to rewrite the expression above using a quadratic form:

$$\boldsymbol{y}^\top \boldsymbol{F}\boldsymbol{y} + h\boldsymbol{f}^\top \boldsymbol{y} + h^2 f = \begin{bmatrix}\boldsymbol{y} \\ h\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{F} & \boldsymbol{f} \\ \boldsymbol{f}^\top & f\end{bmatrix} \begin{bmatrix}\boldsymbol{y} \\ h\end{bmatrix}.$$

In practice, our new constraint also takes on a quadratic form:

$$\begin{bmatrix}\boldsymbol{y} \\ h\end{bmatrix}^\top \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0}^\top & 1\end{bmatrix} \begin{bmatrix}\boldsymbol{y} \\ h\end{bmatrix} = 1. \tag{40}$$

Remark 5. *The astute reader may note that this constraint enforces $h^2 = 1$, which has two solutions: $h = 1$ and $h = -1$. In practice, whenever $h = -1$, we simply flip the sign of the solution, with no effect on the objective or constraints (since they are quadratic).*

Remark 6. *We have shown how homogenization can be performed when the optimization variable is a vector (i.e. $r = 1$). It is less obvious how homogenization can be achieved when $r > 1$. One way to do this is to first re-express the problem as* (QCQP-1) *and then homogenize.*

A.3 Rotation Synchronization Reformulation

We first massage the cost function into a more convenient form. Consider one element of the sum in the objective of (4):

$$\begin{aligned}\sigma_{ij}\|\boldsymbol{R}_i - \tilde{\boldsymbol{R}}_{ij}\boldsymbol{R}_j\|_F^2 &= \sigma_{ij}\operatorname{tr}((\boldsymbol{R}_i - \tilde{\boldsymbol{R}}_{ij}\boldsymbol{R}_j)^T(\boldsymbol{R}_i - \tilde{\boldsymbol{R}}_{ij}\boldsymbol{R}_j)) \\ &= \sigma_{ij}\operatorname{tr}(\boldsymbol{R}_i^T\boldsymbol{R}_i - \boldsymbol{R}_i^T\tilde{\boldsymbol{R}}_{ij}\boldsymbol{R}_j - \boldsymbol{R}_j^T\tilde{\boldsymbol{R}}_{ij}^T\boldsymbol{R}_i + \boldsymbol{R}_j^T\boldsymbol{R}_j) \\ &= \sigma_{ij}2\operatorname{tr}(\boldsymbol{I}) - \sigma_{ij}2\operatorname{tr}(\boldsymbol{R}_i^T\tilde{\boldsymbol{R}}_{ij}\boldsymbol{R}_j),\end{aligned}$$

where we have exploited the cyclic and transpose invariant properties of the trace operator. The first term can be ignored in the optimization since it is constant. Concatenating the rotation variables into one large matrix,

$$\boldsymbol{R}^T = \begin{bmatrix}\boldsymbol{R}_1^T & \dots & \boldsymbol{R}_{N_R}^T\end{bmatrix}, \tag{41}$$

we can write the cost function as follows:

$$\operatorname{tr}(\boldsymbol{R}^T\boldsymbol{C}\boldsymbol{R}), \quad \boldsymbol{C} = \begin{cases}\sigma_{ij}\tilde{\boldsymbol{R}}_{ij}, & \{i,j\} \in \mathcal{E}, \\ \boldsymbol{0}, & \{i,j\} \notin \mathcal{E}.\end{cases} \tag{42}$$

The orthogonality constraints can be enforced element-wise as follows:

$$(\boldsymbol{R}_i\boldsymbol{R}_i^T)_{kl} = \begin{cases}1 & k = l \\ 0 & k \neq l\end{cases} \tag{43}$$

To do so, we introduce symmetric *selection matrices*, $\boldsymbol{E}_{kl}$, which are defined such that:

$$\boldsymbol{X}_{kl} = \operatorname{tr}(\boldsymbol{E}_{kl}\boldsymbol{X}) = \operatorname{tr}(\boldsymbol{R}_i^T\boldsymbol{E}_{kl}\boldsymbol{R}_i).$$

Owing to the symmetry of $\boldsymbol{R}_i\boldsymbol{R}_i^T$, orthogonality can be enforced via six constraints for each rotation matrix. Finally, the selection matrices can be embedded into a larger constraint matrix $\boldsymbol{A}_i$ such that the orthogonality constraints can be represented in our standard form:

$$\operatorname{tr}(\boldsymbol{R}^T\boldsymbol{A}_i\boldsymbol{R}) = b_i.$$

B Additional Notes for Tightening Relaxations

B.1 Algebraic Geometry and Redundant Constraints

In this section, we aim to give an accessible introduction to concepts from algebraic geometry that are useful to understand the concept of redundant constraints and their automatic

discovery. We try to keep things as high-level as possible, and we refer the reader to [30] for a deeper, but accessible, introduction to algebraic geometry.

For simplicity, we first consider problems (POP) without inequality constraints. In this case, the feasible set given by the equality constraints defines a so-called *algebraic variety*. We denote the algebraic variety by Θ and we have:

$$\Theta := \Theta(h_1, \ldots, h_m) = \{\boldsymbol{\theta} \mid h_i(\boldsymbol{\theta}) = 0\}\,. \tag{44}$$

In this context, the polynomials h_i are also called generators of the variety. In the sequel, we will often drop the arguments of polynomials.

In order to tighten the relaxation, we are looking for polynomials other than h_i that also evaluate to zero on the elements of Θ. Such polynomials are exactly the *redundant constraints*, subject of **??**.

One may be tempted to simply consider all polynomials of the form $p = \sum_i^m q_i h_i$ (possibly up to a given degree) for arbitrarily chosen polynomials q_i, to tighten the semidefinite relaxation. In what follows we explain why such an approach is not sufficient, and why we need to delve deeper into the world of algebraic geometry instead.

We use the notation $\langle h_1, \ldots, h_m \rangle$ to describe the set $\{\sum_i^m q_i h_i\}$.[33] Clearly, each element $p_i \in \langle h_1, \ldots, h_m \rangle$ (other than h_i) can be used to formulate a redundant constraint, because $h_i(\boldsymbol{\theta}) = 0$ implies that $p_i(\boldsymbol{\theta}) = 0$. However, not every polynomial that evaluates to zero for elements from Θ can be written as $\sum_i^m q_i h_i$. In what follows, we call the set of polynomials that evaluate to zero for elements from Θ the *vanishing ideal* $\mathrm{I}(\Theta)$.

Consider this toy example as an illustration: $h_1(\boldsymbol{\theta}) = \theta_2(\theta_1 - 1) - 1 = 0$, $h_2(\boldsymbol{\theta}) = \theta_3(\theta_1 - 2) - 1 = 0$. Multiplying h_1 and h_2 by any polynomial leads to valid redundant constraints, with degrees larger than two. We can however construct the following redundant constraint, which is a quadratic constraint and thus clearly not in $\langle h_1, \ldots, h_m \rangle$: $p(\boldsymbol{\theta}) = \theta_2 - \theta_3 + \theta_2\theta_3 = 0$, which is not of the form $\sum_i^m q_i h_i$ but is also implied by the h_i.

If simply considering $\sum_i q_i h_i$ does not work, how can we find the redundant constraints instead? The observation from the example has the following interpretation: the polynomials h_1, h_2 are not a sufficient basis to describe $\mathrm{I}(\Theta)$: we also need the basis vector h_2. Only then is each element of $\mathrm{I}(\Theta)$ an element of $\langle h_1, h_2, h_3 \rangle$, i.e., can be written as $\sum_i q_i h_i$ for some q_i.

The three polynomials h_1, h_2, h_3 correspond to a Gröbner basis of the vanishing ideal $\mathrm{I}(\Theta)$. More generally, given some constraints $h_1, \ldots, h_m$ of a variety Θ, we can find additional constraints $h_m + 1, \ldots, h_m + r$ that also vanish on the variety by finding a Gröbner basis of $\langle h_1, \ldots, h_m \rangle$ [22].

[33] This set is actually called an ideal. Note that it looks a lot like the span of some basis in linear algebra, only that the coefficients are now arbitrary polynomials, and the basis vectors are polynomials as well.

Rotation Synchronization We illustrate the different approaches to finding redundant constraints in more detail with the special orthogonal group constraints of the RS example. For simplicity, we focus on $\mathrm{SO}(2)$ here. We parameterize the rotation matrix using $\boldsymbol{\theta} = [a, b, c, d]^\top$, with

$$\boldsymbol{R} \in \mathrm{SO}(2), \boldsymbol{R} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}. \tag{45}$$

The constraints $\boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{I} = \boldsymbol{0}$ and $\det \boldsymbol{R} = 1$ are then equivalent to the following equality constraints:

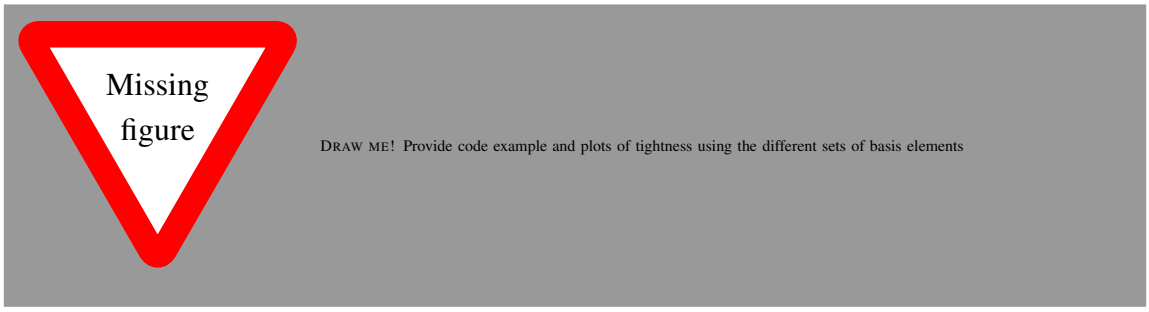
$$h_1(\boldsymbol{\theta}) = a^2 + c^2 - 1 \tag{46}$$
$$h_2(\boldsymbol{\theta}) = ab + cd \tag{47}$$
$$h_3(\boldsymbol{\theta}) = b^2 + d^2 - 1 \tag{48}$$
$$h_4(\boldsymbol{\theta}) = ad - bc - 1 \tag{49}$$

Manual Approach Using the manual approach, we know that at least the following constraints, derived from $\boldsymbol{R}\boldsymbol{R}^\top - \boldsymbol{I}$, must also hold:

$$h_5(\boldsymbol{\theta}) = a^2 + b^2 - 1 \tag{50}$$
$$h_6(\boldsymbol{\theta}) = ac + bd \tag{51}$$
$$h_7(\boldsymbol{\theta}) = c^2 + d^2 - 1 \tag{52}$$

Symbolic Approach Calling the Buchberger algorithm from `sympy` to find a Gröbner basis of the ideal of $\langle h_1, \ldots, h_4 \rangle$, we obtain the following elements:

$$\bar{h}_1(\boldsymbol{\theta}) = a - d \tag{53}$$
$$\bar{h}_2(\boldsymbol{\theta}) = b + c \tag{54}$$
$$\bar{h}_3(\boldsymbol{\theta}) = c^2 + d^2 - 1 \tag{55}$$



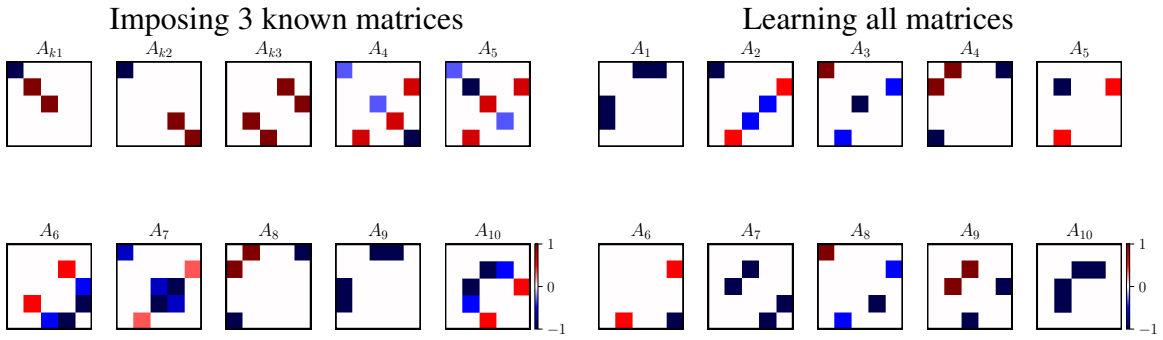


Figure 4: Results of AUTOTIGHT applied to $\mathrm{SO}(2)$ constraints. In the left 2×5 blocks, the constraints $\boldsymbol{A}_{k1}$ to $\boldsymbol{A}_{k3}$, corresponding to h_1 to h_3, are imposed. In the right 2×5 blocks, all constraints are learned.

Numeric Approach Finally, using the numerical algorithm we find in total 10 matrices, as shown in Figure 4. Interestingly, the determinant constraint actually yields a constraint that is redundant in the QCQP formulation, so it has to be treated as a redundant constraint here.

We highlight that neither the Gröbner basis nor the numerically found redundant constraints are unique. Running the above example with a different variable ordering for the Gröbner basis, or with different random samples for the numeric approach, may output different results.

References

[1] F Alizadeh. “Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization”. In: ().

[2] Farid Alizadeh, Jean-Pierre A. Haeberly, and Michael L. Overton. “Complementarity and Nondegeneracy in Semidefinite Programming”. In: *Mathematical Programming* 77.1 (Apr. 1997), pp. 111–128.

[3] MOSEK ApS. “MOSEK optimization toolbox for MATLAB”. In: *User’s Guide and Reference Manual, Version* 4.1 (2019), p. 116.

[4] Afonso S Bandeira. “A note on probably certifiably correct algorithms”. In: *Comptes Rendus. Mathématique* 354.3 (2016), pp. 329–333.

[5] Afonso S. Bandeira, Nicolas Boumal, and Amit Singer. “Tightness of the Maximum Likelihood Semidefinite Relaxation for Angular Synchronization”. In: *Mathematical Programming* 163.1-2 (2017). http://arxiv.org/abs/1411.3272, pp. 145–167.

[6] Timothy D Barfoot, Connor Holmes, and Frederike Dümbgen. "Certifiably Optimal Rotation and Pose Estimation Based on the Cayley Map". In: *International Journal of Robotics Research* (2024). https://doi.org/10.1177/02783649241269337.

[7] Timothy D. Barfoot. *State Estimation for Robotics*. https://www.cambridge.org/core/product/identifier/9781316671528/type/book. Cambridge University Press, 2017.

[8] Stefania Bellavia, Jacek Gondzio, and Margherita Porcelli. "A Relaxed Interior Point Method for Low-Rank Semidefinite Programming Problems with Applications to Matrix Completion". In: *Journal of Scientific Computing* 89.2 (Oct. 2021), p. 46.

[9] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes". In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606.

[10] John T Betts. *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

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

[12] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[13] Nicolas Boumal. *A Riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints*. en. arXiv:1506.00575 [math]. Jan. 2016. URL: http://arxiv.org/abs/1506.00575.

[14] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[15] Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. "Manopt, a Matlab toolbox for optimization on manifolds". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1455–1459.

[16] Nicolas Boumal, Vladislav Voroninski, and Afonso S Bandeira. "The non-convex Burer–Monteiro approach works on smooth semidefinite programs". en. In: (2016).

[17] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. SIAM, 1994.

[18] Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge, UK ; New York: Cambridge University Press, 2004.

[19] Jesus Briales and Javier Gonzalez-Jimenez. "Cartan-Sync: Fast and Global SE(d)-Synchronization". In: *IEEE Robotics and Automation Letters* 2.4 (Oct. 2017), pp. 2127–2134.

[20] Jesus Briales and Javier Gonzalez-Jimenez. "Convex Global 3D Registration with Lagrangian Duality". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. http://ieeexplore.ieee.org/document/8100078/. 2017, pp. 5612–5621.

[21] Lucas Brynte, Viktor Larsson, José Pedro Iglesias, Carl Olsson, and Fredrik Kahl. "On the Tightness of Semidefinite Relaxations for Rotation Estimation". In: *Journal of Mathematical Imaging and Vision* 64.1 (Jan. 2022), pp. 57–67.

[22] Bruno Buchberger. "An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal". https://linkinghub.elsevier.com/retrieve/pii/S0747717105001483. PhD thesis. Johannes Kepler University of Linz, 1965.

[23] Samuel Burer and Renato D.C. Monteiro. "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization". en. In: *Mathematical Programming* 95.2 (Feb. 2003), pp. 329–357. ISSN: 0025-5610, 1436-4646. URL: http://link.springer.com/10.1007/s10107-002-0352-8.

[24] Samuel Burer and Renato D.C. Monteiro. "Local minima and convergence in low-rank semidefinite programming". en. In: *Mathematical Programming* 103.3 (July 2005), pp. 427–444. ISSN: 0025-5610, 1436-4646. URL: http://link.springer.com/10.1007/s10107-004-0564-1.

[25] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[26] Luca Carlone. "Estimation Contracts for Outlier-Robust Geometric Perception". In: *Foundations and Trends in Robotics* 11.2-3 (2023). http://arxiv.org/abs/2208.10521, pp. 90–224.

[27] Graziano Chesi. "LMI techniques for optimization over polynomials in control: a survey". In: *IEEE transactions on Automatic Control* 55.11 (2010), pp. 2500–2510.

[28] Diego Cifuentes, Sameer Agarwal, Pablo A. Parrilo, and Rekha R. Thomas. "On the Local Stability of Semidefinite Relaxations". In: *Mathematical Programming* 193.2 (June 2022), pp. 629–663.

[29] Diego Cifuentes and Pablo A. Parrilo. "Sampling Algebraic Varieties for Sum of Squares Programs". In: *SIAM Journal on Optimization* 27.4 (2017). http://arxiv.org/abs/1511.06751, pp. 2381–2404.

[30] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Undergraduate Texts in Mathematics. https://link.springer.com/10.1007/978-3-319-16721-3. Cham: Springer International Publishing, 2015.

[31] Alexandre d'Aspremont, Damien Scieur, Adrien Taylor, et al. "Acceleration methods". In: *Foundations and Trends® in Optimization* 5.1-2 (2021), pp. 1–245.

[32] Frank Dellaert and Michael Kaess. "Factor Graphs for Robot Perception". In: *Foundations and Trends® in Robotics* 6.1-2 (2017). https://www.nowpublishers.com/article/Details/ROB-043, pp. 1–139.

[33] Frank Dellaert, David M. Rosen, Jing Wu, Robert Mahony, and Luca Carlone. "Shonan Rotation Averaging: Global Optimality by Surfing $SO(p)^n$". In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Vol. 12351. Cham: Springer International Publishing, 2020, pp. 292–308.

[34] Remy Derollez, Simon Le Cleac'h, and Zachary Manchester. "Robust Entry Vehicle Guidance with Sampling-Based Invariant Funnels". In: *2021 IEEE Aerospace Conference (50100)*. https://ieeexplore.ieee.org/abstract/document/9438259. Mar. 2021, pp. 1–9.

[35] Steven Diamond and Stephen Boyd. "CVXPY: A Python-embedded Modeling Language for Convex Optimization". In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[36] Dmitriy Drusvyatskiy and Henry Wolkowicz. "The Many Faces of Degeneracy in Conic Optimization". In: *arXiv:1706.03705* (June 2017). arXiv: 1706.03705 [math].

[37] Frederike Dümbgen, Connor Holmes, Ben Agro, and Timothy D. Barfoot. "Toward Globally Optimal State Estimation Using Automatically Tightened Semidefinite Relaxations". In: *IEEE Transactions on Robotics* 40 (2024). http://arxiv.org/abs/2308.05783, pp. 4338–4358.

[38] Mirjam Dur. "Copositive Programming – a Survey". In: ().

[39] Anders Eriksson, Carl Olsson, Fredrik Kahl, and Tat-Jun Chin. "Rotation Averaging and Strong Duality". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, June 2018, pp. 127–135.

[40] Osman Gfiler. "Convergence Behavior of Interior-Point Algorithms". In: ().

[41] Matthew Giamou, Ziye Ma, Valentin Peretroukhin, and Jonathan Kelly. "Certifiably Globally Optimal Extrinsic Calibration From Per-Sensor Egomotion". In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 367–374.

[42] Michel X. Goemans and David P. Williamson. "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming". In: *Journal of the ACM* 42.6 (1995). https://dl.acm.org/doi/10.1145/227683.227684, pp. 1115–1145.

[43] Abhishek Goudar, Frederike Dümbgen, Timothy D. Barfoot, and Angela P. Schoellig. "Optimal Initialization Strategies for Range-Only Trajectory Estimation". In: *IEEE Robotics and Automation Letters* 9.3 (2024). https://ieeexplore.ieee.org/document/10400824, pp. 2160–2167.

[44] Bernhard P. Graesdal, Shao Y. C. Chia, Tobia Marcucci, Savva Morozov, Alexandre Amice, Pablo A. Parrilo, and Russ Tedrake. "Towards Tight Convex Relaxations for Contact-Rich Manipulation". In: *Robotics: Science and System*. http://arxiv.org/abs/2402.10312. 2024.

[45] Michael Grant and Stephen Boyd. *CVX: Matlab Software for Disciplined Convex Programming, Version 2.1*. Mar. 2014.

[46] Gert-Martin Greuel. *Computer Algebra and Algebraic Geometry - Achievements and Perspectives*. http://arxiv.org/abs/math/0002247. 2000.

[47] Robert Grone, Charles R. Johnson, Eduardo M. Sá, and Henry Wolkowicz. "Positive definite completions of partial Hermitian matrices". In: *Linear algebra and its applications* 58 (1984), pp. 109–124.

[48] Soodeh Habibi, Michal Kočvara, and Michael Stingl. "Loraine – an interior-point solver for low-rank semidefinite programming". en. In: *Optimization Methods and Software* 39.6 (Nov. 2024), pp. 1185–1215. ISSN: 1055-6788, 1029-4937. URL: https://www.tandfonline.com/doi/full/10.1080/10556788.2023.2250522.

[49] Nicklas Hansen and Andreas Ostermeier. "Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation." In: *International Conference on Evolutionary Computation (ICEC)*. http://www.cmap.polytechnique.fr/~nikolaus.hansen/CMAES.pdf. 1996, pp. 312–317.

[50] Didier Henrion and Jean-Bernard Lasserre. "GloptiPoly: Global Optimization over Polynomials with Matlab and SeDuMi". In: ().

[51] Connor Holmes, Frederike Dümbgen, and Timothy Barfoot. "On Semidefinite Relaxations for Matrix-Weighted State-Estimation Problems in Robotics". In: *IEEE Transactions on Robotics* 40 (2024), pp. 4805–4824.

[52] Connor Holmes, Frederike Dümbgen, and Timothy D. Barfoot. "On Semidefinite Relaxations for Matrix-Weighted State-Estimation Problems in Robotics". In: *IEEE Transactions on Robotics* 40 (2024). https://doi.org/10.1109/TRO.2024.3475220, pp. 4805–4824.

[53] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. "A Faster Interior Point Method for Semidefinite Programming". In: *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*. Durham, NC, USA: IEEE, Nov. 2020, pp. 910–918.

[54] Xin Jiang, Yifan Sun, Martin Skovgaard Andersen, and Lieven Vandenberghe. "Minimum-rank positive semidefinite matrix completion with chordal patterns and applications to semidefinite relaxations". In: *Applied Set-Valued Analysis and Optimization* 5.2 (2023), pp. 265–283.

[55] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. "STOMP: Stochastic trajectory optimization for motion planning". In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 4569–4574.

[56] Andrew V Knyazev. "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method". In: *SIAM journal on scientific computing* 23.2 (2001), pp. 517–541.

[57] Vassili Korotkine, Mitchell Cohen, and James Richard Forbes. *Globally Optimal Data-Association-Free Landmark-Based Localization Using Semidefinite Relaxations*. http://arxiv.org/abs/2504.08547. Apr. 2025.

[58] Jean B. Lasserre. "Global Optimization with Polynomials and the Problem of Moments". In: *SIAM Journal on Optimization* 11.3 (2001). http://epubs.siam.org/doi/10.1137/S1052623400366802, pp. 796–817.

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

[60] Alex Lemon, Anthony Man-Cho So, and Yinyu Ye. "Low-Rank Semidefinite Programming: Theory and Applications". In: *Foundations and Trends® in Optimization* 2.1-2 (2016), pp. 1–156.

[61] Jörg Liesen and Zdenek Strakos. *Krylov subspace methods: principles and analysis*. Numerical Mathematics and Scie, 2013.

[62] Marco Locatelli and Fabio Schoen. *Global Optimization*. MOS-SIAM Series on Optimization. https://epubs.siam.org/doi/book/10.1137/1.9781611972672. Society for Industrial and Applied Mathematics, Oct. 2013.

[63] Zhi-quan Luo, Wing-kin Ma, Anthony So, Yinyu Ye, and Shuzhong Zhang. "Semidefinite Relaxation of Quadratic Optimization Problems". In: *IEEE Signal Processing Magazine* 27.3 (May 2010), pp. 20–34.

[64] Jan R. Magnus and Heinz Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Third edition. Wiley Series in Probability and Statistics. Hoboken, NJ: Wiley, 2019.

[65] Danylo Malyuta, Taylor P Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behçet Açıkmeşe. "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently". In: *IEEE Control Systems Magazine* 42.5 (2022), pp. 40–113.

[66] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. "Motion Planning around Obstacles with Convex Optimization". In: *Science Robotics* 8.84 (2023). https://www.science.org/doi/10.1126/scirobotics.adf7843.

[67] Alexander Morgan and Andrew Sommese. "A Homotopy for Solving General Polynomial Systems That Respects *M*-Homogeneous Structures". In: *Applied Mathematics and Computation* 24.2 (1987). https://www.sciencedirect.com/science/article/pii/0096300387900634, pp. 101–113.

[68] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, Jan. 1994.

[69] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2nd ed. Springer Series in Operations Research. New York: Springer, 2006.

[70] Carl Olsson, Yaroslava Lochman, Johan Malmport, and Christopher Zach. "Certifiably Optimal Anisotropic Rotation Averaging". In: *arXiv:2503.07353* (Mar. 2025). arXiv: 2503.07353 [cs].

[71] Antonis Papachristodoulou, James Anderson, Giorgio Valmorbida, Stephen Prajna, Pete Seiler, Pablo Parrilo, Matthew M. Peet, and Declan Jagt. "SOSTOOLS Version 4.00 Sum of Squares Optimization Toolbox for MATLAB". In: *arXiv:1310.4716* (Dec. 2021). arXiv: 1310.4716 [math].

[72] Alan Papalia, Andrew Fishberg, Brendan W. O'Neill, Jonathan P. How, David M. Rosen, and John J. Leonard. "Certifiably Correct Range-Aided SLAM". In: *IEEE Transactions on Robotics* (2024). https://doi.org/10.48550/arXiv.2302.11614.

[73] Alan Papalia, Joseph Morales, Kevin J. Doherty, David M. Rosen, and John J. Leonard. "SCORE: A Second-Order Conic Initialization for Range-Aided SLAM". In: *IEEE International Conference on Robotcis and Automation (ICRA)*. http://arxiv.org/abs/2210.03177. 2023.

[74] P.A. Parrilo. "Exploiting Structure in Sum of Squares Programs". In: *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*. Vol. 5. https://ieeexplore.ieee.org/document/1272305/?arnumber=1272305. 2003, 4664–4669 Vol.5.

[75] Pablo A. Parrilo. *MIT 6.972: Algebraic Techniques and Semidefinite Optimization*. Tech. rep. https://ocw.mit.edu/courses/6-972-algebraic-techniques-and-semidefinite-optimization-spring-2006/. MIT, 2006.

[76] Danda Pani Paudel and Luc Van Gool. "Sampling Algebraic Varieties for Robust Camera Autocalibration". In: *ECCV*. Vol. 11216. https://link.springer.com/10.1007/978-3-030-01258-8_17. Cham: Springer International Publishing, 2018, pp. 275–292.

[77] Frank Permenter and Pablo Parrilo. “Partial Facial Reduction: Simplified, Equivalent SDPs via Approximations of the PSD Cone”. In: *arXiv:1408.4685* 171 (Sept. 2018), pp. 1–54. arXiv: 1408.4685 [math].

[78] Michael Posa, Cecilia Cantu, and Russ Tedrake. “A direct method for trajectory optimization of rigid bodies through contact”. In: *The International Journal of Robotics Research* 33.1 (2014), pp. 69–81.

[79] James Blake Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd. Santa Barbara, California: Nob Hill Publishing, 2020.

[80] David M Rosen. “Accelerating certifiable estimation with preconditioned eigensolvers”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 12507–12514.

[81] David M Rosen. “Scalable low-rank semidefinite programming for certifiably correct machine perception”. In: *International Workshop on the Algorithmic Foundations of Robotics*. Springer. 2020, pp. 551–566.

[82] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. “SE-Sync: A Certifiably Correct Algorithm for Synchronization over the Special Euclidean Group”. In: *The International Journal of Robotics Research* 38.2-3 (Mar. 2019), pp. 95–125.

[83] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[84] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. “Motion planning with sequential convex optimization and convex collision checking”. In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1251–1270.

[85] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. “Taking the Human Out of the Loop: A Review of Bayesian Optimization”. In: *Proceedings of the IEEE* 104.1 (Jan. 2016). https://ieeexplore.ieee.org/document/7352306, pp. 148–175.

[86] Shen Shen and Russ Tedrake. “Sampling Quotient-Ring Sum-of-Squares Programs for Scalable Verification of Nonlinear Systems”. In: *IEEE Conference on Decision and Control (CDC)*. 2020, pp. 2535–2542.

[87] Jingnan Shi, Heng Yang, and Luca Carlone. “Optimal Pose and Shape Estimation for Category-level 3D Object Perception”. In: *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, July 2021.

[88] N.Z. Shor. “Quadratic Optimization Problems”. In: *Soviet Journal of Computer and Systems Sciences* 25 (1987), pp. 1–11.

[89] Justin Solomon. *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*. Boca Raton: CRC Press, Taylor & Francis Group, 2015.

[90] Andrew J Sommese and Charles W Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. https://www.worldscientific.com/worldscibooks/10.1142/5763. 2005.

[91] Bernd Sturmfels. *Solving Systems of Polynomial Equations*. 2002.

[92] Yuval Tassa, Tom Erez, and Emanuel Todorov. "Synthesis and stabilization of complex behaviors through online trajectory optimization". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 4906–4913.

[93] Sangli Teng, Ashkan Jasour, Ram Vasudevan, and Maani Jadidi. "Convex Geometric Motion Planning on Lie Groups via Moment Relaxation". In: *Robotics: Science and Systems XIX*. http://www.roboticsproceedings.org/rss19/p058.pdf. 2023.

[94] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005.

[95] Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. "On the implementation and usage of SDPT3 –a MATLAB software package for semidefinite-quadratic-linear programming, version 4.0". In: *Handbook on semidefinite, conic and polynomial optimization* (2012), pp. 715–754.

[96] Lieven Vandenberghe and Martin S. Andersen. "Chordal graphs and semidefinite optimization". In: *Foundations and Trends® in Optimization* 1.4 (2015), pp. 241–433.

[97] Gerd Wachsmuth. "On LICQ and the uniqueness of Lagrange multipliers". In: *Operations Research Letters* 41.1 (2013), pp. 78–80.

[98] Jie Wang and Liangbing Hu. "Solving low-rank semidefinite programs via manifol optimization". en. In: (2023).

[99] Jie Wang, Victor Magron, and Jean-Bernard Lasserre. "TSSOS: A Moment-SOS Hierarchy That Exploits Term Sparsity". In: *SIAM Journal on Optimization* 31.1 (Jan. 2021), pp. 30–58.

[100] Jie Wang, Victor Magron, and Jean-Bernard Lasserre. "TSSOS: A Moment-SOS Hierarchy That Exploits Term Sparsity". In: *SIAM Journal on Optimization* 31.1 (2021). https://epubs.siam.org/doi/10.1137/19M1307871, pp. 30–58.

[101] Jie Wang, View Profile, Victor Magron, View Profile, J. B. Lasserre, View Profile, Ngoc Hoang Anh Mai, and View Profile. "CS-TSSOS: Correlative and Term Sparsity for Large-Scale Polynomial Optimization". In: *ACM Transactions on Mathematical Software* 48.4 (Dec. 2022), pp. 1–26.

[102] Heng Yang and Luca Carlone. "Certifiably Optimal Outlier-Robust Geometric Perception: Semidefinite Relaxations and Scalable Global Optimization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (2023). http://arxiv.org/abs/2109.03349, pp. 2816–2834.

[103] Heng Yang and Luca Carlone. "Certifiably Optimal Outlier-Robust Geometric Perception: Semidefinite Relaxations and Scalable Global Optimization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (Mar. 2023), pp. 2816–2834.

[104] Heng Yang, Ling Liang, Luca Carlone, and Kim-Chuan Toh. "An inexact projected gradient method with rounding and lifting by nonlinear programming for solving rank-one semidefinite relaxation of polynomial optimization". en. In: *Mathematical Programming* 201.1-2 (Sept. 2023), pp. 409–472. ISSN: 0025-5610, 1436-4646. URL: https://link.springer.com/10.1007/s10107-022-01912-6.

[105] Xihang Yu and Heng Yang. "SIM-Sync: From Certifiably Optimal Synchronization Over the 3D Similarity Group to Scene Reconstruction With Learned Depth". In: *IEEE Robotics and Automation Letters* 9.5 (May 2024), pp. 4471–4478.

[106] Yang Zheng, Giovanni Fantuzzi, and Antonis Papachristodoulou. "Chordal and factor-width decompositions for scalable semidefinite and polynomial optimization". In: *Annual Reviews in Control* 52 (2021), pp. 243–279.

[107] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. "CHOMP: Covariant hamiltonian optimization for motion planning". In: *The International journal of robotics research* 32.9-10 (2013), pp. 1164–1193.