Convergence failures of the Burer-Monteiro factorization algorithm

Applying the latest research findings to the trust-region subproblem

Maurits Brinkman
S3344711

A thesis presented for the course
Bachelor's Project Mathematics and Applied Mathematics

Faculty of Science and Engineering
University of Groningen
Netherlands
November 2020

Convergence failures of the Burer-Monteiro factorization algorithm

Applying the latest research findings to the trust-region subproblem

Maurits Brinkman

Abstract

In recent literature, the Burer-Monteiro factorization algorithm has shown to be an efficient method to solve large scale semidefinite programs that admit a solution of low rank: instead of optimizing over a sizable matrix X, one splits this matrix apart into two smaller ones by factorizing $X = VV^\top$. The disadvantage of factorizing X in this way is that possibly extra non-optimal second-order critical V's are generated. While the article [Bandeira, Boumal, and Voroninski 2020] shows that in many situations this downside of the Burer-Monteiro approach is benign, recently [Waldspurger and Waters 2019] determined cases for which this factorization approach leads to an incorrect solution. In this current thesis, the results of both of these papers are combined by considering a classic minimization problem concerning optimization over quadratics, known as the trust-region subproblem. In particular, Chapter 4 will provide two examples of how to construct the trust-region subproblem in such a way that factorizing $X = VV^\top$ is not benign, and will therefore lead to convergence failure when applying the Burer-Monteiro factorization algorithm. Both of these examples are built on a step-by-step framework for how to deliberately bring about convergence failure of the Burer-Monteiro factorization algorithm, which will be discussed in Chapter 3.

Contents

Chapter 1

Introduction

The *Burer-Monteiro factorization algorithm* is a relatively new approach to solving semidefinite programs (SDPs), designed by Samuel Burer and Renato Monteiro [Burer and Monteiro 2003; Burer and Monteiro 2005]. An SDP is a problem of the following form:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{such that} \quad & \mathcal{A}(X) = b, \\ & X \succeq 0, \end{aligned} \tag{SDP}$$

where $\langle C, X \rangle = \text{Tr}(C^\top X)$, as the latter denotes the trace of the $n \times n$-matrix $C^\top X$. Here, we minimize over the positive semidefinite matrix X in the space of symmetric $n \times n$-matrices. Besides that, $C \in \mathbb{R}^{n \times n}$ is a fixed and symmetric cost matrix, and $\mathcal{A} : \mathbb{R}^{n \times n} \to \mathbb{R}^m$ is a linear operator that captures m equality constraints, for which $b \in \mathbb{R}^m$. Lastly, $X \succeq 0$ refers to the requirement that X is positive semidefinite.

Problem (SDP) could be used as a relaxation for various difficult problems, with a suitable change of variables, to make them easier to solve. One of the classic examples in which SDPs can be used as a relaxation is *MaxCut*, which is a problem from graph theory that first appeared in [Delorme and Poljak 1993; Svatopluk Poljak 1995] and made famous by the work [Goemans and Williamson 1995]. In particular, in the MaxCut problem, one considers a graph of vertices (points) of which some are connected by edges, and where the goal is to divide the vertices into two sets, such that the line drawn to separate these sets cuts the maximum number of edges.

In order to solve the SDP-relaxations of difficult problems like MaxCut, nowadays one can choose from a whole variety of iterative solvers (interior point methods) to obtain the solution to Problem (SDP). In small scale cases, using these iterative algorithms is efficient, since SDPs are convex. This latter fact allows one to solve these programs in polynomial time[1], while the original problems (without using any relaxation) are in many practical settings NP-hard[2]. This makes rewriting complex problems into the form of Problem (SDP) an already established and often-used method to solve optimization problems.

The issue here, is that in many practical cases X is a very large matrix, which makes conventional iteration methods for solving Problem (SDP) computationally demanding and slow. To overcome this, [Burer and Monteiro 2003] used the main results of [Pataki 1998] and

[1] When an optimization problem can be solved in polynomial time, it is possible to solve the problem without waiting an infinite amount of hours.

[2] NP-hard refers to the fact that it is impossible to solve that particular problem directly (i.e. without using a relaxation).

[Barvinok 1995] to come up with an algorithm that solves Problem (SDP) more efficiently. Namely, the latter two articles pointed out that the optimal solution of Problem (SDP), which we will denote as X_{opt}, is often of low rank[3]. Now, one of the useful properties of low-rank semidefinite matrices $X \in \mathbb{R}^{n\times n}$, is that they can easily be factored out in the form $X = VV^\top$, where $V \in \mathbb{R}^{n\times p}$. This is convenient, since it is true that in empirical settings $p \ll n$, and hence V is a significantly smaller than X.

The Burer-Monteiro factorization approach substitutes $VV^\top$ for X into Problem (SDP), which yields

$$\begin{aligned} \text{minimize} \quad & \langle CV, V\rangle \\ \text{such that} \quad & \mathcal{A}(VV^\top) = b, \end{aligned} \tag{Factorized SDP}$$

where we now optimize over $V \in \mathbb{R}^{n\times p}$, instead of optimizing over $X \in \mathbb{R}^{n\times n}$. Also, we note that $\langle CVV^\top\rangle = \langle CV, V\rangle = \mathrm{Tr}(C^\top VV^\top)$.

Motivation behind the Burer-Monteiro factorization algorithm

When compared with Problem (SDP), solving Problem (Factorized SDP) is less computationally demanding, since we optimize over a smaller matrix, as we already mentioned that in most applications $p \ll n$. In Problem (SDP) we had to deal with the $n \times n$-matrix X, while when using Problem (Factorized SDP), one works with the $n \times p$-matrix V. Now, in the former case we have to optimize over n^2 variables, while in the latter we only have np variables. This makes solving Problem (Factorized SDP) far less computationally expensive than finding the global minimum of Problem (SDP), when applying iterative solvers. Another advantage of the Burer-Monteiro factorization is that the positive semidefiniteness constraint in Problem (SDP) can be omitted, since it will be automatically satisfied using $X = VV^\top$. This last point is very relevant, since $X \succeq 0$ is "the most challenging aspect" of solving Problem (SDP), as Burer and Monteiro mention in their paper [Burer and Monteiro 2003, page 330].

There has been a lot of work on the Burer-Monteiro approach in recent years. Research of great value such as the first manifold-inspired perspective [Journée et al. 2010] and the fact that points which satisfy necessary optimality conditions approximately are also approximately optimal [Bhojanapalli et al. 2018; Pumir, Jelassi, and Boumal 2018; Cifuentes and Moitra 2019] can therefore not be missed.

Downsides of the Burer-Monteiro approach

Although rewriting Problem (SDP) into Problem (Factorized SDP) makes solving computationally easier, there are a few downsides when using this factorization algorithm. Firstly, when substituting $VV^\top$ for X into the objective function of the SDP, it may be that extra non-optimal second-order critical points are introduced. For the case that such an additional critical point has lower objective function than the V that corresponds to the global optimum of the SDP[4], optimization algorithms over the factorized problem are doomed to find the V

[3] In [Pataki 1998, Theorem 2.1] it is stated $\mathrm{rank}(X_{opt}) \lesssim \sqrt{2m}$, where we recall that m is the number of affine contraints. Hence, $\sqrt{2m}$ defines an approximate upper bound for the rank of the optimal solution which, we can now say, must be of low rank.

[4] As we will see in Chapter 3, this situation lies at the base of Step 5 of examples 1 and 2. Namely, we can use it for showing that the V_0 (that corresponds to the optimal solution of the SDP) is a non-optimal second-order

that is optimal for the Burer-Monteiro factorization. Instead, they find a (for the factorized problem) global optimal V that corresponds to a non-optimal X of the SDP. This issue is fully dependent on the cost matrix C and on p, as known from [Waldspurger and Waters 2019, Sections 3.3, Section 5.1].

To make matters even worse, by going from the convex Problem (SDP) to the non-convex Problem (Factorized SDP), it may be that the optimization algorithms that are used over the Burer-Monteiro factorization cannot even find the global minimum. In particular, it can be that (Riemannian[5]) optimization algorithms get stuck at a non-optimal second-order critical point, instead of finding a global minimizer of the SDP, due to non-convexity of the factorized problem. Whether this issue arises or not depends on the following pitfall of Problem (Factorized SDP), namely the choice of the column dimension of V, denoted by p.

In particular, the third downside of using Problem (Factorized SDP) is that the factorization rank[6] p has to be chosen. Nevertheless, in many emperical experiments, interior point methods solve Problem (Factorized SDP) when $p \geq \text{rank}(X_{opt})$. Numerical examples of this result can be found in [Burer and Monteiro 2003], [Boumal 2016], [Bandeira, Boumal, and Voroninski 2016a] and [Rosen et al. 2019]. In all of these papers, restrictive assumptions are made on the cost matrix C in order for local optimization algorithms to solve Problem (Factorized SDP). If no restrictions on C are made, there are no guarantees for convergence to a global minimum, unless $p \gtrsim \sqrt{2m}$, where we recall that m is the number of constraints, as can be found in [Bandeira, Boumal, and Voroninski 2020][7]. Recently, [Waldspurger and Waters 2019] slightly improved this bound, but confirmed that if $p \lesssim \sqrt{2m}$, Riemannian optimization algorithms cannot be certified correct without assumptions on C.

Lastly, it is important to note that while (the searching space of) Problem (SDP) is convex, Problem (Factorized SDP) is not. For this disadvantage [Bandeira, Boumal, and Voroninski 2016b] showed that non-convexity is benign when the threshold of $p \gtrsim \sqrt{2m}$ is satisfied.

Convergence failures for the Burer-Monteiro appraoch

In this thesis we will explore the phenomenon that the choice of both the factorization rank p and cost matrix C determines the possibility of convergence failure[8] of Riemannian optimization algorithms on Problem (Factorized SDP). This will be done by reconsidering one of the applications from the latest research of [Bandeira, Boumal, and Voroninski 2020, Section 5.2], namely the trust-region subproblem (which nowadays is a classic minimization problem concerning optimization over quadratics).

We will see that actually it is not difficult to come up with C's that lead to an incorrect

critical point: there exists a feasible second-order critical V_1 that has lower objective function than V_0, which satisfies $X_0 = V_0 V_0$ and where X_0 is the global minimum of the SDP.

[5]Riemannian optimization algorithms are a specific class of interior point methods. Nowadays, they are considered as the most efficient solvers for SDPs, since these algorithms iterate on the manifold directly: all iterates satisfy constraints up to numerical accuracy.

[6]Instead of saying "the column dimension of V", the common term used in the literature is *the factorization rank p*.

[7]The only assumption made in this paper is that the feasible set of matrices defined as $\mathcal{M}_p = \{V \in \mathbb{R}^{n \times p} : \mathcal{A}(VV^\top) = b$ is a smooth manifold. Under this assumption Problem (Factorized SDP) satisfies the Karush-Kuhn-Tucker (KKT) conditions, which makes points (i.e. matrices) more efficiently computable. Analysis of cases for which the Burer-Monteiro approach works for applications where the searching space for V is not a smooth manifold can be found in [Bhojanapalli et al. 2018; Cifuentes 2020].

[8]The precise definition of convergence failure will be stated in Chapter 2.

solution for Problem (SDP) when the Burer-Monteiro factorization algorithm is applied. In particular, when we add just one extra constraint to the trust-region subproblem given in [Bandeira, Boumal, and Voroninski 2020, Section 5.2], we can generate a list consisting of many 'bad' cost matrices for which the Burer-Monteiro factorization cannot find the global minimizer of Problem (SDP). This latter fact can easily be confirmed by the main result of [Waldspurger and Waters 2019], which stated that when $\frac{p(p+1)}{2} + r_0 p < m$ (where p is again the column dimension of V, m denotes the number of constraints, and $r_0 = \text{rank}(X_{opt})$), there exists a set of cost matrices for which solving Problem (Factorized SDP) results in a local optimum that is not a global minimum. In particular, in Chapter 4 we will commit to the choice $p = 1$, $m = 2$ and $r_0 = 1$ for which the assumption of [Waldspurger and Waters 2019, Theorem 2] is indeed satisfied. Therefore, there must exist a set of cost matrices C for which the optimal solution of Problem (Factorized SDP) does not correspond to the global minimum of Problem (SDP).

Now, it would be interesting to apply [Waldspurger and Waters 2019, Theorem 2] to applications given in the latest research of [Bandeira, Boumal, and Voroninski 2020, Section 5.2] to explore how constructing cost matrices that lead to convergence failure of the Burer-Monteiro factorization algorithm works in practice. This point leads to the first research goal of this thesis, namely computing a list of cost matrices C that result in convergence failure for the Burer-Monteiro factorization algorithm for the trust-region subproblem. Here, 'convergence failure' refers to two distinct possible cases, which are results of using iterative solvers to find the global optimal solution for Problem (Factorized SDP): (1) the case in which the sequence provided by the iterative solver is non-convergent, and (2) the case in which possible solutions do converge, but to a V that does not correspond[9] to the optimal solution of Problem (SDP)). This means that when $\frac{p(p+1)}{2} + pr_0 \leq m$, there exists a list of bad C's for which the Burer-Monteiro factorization gives either a wrong solution or no solution at all. In the former case, there exists a unique global minimizer for Problem (SDP), but the corresponding V is not a global optimum for Problem (Factorized SDP).

Thesis outline and research goals

Now, the point that [Waldspurger and Waters 2019, Theorem 2] has given us the guarantee of existence of a set of cost matrices for which the Burer-Monteiro factorization algorithm does not work, we will use this as the motivation for the research goal of this current thesis. The reader will be provided two examples of the trust-region subproblem of [Bandeira, Boumal, and Voroninski 2020, Section 5.2] where we will compute a list of cost matrices that will lead to convergence failure of the Burer-Monteiro factorization algorithm, which corresponds to the following research goal:

Compute a list of cost matrices C that result in convergence failure for the Burer-Monteiro factorization algorithm for some modified version of the trust-region subproblem as given in [Bandeira, Boumal, and Voroninski 2020, Section 5.2].

In order to make all of this possible, Chapter 2 will look at the preliminaries that are needed to understand when a specific point is locally or globally optimal and what it means to be a

[9] Applying $X = VV^\top$.

unique global optimum. Using those preliminaries, in Chapter 3 a step-by-step approach is constructed for generating C's for which applying the Burer-Monteiro factorization algorithm does not result in a global minimum X_{opt}, which we in turn will use for the two examples of convergence failure in Chapter 4.

As the author of this report, this pair of goals allowed me to apply new-learned theoretical knowledge to practical applications. Also, the amount of scientific articles that was needed to provide a solid background for generating the examples in this thesis, resulted in the fact that now I am not only able to read elementary scientific mathematics articles, but also I can extract the essence and most important content of these articles.

It is worth mentioning that the research goal of this thesis has been inspired by the work of [Celis 1984], which considered examples of convergence failure for iterative algorithms on equality constrained optimization problems.

Chapter 2

Preliminaries

2.1 Definitions

Definition 1. A *semidefinite program (SDP)* is a linear optimization program of the following form:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{such that} \quad & \mathcal{A}(X) = b, \\ & X \succeq 0 \end{aligned} \tag{SDP}$$

where we optimize over $X \in \mathbb{S}^{n\times n}$ and $\langle C, X\rangle = \mathrm{Tr}(C^\top X)$, as the latter denotes the trace of the $n \times n$-matrix $C^\top X$. Here, $\mathbb{S}^{n\times n}$ refers to the space of symmetric $n \times n$ matrices, $\mathcal{A} : \mathbb{S}^{n\times n} \to \mathbb{R}^m$ is a fixed linear map, $b \in \mathbb{R}^m$ is fixed, and $C \in \mathbb{S}^{n\times n}$ is called the *cost matrix*. Also, here $\langle C, X\rangle$ denotes the *objective function*.

Remark. Since C is a symmetric matrix, it follows that $C^\top = C$, and we will therefore omit the transpose symbol when using the Burer-Monteiro relaxation for notational convenience. For example, when in Chapter 4 we compute the objective function $\langle C, X\rangle = \mathrm{Tr}(CX)$, we will write the latter instead of $\mathrm{Tr}(C^\top X)$.

Definition 2. The *set of feasible points* $\mathcal{C}$ of an SDP (also referred to as the *searching space* of an SDP) is given by $\mathcal{C} = \{X \in \mathbb{S}^{n\times n} : \mathcal{A}(X) = b, X \succeq 0\}$.

As motivated in the introduction in Chapter 1, when one assumes that Problem (SDP) allows an optimal solution X_{opt} with rank r, and fixes some $p \geq r$, it is equivalent to its *rank p Burer-Monteiro factorization*.

Definition 3. A *Burer-Monteiro factorization* of an SDP, also referred to as *the factorized problem*, is a linear optimizationn program of the following form:

$$\begin{aligned} \text{minimize} \quad & \langle C, VV^\top \rangle \\ \text{such that} \quad & \mathcal{A}(VV^\top) = b, \end{aligned} \tag{Factorized SDP}$$

where we optimize over $V \in \mathbb{R}^{n\times p}$, and where p is called *factorization rank*. Here, $V^\top$ denotes the transpose of V, $\mathcal{A} : \mathbb{S}^{n\times n} \to \mathbb{R}^m$ is a fixed linear map, $b \in \mathbb{R}^m$ is fixed, and $C \in \mathbb{S}^{n\times n}$. Also, $\langle C, VV^\top\rangle = \langle CV, V\rangle = \mathrm{Tr}(CVV^\top)$.

Definition 4. The *set of feasible points* $\mathcal{M}_p$ of a Burer-Monteiro factorization (also referred to as the *searching space* of an Burer-Monteiro factorization) is given by $\mathcal{M}_p = \{V \in \mathbb{R}^{n \times p} : \mathcal{A}(VV^\top) = b\}$.

We note that when the factorization rank p is clear from the context, we will often write $\mathcal{M}$ instead of $\mathcal{M}_p$.

Definition 5. Let $\mathcal{N}$ be a Riemannian manifold, and $f : \mathcal{N} \to \mathbb{R}$ a smooth function. For any $x \in \mathcal{N}$, let $\nabla f(x)$ denote the gradient of f at x. We say that x is a *first-order critical point of* f iff $\nabla f(x) = 0$.

Next, we will provide the definition of a second-order critical point. This refers to local as well as global optimal points.

Definition 6. Let $\mathcal{N}$ be a Riemannian manifold, and $f : \mathcal{N} \to \mathbb{R}$ a smooth function. For any $x \in \mathcal{N}$, let $\operatorname{Hess} f(x)$ denote the Hessian of f at x. We say that x is a *optimal second-order critical point of* f iff both $\nabla f(x) = 0$ and $\operatorname{Hess} f(x) \succeq 0$.

One can distinguish optimal and non-optimal points of a minimization problem by the fact that a feasible second-order critical point X is non-optimal when there exists another feasible second-order critical point X_1 that leads to a lower output of the objective function. From this, it follows that a point X is a global optimum, when no such X_1 exists. Knowing this, the following definition can be formulated.

Definition 7. A cost matrix C is called *bad* iff both $\nabla\langle CX\rangle = \nabla\langle CV, V\rangle) = 0$ and $\operatorname{Hess}\langle CX\rangle \succeq 0$, and $\operatorname{Hess}\langle CV, V\rangle \succeq 0$, where X is the global optimum of the SDP, while V is a non-optimal second-order critical point for the factorized problem, satisfying $X = VV^\top$.

It follows that when a cost matrix is bad, applying the Burer-Monteiro factorization algorithm will lead to convergence failure. Namely, when one solves the factorized problem of a particular SDP, iterative algorithms such as Riemannian optimization will lead to the optimal solution V_{opt} of this particular Burer-Monteiro factorization. The caveat here is that the V_0 that corresponds to the optimal solution of the SDP (hence $X = V_0V_0^\top$) is non-optimal for the factorized problem and hence $V_0 \neq V_{opt}$. Informally, this means that Riemannian optimization algorithms cannot "find" the V_0 that corresponds to the global minimum of the SDP.

Definition 8. We say that *convergence failure* for the Burer-Monteiro factorization is the case when (1) the sequence provided by the iterative solver is non-convergent, or (2) possible solutions do converge, but to an X for which the corresponding V is not an optimal solution of the Burer-Monteiro factorization.

Here, *iterative solvers* refers to the iterative algorithms that are used to solve a Burer-Monteiro factorization. Often these are classes of Riemannian optimization algorithms (which are interior point solvers). Moreover, Burer and Monteiro proposed in [Burer and Monteiro 2003] to use methods such as Augmented Lagrangian and Dual Ascent. Since convergence failure is guaranteed when the global optimum of and SPD does not correspond to the global optimum of its factorized problem, it is not needed to use such advanced iterative algorithms to construct cost matrices that lead to convergence failure. Also, it can be helpful to know that a Burer-Monteiro factorization is slightly different than the Burer-Monteiro factorization *algorithm*. Namely, the Burer-Monteiro factorization algorithm proposes to factorize an SDP

into a Burer-Monteiro factorization, after which a specific iterative optimization algorithm is used to find a V_{opt} that is guaranteed to correspond to the optimal solution X_{opt} of the SDP when $p \gtrsim \sqrt{2m}$.

The next definition is important for stating that the minimizer X_{opt} of an SDP is *unique*, as we will see from Proposition 2.

Definition 9. We say that a solution X_{opt} of Problem (SDP) satisfies *strict complementary slackness at* X_{opt} if and only if for all $\varepsilon > 0$ there exists a neighbourhood $B_\varepsilon(C)$ around C, such that for all $\hat{C} \in B_\varepsilon(C)$ it holds that X_{opt} is also a solution of Problem (SDP) with the objective function $\langle \hat{C}, X \rangle$.

2.2 Basic properties

Proposition 1. A matrix $X \in \mathcal{C}$ is a solution of Problem (SDP) if and only if it is a second-order critical point.

Corollary 1. A matrix V is not a solution of Problem (Factorized SDP) if it is a non-optimal second-order critical point.

Proposition 2. A matrix $X \in \mathcal{C}$ is a *unique* solution of Problem (SDP) if and only if it is both a globally optimal second-order critical point and strict complementary slackness holds at X.

We omit the proofs of the above two propositions here, since these can be found in [Waldspurger and Waters 2019, Appendix A.1] and [Waldspurger and Waters 2019, Appendix A.2], respectively.

Now, the following theorem is important for the upcoming examples of convergence failure, since it gives a guarantee of existence of bad C's. This can be rather helpful when deliberately searching for these kind of cost matrices, as we will do in examples 1 and 2 of Chapter 4.

Theorem 1. Let $r_0 \in \mathbb{N}^*$ be fixed. Let $p \geq r_0$ be such that $\frac{p(p+1)}{2} + pr_0 \leq m$. If the following assumptions are satisfied, then there exists a neighbourhood of bad cost matrices C such that Problem (SDP) has a unique global minimizer of rank r_0 and Problem (Factorized SDP) has a non-optimal second-order critical point:

1. $\mathcal{C}$ has at least one extreme point X_{opt} with rank r_0;
2. $(\mathcal{A}, b)$ is p-regular;
3. There exists $V \in \mathcal{M}_p$ such that $\mathcal{M}_p$ is X_{opt}-minimally secant at V.

This theorem is the main result of [Waldspurger and Waters 2019], which is proved in Section 5 of that same paper. Here, $r_0 \in \mathbb{N}^*$ refers to the fact that r_0 is the rank of the optimal solution X_{opt}. Since the three assumptions above are always satisfied for any version of the trust-region subproblem, and for simplicity, we will assume that they are satisfied for the trust-region subproblems of Example 1 and Example 2.

Chapter 3

Step-by-step approach for constructing C

In this section, we will give a general outline of how to construct C's that lead to convergence failure of the Burer-Monteiro approach and will therefore not lead to a global minimum of the corresponding semidefinite program. To do this, we use the proof given in [Waldspurger and Waters 2019, Section 5] which illuminates how to construct such a bad C. In particular, the proof not only gives the reader an idea of how to construct such a bad cost matrix C, it also used classical geometrical arguments such as the implicit function theorem to show that there exists an open neighbourhood around a bad C in which all cost matrices are bad. This means that when we have found one bad C, we can apply small perturbations in order to come up with more such matrices. This latter result is important for generating the bad cost matrices in examples 1 and 2 of Chapter 4.

Now, consider the case in which you are dealing with a difficult optimization problem which can be relaxed to an SDP and its corresponding Burer-Monteiro factorization. Step 1 of constructing a bad C is reassuring the existence of a set of bad cost matrices, using Theorem 1:

Step 1 Check whether the SDP at hand satisfies $\frac{p(p+1)}{2} + pr_0 \leq m$.

Here, p denotes the factorization rank, m is the number of constraints and r_0 is the rank of the optimal solution of the SDP.

Step 2 Fix a feasible $X_0 \in \mathcal{C}$ that has rank r_0 and fix a $V_0 \in \mathcal{M}_p$.

To be clear, by Definition 2, we can choose an arbitrary matrix X_0 that satisfies the constraints of the SDP. Also, note that we are not fixing an *arbitrary* V_0, since obtaining V_0 will follow as a consequence of fixing X_0 by the fact that $X_0 = V_0 V_0^\top$.

After this, the next step reads:

Step 3 Construct C such that X_0 and V_0 are both second-order critical points for the SDP and its factorized problem, respectively (according to Definition 6).

By Definition 6, Step 3 equals finding a C for which $\nabla \operatorname{Tr}(C, X_0) = 0$, $\nabla \operatorname{Tr}(C, V_0 V_0^\top) = 0$ and $\operatorname{Hess} \operatorname{Tr}(C, X_0) \succeq 0$, and $\operatorname{Hess} \operatorname{Tr}(CV_0, V_0) \succeq 0$.

Step 4 Apply small perturbations to C in order to find other cost matrices that satisfy the conditions given in Step 3.

This step is substantiated by [pp. 15 Waldspurger and Waters 2019, Lemma 4], namely that Theorem 1 provides us existence of a *set* of bad cost matrices, which can be obtained by slightly changing a bad C. This means that there are cost matrices close to the bad C for which both X_0 and V_0 are still second-order critical.

Lastly, we need to make sure that X_0 is not just a local optimal point, but is actually the unique global minimizer of the SDP, that corresponds to a V that is not optimal (hence, a local minimizer) for the factorized problem. Namely, Step 3 guaranteed that the generated cost matrix will be such that X_0 and V_0 are second-order critical, but does not provide any information about the local or global optima. Therefore, the last step will provide assurance that X_0 is the unique global minimizer for the SDP at hand, but V_0 is a non-optimal point for the corresponding factorized problem.

Step 5 Check if it is indeed the case that there exists no feasible second-order critical X that lead to a lower output of the objective function of the SDP than the value of the objective function using X_0. Also, produce a feasible second-order critical V that has lower objective function output than V_0.

Now, in the next two chapters, the step-by-step approach we constructed above will be used as a guideline for finding C's for which the Burer-Monteiro approach leads to convergence failure.

Chapter 4

Convergence failures

In this chapter we will consider a slightly modified version of the trust-region subproblem (TRS) as given in [Bandeira, Boumal, and Voroninski 2020, Section 5.2]. The goal of this chapter is to apply Theorem 1 to a specific application, in this case the trust-region subproblem for which we restrict ourselves to small-dimensional cases, for computational convenience. Namely, for large-dimensional cases of this optimization problem, the general procedure of constructing bad C's will be exactly the same.

Trust-region subproblem SDP

The trust-region subproblem, as given in [Bandeira, Boumal, and Voroninski 2020, Section 5.2], consists in approximately solving a problem of the form:

$$\begin{aligned} \text{minimize} \quad & x^\top A x + 2b^\top x + c \\ \text{such that} \quad & ||x||^2 = 1 \end{aligned} \tag{TRS}$$

where we optimize over the variable $x \in \mathbb{R}^n$, which is a vector with $n \geq 3$, and where $(A, b, c) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n \times \mathbb{R}$ are fixed. In order to rewrite Problem (TRS) to the form of Problem (SDP), we will introduce

$$X = \begin{pmatrix} x \\ 1 \end{pmatrix} \begin{pmatrix} x^\top & 1 \end{pmatrix} = \begin{pmatrix} xx^\top & x \\ x^\top & 1 \end{pmatrix}, \quad \text{and} \quad C = \begin{pmatrix} A & b \\ b^\top & c \end{pmatrix}. \tag{4.1}$$

Since C is linear in X, we can formulate the following SDP relaxation:

$$\begin{aligned} \text{minimize} \quad & \langle C, X \rangle \\ \text{such that} \quad & \text{Trace}(X_{1:n,1:n}) = 1, \ X_{n+1,n+1} \\ & X \succeq 0. \end{aligned} \tag{TRS-SDP}$$

Here, again, we minimize over $X \in \mathbb{S}^{(n+1)\times(n+1)}$ and $X_{1:n,1:n}$ denotes the first $n \times n$ block of X in the obvious way.

The Burer-Monteiro factorization to this problem reads:

$$\begin{aligned} \text{minimize} \quad & \langle CV, V \rangle \\ \text{such that} \quad & ||V_1||^2 = 1, \ ||\mathrm{v}_2||^2 = 1 \\ & \text{with } V = \begin{pmatrix} V_1 \\ \mathrm{v}_2^\top \end{pmatrix} \end{aligned} \tag{TRS-BM}$$

where $V_1 \in \mathbb{R}^{n\times p}$ and $v_2 \in \mathbb{R}^p$ with factorization rank p, and hence $V \in \mathbb{R}^{n+1}$.

In order to come up with convergence failures for the Burer-Monteiro approach to solve Problem (TRS) in the upcoming examples, we will show how to deliberately construct (A, b, c) for Problem (TRS) such that this results in a 'bad' C for Problem (TRS-SDP).

As mentioned in the introduction of the current chapter, we will restrict ourselves to a small-dimensional case of Problem (TRS). In particular, this restriction allows us to present the application of the step-by-step approach constructed in Chapter 3 in a clear way, without having to work with a vast amount of variables. Therefore, we will now consider the trust-region subproblem and its corresponding SDP and Burer-Monteiro factorization for which $n = 3$, $m = 2$, $p = 1$ and $r_0 = 1$.

Trust-region subproblem where $n = 3$, $m = 2$, $p = 1$ and $r_0 = 1$

Now, we have finally reached the point at which we can apply the content provided in the former two chapters. First, we will apply Step 1 of the approach described in Chapter 2. After that, two concrete examples will be considered for which we fix an X and its corresponding V and compute a list of bad C's, for which steps 2, 3 and 4 of the step-by-step approach are used.

Step 1 We need to make sure that the problem we will use to computing bad cost matrices satisfies $\frac{p(p+1)}{2} + pr_0 \leq m$. This is exactly why for the trust-region subproblem that will be used in the upcoming two examples we set the variables $n = 3$, $m = 2$, $p = 1$ and $r_0 = 1$.

Also, since $m = 2$ refers to the choice to set the number of constraints of the SDP to two, one needs to have two constraints for Problem (TRS). In the upcoming two examples, we will do this by setting the last coefficient of x to 0 and $1/2$, respectively.

Now, the above choice of variables leads to the following problem description:

$$\begin{aligned} \text{minimize} \quad & x^\top Ax + 2b^\top x + c \\ \text{such that} \quad & ||x||^2 = 1, \\ & g(x) = k \end{aligned}$$

where $g : \mathbb{R}^3 \to \mathbb{R}$ represents a second constraint to satisfy $m = 2$.

Note that we need to compute the gradient and Hessian of the objective function of Problem (TRS-SDP) and Problem (TRS-BM) later on, specifically for Step 3 of our step-by-step approach[1]. Also, it is important to mention here that the gradient and Hessian can only be calculated when we consider x in terms of its variable coefficients. Therefore it is now convenient to express x, A, b and c into variables in order to compute the objective functions of its SDP and Burer-Monteiro factorization, from one can easily obtain the gradient and Hessian.

Following this line of thought, since $n = 3$, we can denote

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3, \quad A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} \in \mathbb{R}^{3\times 3}, \quad \text{and} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \in \mathbb{R}^3,$$

[1] See Chapter 2.

for computational use. It should be pointed out that A is symmetric, since we want the cost matrix C occurring in Problem (TRS-SDP) to be symmetric[2]. Following the construction of X described in (4.1), together with the Burer-Monteiro factorization $X = VV^\top$ leads to

$$X = \begin{pmatrix} xx^\top & x \\ x^\top & 1 \end{pmatrix} = \begin{pmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1 \\ x_2x_1 & x_2^2 & x_2x_3 & x_2 \\ x_3x_1 & x_3x_2 & x_3^2 & x_3 \\ x_1 & x_2 & x_3 & 1 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \begin{pmatrix} v_1 & v_2 & v_3 & v_4 \end{pmatrix} = VV^\top, \tag{4.2}$$

$$\text{and} \quad C = \begin{pmatrix} A & b \\ b^\top & c \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{12} & a_{22} & a_{23} & b_2 \\ a_{13} & a_{23} & a_{33} & b_3 \\ b_1 & b_2 & b_3 & c \end{pmatrix} \tag{4.3}$$

where we can see from Problem (TRS-BM) that $V_1 = \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}^\top \in \mathbb{R}^3$ and $\mathrm{v}_2 = v_4 \in \mathbb{R}$.

Next, we compute the objective functions of Problem (TRS-SDP) and Problem (TRS-BM), which are given by

$$\begin{aligned} \langle C, X \rangle &= \mathrm{Trace}(C^\top X) \\ &= a_{11}x_1^2 + 2a_{12}x_1x_2 + 2a_{13}x_1x_3 + a_{22}x_2^2 + 2a_{23}x_2x_3 \\ &\quad + a_{33}x_3^2 + 2b_1x_1 + 2b_2x_2 + 2b_3x_3 + c \end{aligned}$$

$$\begin{aligned} \langle CV, V \rangle &= \mathrm{Trace}(C^\top VV^\top) \\ &= a_{11}v_1^2 + 2a_{12}v_1v_2 + 2a_{13}v_1v_3 - 2a_{11}v_1v_4 \\ &\quad + a_{22}v_2^2 + 2a_{23}v_2v_3 - 2a_{12}v_2v_4 + a_{33}v_3^2 - 2a_{13}v_3v_4 + cv_4^2 \end{aligned}$$

From here, we can easily work out the gradient and Hessian of $\langle C, X \rangle$, denoted by $\nabla\langle C, X \rangle$ and $\mathrm{Hess}\langle C, X \rangle$, respectively.

$$\nabla\langle C, X \rangle = 2 \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + b_1 \\ a_{12}x_1 + a_{22}x_2 + a_{23}x_3 + b_2 \\ a_{13}x_1 + a_{23}x_2 + a_{33}x_3 + b_3 \end{pmatrix} \tag{4.4}$$

$$\mathrm{Hess}\langle C, X \rangle = 2 \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix} = 2A \tag{4.5}$$

Also, from the objective function $\langle CV, V \rangle$ the gradient and Hessian can be computed, which results in

$$\nabla\langle CV, V \rangle = 2 \begin{pmatrix} a_{11}v_1 + a_{12}v_2 + a_{13}v_3 - a_{11}v_4 \\ a_{12}v_1 + a_{22}v_2 + a_{23}v_3 - a_{12}v_4 \\ a_{13}v_1 + a_{23}v_2 + a_{33}v_3 - a_{13}v_4 \\ -a_{11}v_1 - a_{12}v_2 - a_{13}v_3 + cv_4 \end{pmatrix} \tag{4.6}$$

$$\mathrm{Hess}\langle CV, V \rangle = 2 \begin{pmatrix} a_{11} & a_{12} & a_{13} & -a_{11} \\ a_{12} & a_{22} & a_{23} & -a_{12} \\ a_{13} & a_{23} & a_{33} & -a_{13} \\ -a_{11} & -a_{12} & -a_{13} & -c \end{pmatrix} \tag{4.7}$$

[2]Namely, by the definition of an SDP (Definition 1) C has to be symmetric.

Now we have computed the necessary formulas for applying Step 3, we can look at specific examples. In the upcoming examples, we will look for the notorious bad C's. By Definition 7, these C's result in the fact that both $\langle C, X\rangle$ and $\langle CV, V\rangle$ satisfy second-order optimality conditions, and Problem (TRS-SDP) has X as its global minimizer, while the corresponding V is a non-optimal point of Problem (TRS-BM).

Example 1. $n = 3, \quad m = 2, \quad p = 1, \quad r_0 = 1$. Find $(A, b, c) \in \mathbb{R}^{3\times 3} \times \mathbb{R}^3 \times \mathbb{R}$, for a chosen $x \in \mathbb{R}^3$, as in Problem (TRS-EX1) below, such that the Burer-Monteiro approach does not lead to a global optimum.

$$\begin{aligned} \text{minimize} \quad & x^\top Ax + 2b^\top x + c \\ \text{such that} \quad & ||x||^2 = 1, \\ & x_3 = 0, \end{aligned} \tag{TRS-EX1}$$

where we minimize over $x \in \mathbb{R}^3$.

By the above information, it is not difficult to see that the corresponding SDP-relaxation and Burer-Monteiro factorization of the above problem are equivalent to Problem (TRS-SDP) and Problem (TRS-BM), respectively, where we add the extra constraint $x_3 = 0$.

Step 1 As we explained in the subsection above, the inequality $\frac{p(p+1)}{2} + pr_0 \leq m$ is indeed satisfied when we choose $p = 1$, $m = 2$ and $r_0 = 1$.

Step 2 We will fix a feasible $X_0 \in \mathcal{C}$ that has rank $r_0 = 1$ by first fixing $x \in \mathbb{R}^3$. For this example, we will choose $x = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}^\top = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^\top$.

This x is indeed feasible and will lead to

$$X_0 = \begin{pmatrix} xx^\top & x \\ x^\top & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}^\top = V_0 V_0^\top \tag{4.8}$$

as proposed in (4.1). We confirm that indeed $\operatorname{rank}(X_0) = 1 = r_0$ and X_0 is positive semidefinite since its eigenvalues are $\lambda = 0$ (multiplicity 3) and $\lambda = 2$. For this choice of V_0, the constraints given in Problem (TRS-BM) are trivially satisfied. Hence, both X_0 and V_0 are feasible for Problem (TRS-SDP) and Problem (TRS-BM), respectively.

Step 3 According to Step 3 (again, we refer to the method proposed in Chapter 2) we now need to construct C such that the SDP of Problem (TRS-EX1) has X_0 and V_0 are second-order critical points.

First, it is necessary for X_0 and V_0 to satisfy the first-order optimality conditions, as defined in Definition 5.[3] Therefore, one needs to make sure that both $\nabla\langle C, X_0\rangle = \nabla\langle CV_0, V_0\rangle = 0$. This can be done by simply substituting $x = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^\top$ and $V_0 = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}^\top$ into (4.4) and (4.6), respectively. Now, one obtains

$$\nabla\langle C, X_0\rangle = 2\begin{pmatrix} a_{11} + b_1 \\ a_{12} + b_2 \\ a_{13} + b_3 \end{pmatrix}, \quad \text{and} \quad \nabla\langle CV_0, V_0\rangle = 2\begin{pmatrix} 0 \\ 0 \\ 0 \\ -a_{11} + c \end{pmatrix}. \tag{4.9}$$

[3] The reasoning behind this can be found in the description below Step 3 in Chapter 2.

To automatically satisfy these first-order optimality conditions, the suggested choice of variables for b and c in (4.3) is as follows:

$$\begin{aligned} b_1 &= -a_{11} \\ b_2 &= -a_{12} \qquad \text{and} \qquad c = a_{11}. \\ b_3 &= -a_{13} \end{aligned} \tag{4.10}$$

Namely, if we incorporate these equalities into our construction of C, it implies that for all C's we construct, X_0 and V_0 will satisfy first-order optimality conditions.

In particular, we now know that C has to be of the following form (by substituting the equations given in (4.10) into (4.3)):

$$C = \begin{pmatrix} a_{11} & a_{12} & a_{13} & -a_{11} \\ a_{12} & a_{22} & a_{23} & -2a_{12} \\ a_{13} & a_{23} & a_{33} & -2a_{13} \\ -a_{11} & -2a_{12} & -2a_{13} & a_{11} \end{pmatrix}. \tag{4.11}$$

Now, still following Step 3 of our step-by-step approach, the bad C also has to satisfy other assumptions regarding second-order optimality conditions. Therefore, it is required (as explained in the text below Step 3 in Chapter 2) that $\text{Hess}\langle C, X_0\rangle \succeq 0$ and $\text{Hess}\langle CV_0, V_0\rangle \succeq 0$.

Besides that, from equations (4.5) and (4.7) we observe that both $\text{Hess}\langle C, X_0\rangle$ and $\text{Hess}\langle CV_0, V_0\rangle$ depend only on values of A. Hence, no substitution of coefficients of X_0 and V_0 is needed here. Instead, in order to come up with concrete matrices for C, we will now directly search for cost matrices that are of the form of (4.11) and for which $\text{Hess}\langle C, X_0\rangle \succeq 0$ and $\text{Hess}\langle CV_0, V_0\rangle \succeq 0$. This brings us to the last step – Step 4.

Step 4 It is not difficult to find numerous C's for which both of these constraints are satisfied. Below we provide a list of candidates of bad C's, together with the corresponding values of the determinants of $\text{Hess}\langle C, X\rangle$ and $\text{Hess}\langle CV, V\rangle$. In Appendix A the reader can find a detailed explanation of how Excel was programmed and used to deliberately search for the cost matrices listed below.

$$C_0 = \begin{pmatrix} -0,07723033 & -2 & -4 & 0,07723033 \\ -2 & -1 & 2 & 2 \\ -4 & 2 & 2 & 4 \\ 0,07723033 & 2 & 4 & -0,07723033 \end{pmatrix},$$

$$C_1 = \begin{pmatrix} -0,07723033 & -2,5 & -4 & 0,07723033 \\ -2,5 & -1 & 2 & 2,5 \\ -4 & 2 & 2 & 4 \\ 0,07723033 & 2,5 & 4 & -0,07723033 \end{pmatrix},$$

$$C_2 = \begin{pmatrix} -0,07723033 & -2,5 & -4 & 0,07723033 \\ -2,5 & -1,5 & 2 & 2,5 \\ -4 & 2 & 2 & 4 \\ 0,07723033 & 2,5 & 4 & -0,07723033 \end{pmatrix},$$

$$C_3 = \begin{pmatrix} -0,07723033 & -2,5 & -4 & 0,07723033 \\ -2,5 & -1,5 & 2,5 & 2,5 \\ -4 & 2,5 & 2 & 4 \\ 0,07723033 & 2,5 & 4 & -0,07723033 \end{pmatrix},$$

$$C_4 = \begin{pmatrix} -0,07723033 & -2,5 & -4 & 0,07723033 \\ -2,5 & -1,5 & 2,5 & 2,5 \\ -4 & 2,5 & 1,5 & 4 \\ 0,07723033 & 2,5 & 4 & -0,07723033 \end{pmatrix},$$

$$C_5 = \begin{pmatrix} -0,07723033 & -2,5 & -4 & 0,07723033 \\ -2,5 & -1,5 & 2,5 & 2,5 \\ -4 & 2,5 & 1 & 4 \\ 0,07723033 & 2,5 & 4 & -0,07723033 \end{pmatrix},$$

$$C_6 = \begin{pmatrix} -0,07723033 & -2,5 & -3 & 0,07723033 \\ -2,5 & -1,5 & 2,5 & 2,5 \\ -3 & 2,5 & 1 & 3 \\ 0,07723033 & 2,5 & 3 & -0,07723033 \end{pmatrix},$$

where the values for the determinants of the Hessians correspond to the following:

$$\begin{aligned}
\det(\mathrm{Hess}\langle C_0, X_0\rangle) &= 232,707056, & \det(\mathrm{Hess}\langle C_0V_0, V_0\rangle) &= 100,000011;\\
\det(\mathrm{Hess}\langle C_1, X_0\rangle) &= 351,707056, & \det(\mathrm{Hess}\langle C_1V_0, V_0\rangle) &= 108,649808;\\
\det(\mathrm{Hess}\langle C_2, X_0\rangle) &= 416,324898, & \det(\mathrm{Hess}\langle C_2V_0, V_0\rangle) &= 128,611637;\\
\det(\mathrm{Hess}\langle C_3, X_0\rangle) &= 497,715044, & \det(\mathrm{Hess}\langle C_3V_0, V_0\rangle) &= 153,754788;\\
\det(\mathrm{Hess}\langle C_4, X_0\rangle) &= 522,251662, & \det(\mathrm{Hess}\langle C_4V_0, V_0\rangle) &= 161,334673;\\
\det(\mathrm{Hess}\langle C_5, X_0\rangle) &= 546,78828, & \det(\mathrm{Hess}\langle C_5V_0, V_0\rangle) &= 168,914557;\\
\det(\mathrm{Hess}\langle C_6, X_0\rangle) &= 362,78828, & \det(\mathrm{Hess}\langle C_6V_0, V_0\rangle) &= 112,073034.
\end{aligned}$$

As we stated before, both $\nabla\langle C_i, X_0\rangle = 0$ and $\det(\mathrm{Hess}\langle C_i, X_0\rangle) \geq 0$ (where the latter guarantees that $\mathrm{Hess}\langle C_i, X_0\rangle$ is positive semidefinite) from which it follows that X_0 is a

minimum. Still, it is too soon to conclude whether this is locally or globally, which will be determined in Step 5.

Also, notice that the above list of C's can be extended by slightly increasing or decreasing the values of the entries, for which both $\det\Big(\text{Hess}\langle \hat{C}, X_0\rangle\Big)$ and $\det\Big(\text{Hess}\langle \hat{C}V_0, V_0\rangle\Big)$ will still be positive, and where $\hat{C}$ is the cost matrix with (slightly) perturbed entries.

Step 5 We will end this example by proposing how to determine which of the above cost matrices indeed lead to convergence failure of the Burer-Monteiro factorization algorithm. The latter is the case when X_0 is the unique global maximum of the SDP for all the computed cost matrices, and $V_0 = \begin{pmatrix}1 & 0 & 0 & 1\end{pmatrix}^\top$ is a local minimizer (hence, not the global minimizer) of the factorized problem.

First, we will show that X_0 is the unique global-optimal second-order critical point of the SDP of Problem (TRS-EX1). For this, consider the set of feasible X's

$$\mathcal{C} = \left\{ X = \begin{pmatrix} x_1^2 & x_1x_2 & x_1x_3 & x_1 \\ x_1x_2 & x_2^2 & x_2x_3 & x_2 \\ x_1x_3 & x_2x_3 & x_3^2 & x_3 \\ x_1 & x_2 & x_3 & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4} : \ \text{Tr}(X_{1:3,1:3}) = 1, \ X_{n+1,n+1} = 1, \ x_3 = 0 \right\},$$

where the constraint $X \succeq 0$ can be omitted, since any matrix X this form, always has determinant 0, and is therefore positive semidefinite. We can further simplify notation by defining

$$\mathcal{C} = \left\{ X = \begin{pmatrix} x_1^2 & x_1x_2 & 0 & x_1 \\ x_1x_2 & x_2^2 & 0 & x_2 \\ 0 & 0 & 0 & 0 \\ x_1 & x_2 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4} : \ x_1^2 + x_2^2 = 1 \right\}.$$

From here, we consider the set

$$\mathcal{C}_{\text{crit}} = \{X \in \mathcal{C} : \ \nabla\langle C, X\rangle = 0, \ \text{Hess}\langle C, X\rangle \succeq 0\},$$

which are all critical feasible matrices of the SDP.

By construction of the potentially bad cost matrices above, $\text{Hess}\langle C, X\rangle \succeq 0$ will always be satisfied, since this constraint does not depend on the choice of X, which we recall from (4.5). Now, considering (4.4), we have that

$$\nabla\langle C, X\rangle = 0 \qquad \Rightarrow \qquad \left\{ \begin{matrix} a_{11}x_1 + a_{12}x_2 + b_1 = 0 \\ a_{12}x_1 + a_{22}x_2 + b_2 = 0 \\ a_{13}x_1 + a_{23}x_2 + b_3 = 0 \end{matrix} \right\}. \tag{4.12}$$

From Figure 4.1 below (where the horizontal axis represents x_1 and the vertical axis x_2) it can be read off that X_0 (with $x_1 = 1$, $x_2 = 0$) is actually the only element of $\mathcal{C}_{\text{crit}}$ and is therefore the solution of the SDP of the trust-region subproblem considered in this example. By Figure 4.1, as well as it follows from complementary slackness (Proposition 2), we conclude that X_0 is the *unique* solution.

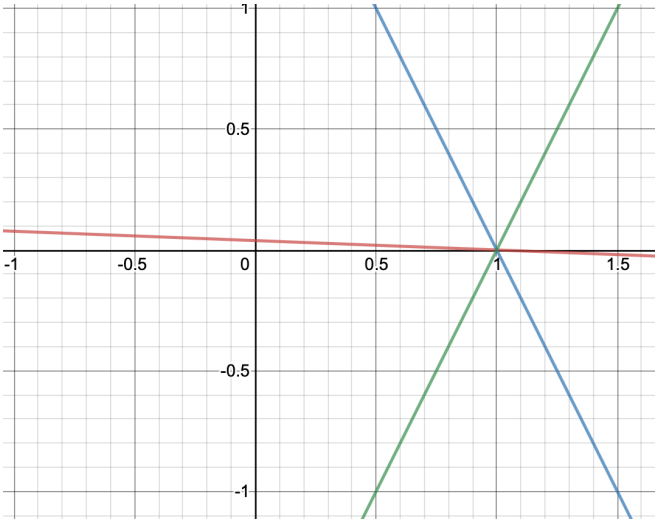


Figure 4.1: Graph of the constraints given in (4.14).

Lastly, we need to show that the corresponding V_0 is a non-optimal second-order critical point of the factorized problem of (TRS-EX1). We will do this by showing that there exists a feasible second-order critical V_1 such that $\langle CV_1, V_1\rangle < \langle CV_0, V_0\rangle$, from which it follows that V_0 cannot be the global minimum.

In order to find such a second-order critical V_1 that is feasible and satisfies $\langle CV_1, V_1\rangle < \langle CV_0, V_0\rangle$, we will first determine the set $\mathcal{M}_{\text{crit}}$ of all $V \in \mathcal{M}$ such that V is second-order critical. Here, C is any of the cost matrices computed above.

For this particular problem all feasible matrices V's are represented by the set

$$\mathcal{M} = \{V = \begin{pmatrix} v_1 & v_2 & v_3 & v_4 \end{pmatrix}^\top \in \mathbb{R}^4 : \left\| \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}^\top \right\| = 1,\ \|v_4\| = 1\},$$

which is equivalent to

$$\mathcal{M} = \{V = \begin{pmatrix} v_1 & v_2 & \pm\sqrt{1 - v_1^2 - v_2^2} & 1 \end{pmatrix}^\top \in \mathbb{R}^4 :\ |v_1|, |v_2| \leq 1\}. \tag{4.13}$$

Here, we used the fact that $\left\| \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix}^\top \right\| = 1$ implies $v_1^2 + v_2^2 + v_3^2 = 1$, and hence $v_3 = \pm\sqrt{1 - v_1^2 - v_2^2}$, where both $|v_1|$ and $|v_2|$ have to be smaller than or equal to 1. Also, we used that $\|v_4\| = 1$ yields $v_4 = 1$.

From here, we now look at the set $\mathcal{M}_{\text{crit}}$ of all feasible V's that are second-order critical:

$$\mathcal{M}_{\text{crit}} \coloneqq \{V \in \mathcal{M}_p :\ \nabla\langle CV, V\rangle = 0,\ \text{Hess}\langle CV, V\rangle \succeq 0\}.$$

From (4.6) we know that

$$\nabla\langle CV, V\rangle = 0 \qquad \Rightarrow \qquad \left\{ \begin{aligned} &a_{11}v_1 + a_{12}v_2 \pm a_{13}\sqrt{1 - v_1^2 - v_2^2} - a_{11} = 0 \\ &a_{12}v_1 + a_{22}v_2 \pm a_{23}\sqrt{1 - v_1^2 - v_2^2} - a_{12} = 0 \\ &a_{13}v_1 + a_{23}v_2 \pm a_{33}\sqrt{1 - v_1^2 - v_2^2} - a_{13} = 0 \\ &-a_{11}v_1 - a_{12}v_2 \pm a_{13}\sqrt{1 - v_1^2 - v_2^2} + c = 0 \end{aligned} \right\}. \tag{4.14}$$

These equalities can be simplified (by Gaussian elimination) to the equality

$$v_1^2 + v_2^2 + \left(\frac{(a_{12} + a_{13})v_1 + (a_{22} + a_{23})v_2 - a_{11} - a_{12} - a_{13} + c}{a_{23} + a_{33}} \right)^2 - 1 = 0. \tag{4.15}$$

Besides this, the condition $\text{Hess}\langle CV, V\rangle \succeq 0$ is invariant under changing X, regardless of the choice of V. Namely, this expression is fully dependent on the entries of C, as can be seen from (4.7). Since all candidates for a bad cost matrix satisfy this constraint by construction, all $V_1 \in \mathcal{M}$ that satisfy (4.15) equal the set $\mathcal{M}_{\text{crit}}$ (with $C = C_i$ for $i = 0, \ldots, 6$), and are therefore second-order critical points.

Now, as an example we will show that V_0 is a non-optimal point of the factorized problem by considering cost matrix $C = C_0$. Recalling that this particular cost matrix is given as

$$C_0 = \begin{pmatrix} -0,07723033 & -2 & -4 & 0,07723033 \\ -2 & -1 & 2 & 2 \\ -4 & 2 & 2 & 4 \\ 0,07723033 & 2 & 4 & -0,07723033 \end{pmatrix},$$

we compare its entries with (4.3) and substitute these into (4.15), which results in the equality

$$v_1^2 + v_2^2 + \left(\frac{-6v_1 + v_2 + 6}{4} \right)^2 - 1 = 0.$$

This is equivalent to solving

$$10v_1^2 + 1\frac{1}{4}v_2^2 - 18v_1 + 3v_2 - 3v_1v_2 + 8 = 0, \tag{4.16}$$

which represents an ellipsoid (minus all points for which $|v_1|, |v_2| > 1$ as known from (4.13)), and give us all $V \in \mathcal{M}_{\text{crit}}$ when considering C_0 as the cost matrix of the factorized problem. See Figure 4.2 below, where the horizontal axis represents values of v_1, and the vertical axis values of v_2.

Considering this figure, one can see that indeed V_0 (for which $v_1 = 1$ and v_2=0) is in $\mathcal{M}_{\text{crit}}$ as expected, since its entries satisfy (4.16).

To show that there exists $V \in \mathcal{M}_{\text{crit}}$ such that $\langle C_0V, V\rangle < \langle C_0V_0, V_0\rangle$, we could take $v_1 = 0,966$ and $v_2 = -0,2572$, which (approximately[4]) satisfies (4.16) and is therefore on the ellipsoid of Figure 4.2, which yields

$$V_1 = \begin{pmatrix} 0,966 \\ -0,2572 \\ 0 \\ 1 \end{pmatrix} \in \mathcal{M}_{\text{crit}}.$$

We observe that indeed $|v_1|, |v_2| \leq 1$ as (4.13) requires, and that this choice of v_1 and v_2 satisfies (4.14) with coefficients of C_0, considering (4.3).

[4]Here, we keep in mind the work of [Bhojanapalli et al. 2018; Pumir, Jelassi, and Boumal 2018; Cifuentes and Moitra 2019] that showed that points which approximately satisfy necessary optimality conditions are approximately optimal.

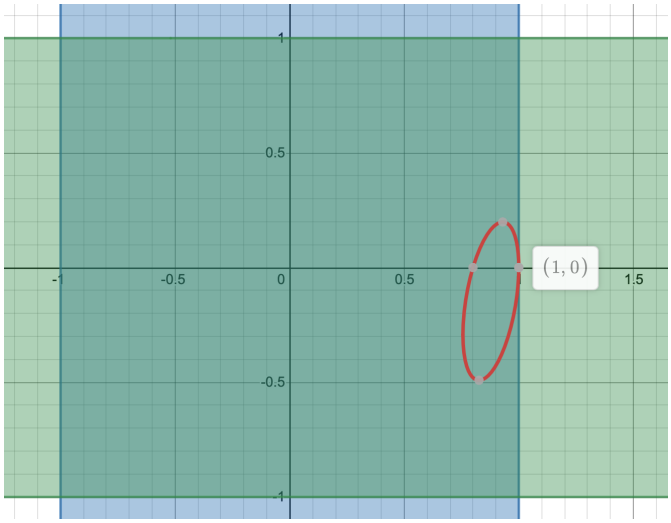


Figure 4.2: Graph of (4.16) (red), $|v_1| \leq 1$ (blue), $|v_2| \leq 1$ (green).

This choice of V results in $\langle C_0 V_1, V_1 \rangle \approx -0,1012$. Fortunately to us, this is indeed strictly smaller than $\langle C_0 V_0, V_0 \rangle = 0$. This confirms that V_0 is non-optimal second-order critical point of the factorized problem. In the same way, we could show that this is also the case for the other C's we computed above, i.e. V_0 is a non-optimal second-order critical point for C_i, where $i = 0, \ldots, 6$. Together with the fact that X_0 is the unique solution to the SDP of Problem (TRS-EX1), we recall Definition 7 to conclude that the C_i's we computed in this example are indeed cost matrices that lead to convergence failure of the Burer-Monteiro factorization algorithm.

Example 2. $n = 3, \quad m = 2, \quad p = 1, \quad r_0 = 1$. Find $(A, b, c) \in \mathbb{R}^{3\times 3} \times \mathbb{R}^3 \times \mathbb{R}$, for a chosen $x \in \mathbb{R}^3$, as in Problem (TRS-EX1) below, such that the Burer-Monteiro approach does not lead to a global optimum.

$$\begin{aligned} \text{minimize} \quad & x^\top A x + 2b^\top x + c \\ \text{such that} \quad & ||x||^2 = 1, \\ & x_3 = \frac{1}{2}, \end{aligned} \tag{TRS-EX2}$$

where we minimize over $x \in \mathbb{R}^3$.

By going over Example 1, we assume that the reader is already familiar with applying the procedure described in Chapter 2, and therefore in this example a less detailed explanation is provided when following our step-by-step approach of constructing bad C's.

Also, we note that it is not difficult to see that the corresponding SDP-relaxation and Burer-Monteiro factorization of the above problem are equivalent to Problem (TRS-SDP) and Problem (TRS-BM), respectively, where we add the extra constraint $x_3 = \frac{1}{2}$.

Step 1 For this step, we need to make sure that the inequality $\frac{p(p+1)}{2} + pr_0 \leq m$ is indeed satisfied. Note that (TRS-EX2) is almost identical to (TRS-EX1), but now the constraint on x_3 has changed a more realistic one, namely $x_3 = \frac{1}{2}$. Hence, the choice of variables n, m, p and r_0 is the same as in Example 1, and we confirm that the inequality above is indeed satisfied.

Step 2 Now, we fix a feasible $x \in \mathbb{R}^3$, such that the constraint $x_3 = \frac{1}{2}$ is satisfied, and such that the corresponding X_0 has $\text{rank}(X_0) = 1$. In this example we will choose $x = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}^\top = \begin{pmatrix} \frac{1}{2}\sqrt{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix}^\top$. Using (4.2), we obtain

$$X_0 = \begin{pmatrix} \frac{1}{2} & \frac{1}{4}\sqrt{2} & \frac{1}{4}\sqrt{2} & \frac{1}{2}\sqrt{2} \\ \frac{1}{4}\sqrt{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{4}\sqrt{2} & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2}\sqrt{2} & \frac{1}{2} & \frac{1}{2} & 1 \end{pmatrix} \quad \text{and} \quad V_0 = \begin{pmatrix} \frac{1}{2}\sqrt{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{pmatrix}.$$

The reader can verify that indeed $r_0 = \text{rank}(X_0) = 1$. Also, X_0 meets the positive semidefinite constraint of the SDP-form of (TRS-EX2), since its eigenvalues are $\lambda = 0$ and $\lambda = 2$.

Step 3 In this step we again first look for which coefficients of C the points X_0 and V_0 satisfy first-order optimality conditions. Therefore, one needs to make sure that both $\nabla\langle C, X_0\rangle = \nabla\langle CV_0, V_0\rangle = 0$. To obtain these properties, we substitute $x = \begin{pmatrix} \frac{1}{2}\sqrt{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix}^\top$ into (4.4) and (4.6), respectively, which yields

$$\nabla\langle C, X_0\rangle = \begin{pmatrix} a_{11}\sqrt{2} + a_{12} + a_{13} + 2b_1 \\ 2a_{12}\sqrt{2} + a_{22} + a_{23} + 2b_2 \\ a_{13}\sqrt{2} + a_{23} + a_{33} + 2b_3 \end{pmatrix}, \tag{4.17}$$

$$\nabla\langle CV_0, V_0\rangle = \begin{pmatrix} (\sqrt{2}-2)a_{11} + a_{12} + a_{13} \\ (\sqrt{2}-2)a_{12} + a_{22} + a_{23} \\ (\sqrt{2}-2)a_{13} + a_{23} + a_{33} \\ -a_{11}\sqrt{2} - a_{12} - a_{13} + 2c \end{pmatrix}. \tag{4.18}$$

When we set both of these gradients equal to 0, we obtain the following suggestion for variables for C as in (4.3):

$$\nabla\langle C, X_0\rangle = 0 \implies \begin{Bmatrix} b_1 = & -\frac{1}{2}a_{11}\sqrt{2} - \frac{1}{2}(a_{12} + a_{13}) \\ b_2 = & -\frac{1}{2}a_{12}\sqrt{2} - \frac{1}{2}a_{22} - \frac{1}{2}a_{23} \\ b_3 = & -\frac{1}{2}a_{13}\sqrt{2} - \frac{1}{2}a_{23} - \frac{1}{2}a_{33} \end{Bmatrix}, \tag{4.19}$$

$$\nabla\langle CV_0, V_0\rangle = 0 \implies \begin{Bmatrix} a_{11} = & \frac{2a_{12}+2a_{13}}{1-\frac{\sqrt{2}}{2}} \\ a_{22} = & \frac{2a_{12}+2a_{13}}{4-2\sqrt{2}} \\ a_{33} = & (2-\sqrt{2})a_{13} - a_{23} \\ c = & \frac{\sqrt{2}a_{11}+a_{12}+a_{13}}{2} \end{Bmatrix}. \tag{4.20}$$

In particular, we now know that every bad C has to be of the form such that the above equalities hold. When curious how exactly these requirements have been incorporated into C, we refer the reader to Appendix B, in which (by substituting (4.19) and (4.20) into (4.3)) the generating process of C's is explained in more detail.

After having satisfied first-order optimality conditions of X_0 and V_0, we now look for which specific C's it holds that X_0 is a second-order critical of Problem (TRS-SDP), keeping in mind that $x_3 = 1/2$. For that, it is necessary that $\text{Hess}\langle C, X\rangle \succeq 0$. Simultaneously, we also want these C's to result in a second-order critical V_0, for which it is needed that

$\text{Hess}\langle CV, V\rangle \succeq 0$. In the next step, we will directly search for cost matrices that meet both of these requirements, together with satisfying the equalities given in (4.19) and (4.20).

Step 4 Also, for this example, it is not difficult to find numerous C's that meet the above requirements. Below we provide a list of possible bad C's, together with the corresponding values of the determinants of $\text{Hess}\langle C, X\rangle$ and $\text{Hess}\langle CV, V\rangle$. In Appendix B the reader can find a detailed explanation of how Excel was programmed and used to deliberately search for the cost matrices listed below.

$$C_0 = \begin{pmatrix} -4205606,32 & -2523587,15 & 60000 & 4205606,32 \\ -2523587,15 & -1478285,1 & 2 & 2523587,15 \\ 6000 & 2 & 35145,1863 & -60000 \\ 4205606,32 & 2523587,15 & -60000 & -4205606,32 \end{pmatrix},$$

$$C_1 = \begin{pmatrix} -4196657,61 & -2518345,11 & 60000 & 4196657,61 \\ -2518345,11 & -1475214,4 & 2 & 2518345,11 \\ 60000 & 2 & 35145,1863 & -60000 \\ 4196657,61 & 2518345,11 & -60000 & -4196657,61 \end{pmatrix},$$

$$C_2 = \begin{pmatrix} -4192183,26 & -2515724,1 & 60000 & 4192183,26 \\ -2515724,1 & -1473679,1 & 2 & 2515724,1 \\ 60000 & 2 & 35145,1863 & -60000 \\ 4192183,26 & 2515724,1 & -60000 & -4192183,26 \end{pmatrix},$$

$$C_3 = \begin{pmatrix} -4187708,92 & -2513103,1 & 60000 & 4187708,92 \\ -2513103,09 & -1472143,7 & 2 & 2513103,09 \\ 60000 & 2 & 35145,1863 & -60000 \\ 4187708,92 & 2513103,09 & -60000 & -4187708,92 \end{pmatrix},$$

$$C_4 = \begin{pmatrix} -4188268,2 & -2513430,71 & 60000 & 4188268,2 \\ -2513430,7 & -1472335,6 & 2 & 2513430,71 \\ 60000 & 2 & 35145,1863 & -60000 \\ 4188268,2 & 2513430,71 & -60000 & -4188268,2 \end{pmatrix},$$

$$C_5 = \begin{pmatrix} -4187848,73 & -2513184,99 & 60000 & 4187848,73 \\ -2513184,99 & -1472191,7 & 2 & 2513184,99 \\ 60000 & 2 & 35145,1863 & -60000 \\ 4187848,73 & 2513184,99 & -60000 & -4187848,73 \end{pmatrix},$$

where the values for the determinants of the Hessians correspond to the following:

$$\begin{aligned}
&\det(\text{Hess}\langle C_0, X_0\rangle) = 105,756441, && \det(\text{Hess}\langle C_0V_0, V_0\rangle) = 1779079823;\\
&\det(\text{Hess}\langle C_1, X_0\rangle) = 263,841906, && \det(\text{Hess}\langle C_1V_0, V_0\rangle) = 4429016552;\\
&\det(\text{Hess}\langle C_2, X_0\rangle) = 8,78557688, && \det(\text{Hess}\langle C_2V_0, V_0\rangle) = 147322993;\\
&\det(\text{Hess}\langle C_3, X_0\rangle) = 175,528473, && \det(\text{Hess}\langle C_3V_0, V_0\rangle) = 2940248607;\\
&\det(\text{Hess}\langle C_4, X_0\rangle) = 193,106491, && \det(\text{Hess}\langle C_4V_0, V_0\rangle) = 3235127099;\\
&\det(\text{Hess}\langle C_5, X_0\rangle) = 61,4369675, && \det(\text{Hess}\langle C_5V_0, V_0\rangle) = 1029154905;
\end{aligned}$$

Step 5 Considering the space of feasible matrices of the SDP of Problem (TRS-EX2), we have that

$$\mathcal{C} = \left\{ X = \begin{pmatrix} x_1^2 & x_1x_2 & \frac{1}{2}x_1 & x_1 \\ x_1x_2 & x_2^2 & \frac{1}{2}x_2 & x_2 \\ \frac{1}{2}x_1 & \frac{1}{2}x_2 & \frac{1}{4} & \frac{1}{2} \\ x_1 & x_2 & \frac{1}{2} & 1 \end{pmatrix} : \; x_1^2 + x_2^2 + \frac{1}{4} = 1 \right\},$$

from which it follows that the set of all feasible critical points is

$$\mathcal{C}_{\text{crit}} = \{X \in \mathcal{C} : \; \langle C, X\rangle = 0, \; \text{Hess}\langle C, X\rangle \succeq 0\}.$$

Now, as an example we pick $C = C_0$ for which $\mathcal{C}_{\text{crit}}$ equals all $X \in \mathcal{C}$ such that

$$\left\{ \begin{matrix} -4205606,32x_1 - 2523587,15x_2 + \frac{1}{2} \cdot 60000 + 4205606,32 = 0 \\ -2523587,15x_1 - 1475214,4x_2 + \frac{1}{2} \cdot 2 + 2518345,11 = 0 \\ 60000x_1 + 2x_2 + \frac{1}{2} \cdot 35145,1863 - 60000 = 0 \end{matrix} \right\} \tag{4.21}$$

are satisfied, which can be seen from comparing the entries of C_0 with (4.3) and implementing these into (4.4). From Figure 4.3 (where the horizontal axis denotes values of x_1, and the vertical axis represents values of x_2) one can see this is only the case when $x_1 \approx \frac{1}{2}\sqrt{2}$, $x_2 \approx \frac{1}{2}$. Therefore we conclude that X_0 is the only element of $\mathcal{C}_{\text{crit}}$ and is therefore the unique global solution of the SDP of Problem (TRS-EX2) with $C = C_0$. Similarly, the same result can be shown for $C = C_i$, where $i = 1, \ldots, 5$.

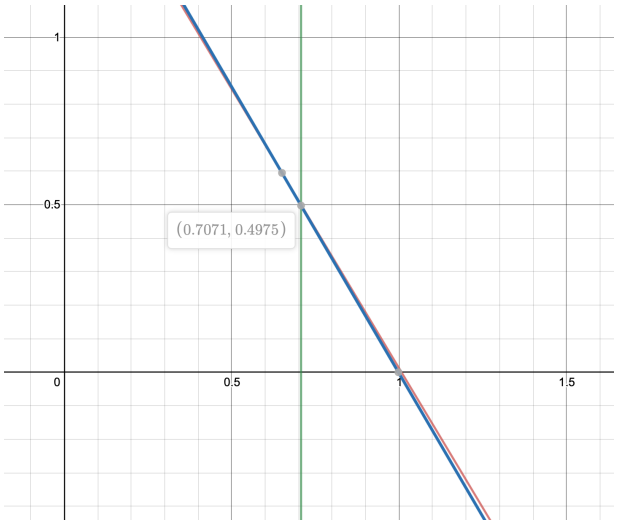


Figure 4.3: Graph of the constraints given in (4.21).

Finally, we will will come up with a $V_1 \in \mathcal{M}_{\text{crit}}$ (where the latter denotes the set of $V_1 \in \mathcal{M}$ such that this V_1 is second order critical) such that $\langle CV_1, V_1\rangle < \langle CV_0, V_0\rangle$. If there indeed exists such a V_1, it follows that V_0 is a non-optimal second-order critical point.

Now, we can use the result of Step 5 of Example 1 that a feasible V is only second-order critical for the factorized problem of Problem (TRS-EX2) (using the cost matrices computed above, for which $\text{Hess}\langle CV, V\rangle \succeq 0$ regardless of the choice of V) if and only if

(4.15) holds. Comparing (4.3) with C_0, we substitute its values into (4.15) which results in

$$v_1^2 + v_2^2 + \left(\frac{-2463587,15v_1 - 1478283,1v_2 + 2463587,15}{-1478283,1} \right)^2 - 1 = 0. \tag{4.22}$$

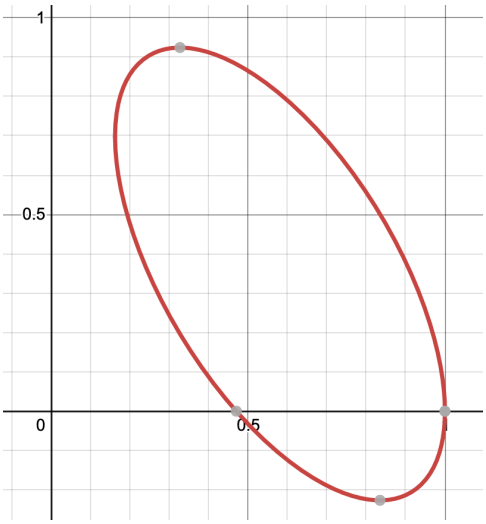


Figure 4.4: Graph of (4.22)

From Figure 4.4 (where the horizontal axis represents v_1 and the vertical axis v_2) it can be seen that $V_0 = \begin{pmatrix} \sqrt{2}/2 & 1/2 & 1/2 & 1 \end{pmatrix}^\top$ is indeed (approximately) an element of $\mathcal{M}_{\text{crit}}$, since it is a point on the ellipse of Figure 4.4. Moreover, we can now pick

$$V_1 = \begin{pmatrix} 0,8352 \\ -0,2265 \\ 0,5 \\ 1 \end{pmatrix} \in \mathcal{M}_{\text{crit}},$$

for which $\langle C_0 V_1, V_1 \rangle = -379541,1$ which is indeed strictly smaller than $\langle C_0 V_0, V_0 \rangle = 0$. This shows that V_0 is a non-optimal second-order critical point for the factorized problem of Problem (TRS-EX2) when $C = C_0$. Similarly, the same result can be shown for $C = C_i$, where $i = 1, \ldots, 5$. From this, it follows that the potential bad cost matrices computed above indeed lead to convergence failure of the Burer-Monteiro factorization algorithm.

Chapter 5

Conclusion

Discussion of the results

The main results of this thesis are the examples 1 and 2 given in Chapter 4, in which a list of potentially bad cost matrices was constructed, followed by an informal proof that showed that these cost matrices will indeed lead to convergence failure of the Burer-Monteiro factorization algorithm. Although literature such as [Burer and Monteiro 2003; Journée et al. 2010; Bandeira, Boumal, and Voroninski 2016a; Ge, Lee, and Ma 2018] state that the Burer-Monteiro factorization algorithm works when $p = r_0$, the results of this thesis emphasize that this choice of factorization rank only works in specific situations. Namely, both examples in Chapter 4 showed convergence failure of the Burer-Monteiro factorization algorithm, despite the fact that $p = r_0$. In particular, the Burer-Monteiro approach can be certified correct when specific assumptions on the cost matrix C are made. If not, it is not difficult to construct C's that will lead to convergence failure of the factorization algorithm, as shown in Example 1 and Example 2.

Also, Step 5 of both examples pointed out that (for the trust-region subproblem) it is important to take the gradient of the objective function of the SDP and its factorized problem into account when considering bad cost matrices. Specifically, the size of the set of second-order critical V mainly depends on this $n \times p$ matrix $\nabla\langle CV, V\rangle$, which has to equal the zero matrix. After this, second-order optimality is easily satisfied, since it does not depend on neither X nor V, and therefore for all cost matrices that are computed on the hand of Step 1-3, X and V are second-order critical if and only if they are first-order critical. Realizing this can drastically shorten the proof that a potentially bad cost matrix (that satisfies Step 3) indeed leads to convergence failure.

Lastly, with both examples that were given in Chapter 4 the research goal of generating a list of C's for which the Burer-Monteiro factorization algorithm cannot provide the global solution of the SDP-relaxation of the trust-region subproblem, is satisfied. Theorem 1 (the main result of [Waldspurger and Waters 2019]) was particularly important for this, so that we had a guarantee for existence of C's for which the V_0 that corresponded to the optimal solution of the SDP was non-optimal for the factorized problem. If one had no such guarantee, finding a V_1 that has lower objective function than V_0 would be based rather on luck than on a deliberate search. Besides that, the research goal of the current thesis, together with the results substantiated a more general statement provided by that same paper [Waldspurger and Waters 2019]: when $p \lesssim \sqrt{2m}$ Riemannian optimization algorithms cannot be certified correct.

Essential justification

Coming back to the choice of values for the parameters used in examples 1 and 2 of Chapter 4, the reader could argue that it would be valuable to consider large-scale SDPs, instead of restricting ourselves to smaller dimensional cases. Moreover, one could justly argue that the Burer-Monteiro approach is specially designed for large scale semidefinite programs, as mentioned in Chapter 1. This limitation to small scale cases of SDPs would indeed be inappropriate when the research goal of this Bachelor's thesis would have been something along the lines: *apply the Burer-Monteiro approach to real-life applications*. Connecting to this vision, it is appropriate to note that Chapter 3 mentioned that the step-by-step procedure we constructed also applies to the large-dimensional semidefinite programs.

On the other hand, the main goal of the current thesis was actually much smaller and more specific than that: discovering the procedure of how to set up examples for which the Burer-Monteiro approach does not converge and *actually computing cost matrices for which this is the case*. Although the procedure of computing these cost matrices would have been the same when the dimension of X_0 would have been much larger, it would not be doable to compute essential features for examples 1 and 2 (such as the gradients and Hessians of the objective functions) by hand, let alone showing global optimality of X_0 and non-optimality of V_0. While for large scale semidefinite programs, this would have required a significant amount of programming, it would not have contributed to the main goal of this thesis, nor the understanding of the Burer-Monteiro algorithm. Merely, it would add a tremendous amount of equalities for e.g. equations (4.4) and (4.6) that needed to be satisfied. Along with that, much more time would have been spended on programming MatLab and Excel, which again does not provide a better understanding the procedure of producing bad C's (compared to restricting oneself to only small scale SDPs).

Further research

In this current Bachelor's thesis, we generated cost matrices (see Chapter 4 for which the Burer-Monteiro approach does not work. This "not working" was based on the fact that, for bad cost matrices, the global optimal solution X of an particular SDP corresponded to a non-global optimal solution of its Burer-Monteiro factorization, using Proposition 1 and Corollary 1. Using this specific property of bad C's, we used Theorem 1 as a suggestion how to set up the examples of convergence failure and to actually generate bad cost matrices. Hence, when the Burer-Monteiro factorization algorithm would be applied, convergence failure (Definition 8) would be the case.

Following this reasoning, the close reader would note that this way of proving convergence failure is rather indirect. When one prefers a direct proof, it would have been appropriate to run the Burer-Monteiro factorization algorithm using some bad cost matrices for the corresponding trust-region subproblems given in Chapter 4. In particular, as Burer and Monteiro proposed in their early findings [Burer and Monteiro 2003] one could use two conventional optimization algorithms for this (together): Augmented Lagrangian and Dual Ascent (respectively) (as already noted in Chapter 2).

Since the main goal of this thesis was to compute bad C's we used the indirect approach rather than the direct approach to show that these cost matrices are indeed bad, simply because proving "badness" of cost matrices is much more efficient when using the former (indirect) approach instead of the latter. Nevertheless, proving convergence failure in the direct way

would have been a powerful extension of the paper of [Waldspurger and Waters 2019], since it allows one to step from theory into a more practical setting. Therefore, a useful proceeding of this current Bachelor's thesis is to use Augmented Lagrangian and Dual Ascent to show that when one uses the bad cost matrices we computed, the Burer-Monteiro factorization algorithm indeed converges to a different V_{opt} than the V_0 that was fixed in examples 1 and 2.

Also, as we saw in examples 1 and 2, it is not difficult to generate lists of bad C's for the trust-region subproblem of [Bandeira, Boumal, and Voroninski 2020, Section 5.2]. In this particular section of Boumal et al. it was stated that for $p \geq 2$ and for *almost* all (A, b, c), second-order critical points of Problem (TRS-BM) are optimal. Now, since the set-up for the examples in Chapter 4 will not significantly differ from the case where $p = 2$ (instead of $p = 1$), it would be interesting and worth wile to generate bad C's for the former choice of p to test [Bandeira, Boumal, and Voroninski 2020, Theorem 1.4]. Moreover, this test of the main result of the latter paper would be even better substantiated when we compute not only the lists of bad cost matrices, but also the volume of the bad cost matrices in the space of feasible cost matrices. Namely, the in the recent literature the expression "for almost all" has been used quite often, where the only clarification of this use of language was in [Waldspurger and Waters 2019]. Computing the volume of bad C's (only if we knew how) in the total space of cost matrices would be a huge contribution to the rather vague use of language that "the Burer-Monteiro approach with a certain choice of factorization rank *works almost always*".

Acknowledgement

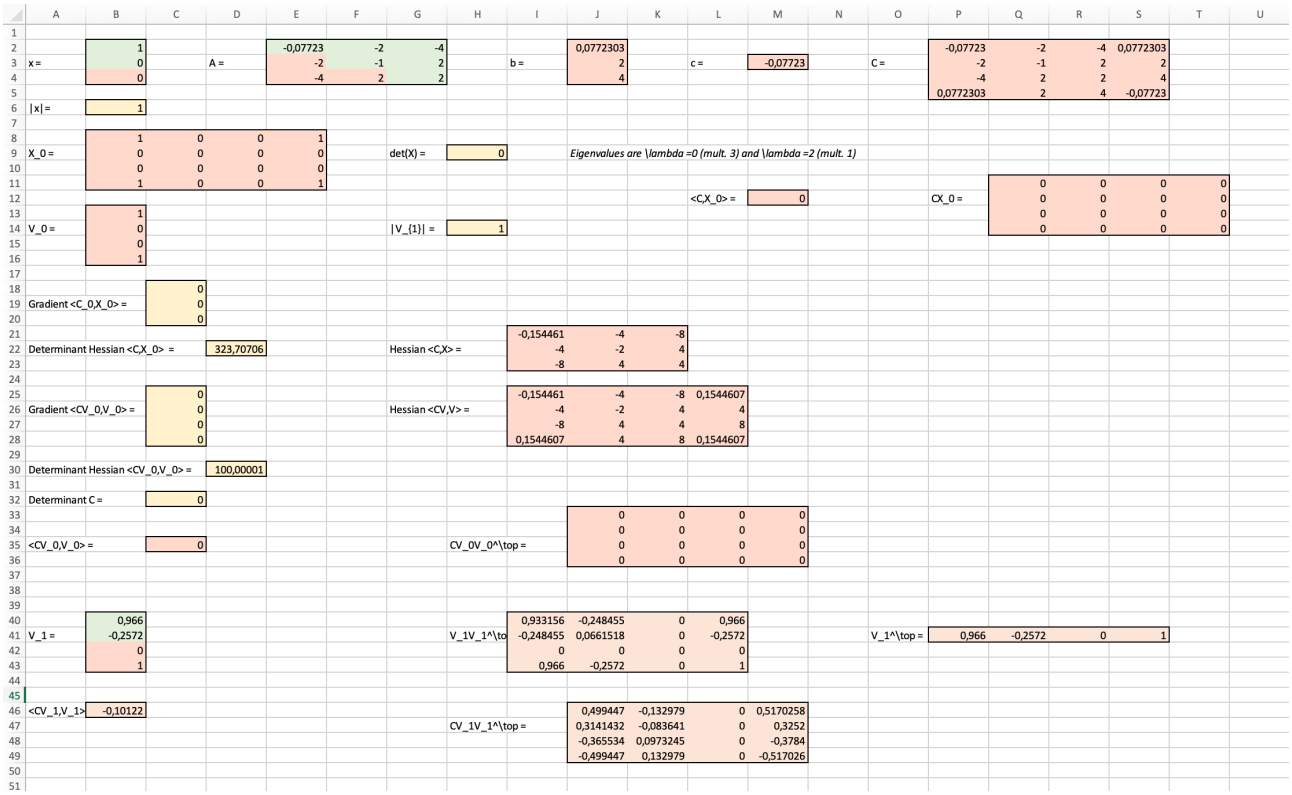
I would like to thank my supervisor Alden Waters, co-author of [Waldspurger and Waters 2019], where the latter article is one of the two main papers on which this thesis is built. Alden has given me the space in which I could formalize the research goals myself, which worked really motivating for me. Too much advice and accompaniment will not lead to a student mind that is eager to discover and try. Little advice and much accompaniment will lead to a valuable experience, from which a student learns not only to how to read scientific articles and writing his own mathematical texts, but is motivated and content to do so. Moreover, seeing improvement in understanding each time one devotes oneself to a new skill is the ultimate drive for gaining intellectual knowledge, even in times of a global pandemic.

Lastly, I am grateful to the second supervisor of this thesis, Fred Wubs, for his comments and pointing out mistakes.

Appendix A

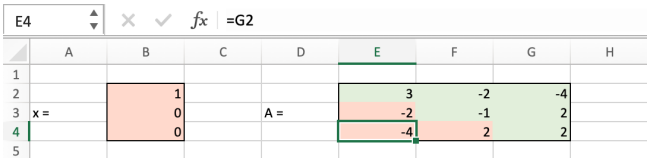
Generating bad C's for Example 1

As promised in Chapter 4, this appendix will explain how Excel was used for generating the cost matrices of Example 1 for which the Burer-Monteiro approach does not work. Here, the setup and implementation of constraints (such as (4.10)) will be discussed, together with screenshots of the Excel spreadsheet itself.



In the above spreadsheet, one can see the program we used for producing the cost matrix C_6 as given in Example 1. All red-coloured boxes are fixed values. For example, the red-coloured matrices on the left side represent x, X_0 and V_0. Since we fixed all of these matrices, all of the coefficients themselves are fixed, and therefore we will not change them when searching for bad cost matrices. Besides that, all yellow-coloured boxes are constraints that need to be satisfied, such as $\nabla\langle C, X_0\rangle = 0$ and $\nabla\langle CV_0, V_0\rangle = 0$ which are the yellow vectors indicated with `Gradient <C,X0> =` and `Gradient <CV0,V0> =`, respectively. Lastly, the green-coloured boxes are the values we are allowed to manipulate when searching for bad C's. For this program, we

are only allowed to change the diagonal and upper-diagonal values of A. Namely, since A must be symmetric (as C is a symmetric cost matrices according to Definition 1), the lower-diagonal values have to equal the corresponding upper-diagonal values, and hence these former values *depend* on the latter ones. We simply incorporated this by setting `E3=F2`, `E4=G2` and `F4=G3`, as can be seen in the screenshot below.



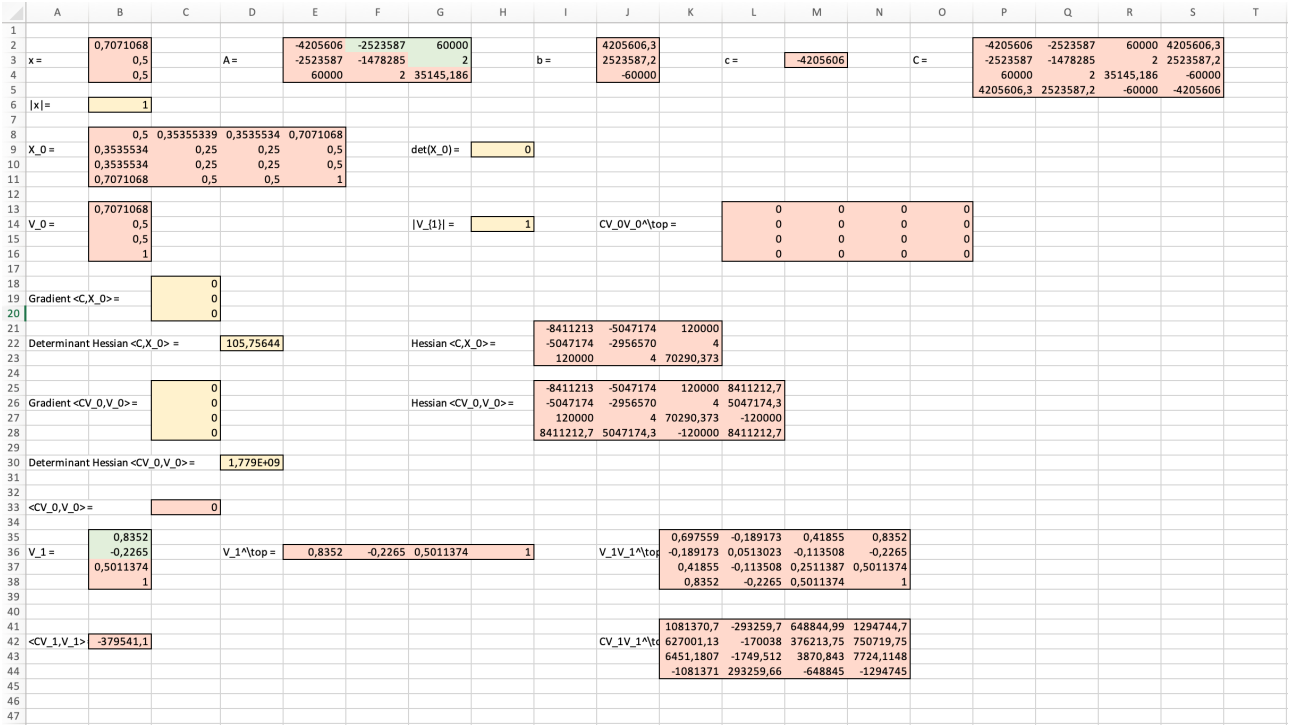
After this, we incorporated equation (4.10) by setting the values of b and c as follows: `J2=-E2`, `J3=-F2`, `J4=-G2` and `M3=E2`. By doing this, b and c now depend on the upper-diagonal values of A, and hence we will designate the colour red to the former vectors. Besides that, the gradients $\nabla\langle C, X_0\rangle$ and $\nabla\langle CV_0, V_0\rangle$ will also be implemented in the spreadsheet by setting the coefficients equal to the corresponding combinations of the values of `A`, `b` and `c` as proposed in (4.9). Note that by construction (`J2=-E2`, `J3=-F2`, `J4=-G2`, `M3=E2`) the gradients will always equal the zero vector, no matter how we change the green boxes (diagonal and upper-diagonal values of `A`). The Hessian matrices indicated by `Hessian <C,X0>` and `Hessian <CV0,V0>` are constructed in the same way by considering (4.5) and (4.7). The determinants of these matrices can be calculated easily by using the `MDETERM`-command, which is built in by default into Excel.

Now, since C consists of A, b and c as given in (4.3), we will set `P2=E2`, `Q2=F2`, `R2=G2`, `S2=J2`, *etc.* From here, considering the computationally undemanding choice of x this example, the bad matrices C were found by experimenting with the green boxes of the matrix `A`, and observing when the values of `Determinant Hessian <C,X0>` and `Determinant Hessian <CV0,V0>` were positive. In contrast with this, for Example 2 these determinant values were much harder to realize, as is explained in the next appendix.

Appendix B

Generating bad C's for Example 2

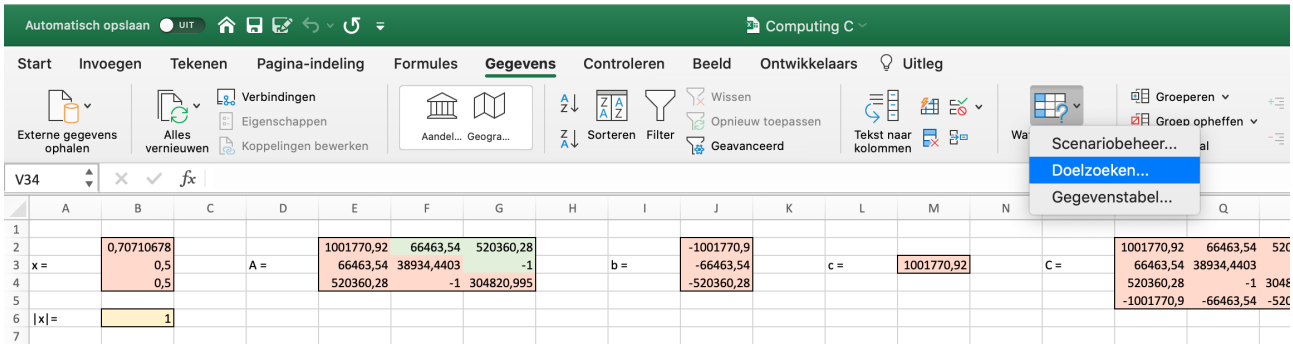
This last appendix justifies the generating process of bad cost matrices for Example 2 of Chapter 4. The overall setup corresponds to the one of Example 1 that was described in Appendix A.



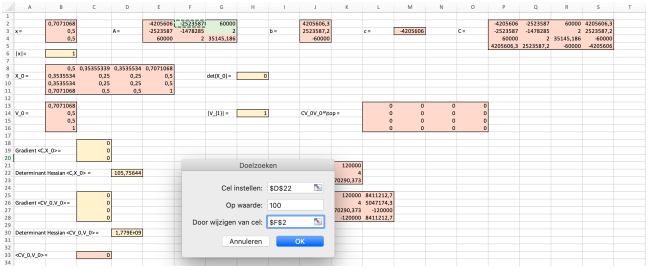
		0,7071068			-4205606	-2523587	60000			4205606,3					-4205606	-2523587	60000	4205606,3
x =		0,5		A =	-2523587	-1478285	2		b =	2523587,2		c =	-4205606	C =	-2523587	-1478285	2	2523587,2
		0,5			60000	2	35145,186			-60000					60000	2	35145,186	-60000
															4205606,3	2523587,2	-60000	-4205606
|x| =		1																
		0,5	0,35355339	0,3535534	0,7071068													
X_0 =		0,3535534	0,25	0,25	0,5		det(X_0) =	0										
		0,3535534	0,25	0,25	0,5													
		0,7071068	0,5	0,5	1													
		0,7071068										0	0	0	0			
V_0 =		0,5					|V_{1}| =	1		CV_0V_0^\top =		0	0	0	0			
		0,5										0	0	0	0			
		1										0	0	0	0			
			0															
Gradient <C,X_0> =			0															
			0															
									-8411213	-5047174	120000							
Determinant Hessian <C,X_0> =				105,75644			Hessian <C,X_0> =		-5047174	-2956570	4							
									120000	4	70290,373							
			0						-8411213	-5047174	120000	8411212,7						
Gradient <CV_0,V_0> =			0				Hessian <CV_0,V_0> =		-5047174	-2956570	4	5047174,3						
			0						120000	4	70290,373	-120000						
			0						8411212,7	5047174,3	-120000	8411212,7						
Determinant Hessian <CV_0,V_0> =				1,779E+09														
<CV_0,V_0> =			0															
		0,8352										0,697559	-0,189173	0,41855	0,8352			
V_1 =		-0,2265		V_1^\top =	0,8352	-0,2265	0,5011374	1		V_1V_1^\top		-0,189173	0,0513023	-0,113508	-0,2265			
		0,5011374										0,41855	-0,113508	0,2511387	0,5011374			
		1										0,8352	-0,2265	0,5011374	1			
												1081370,7	-293259,7	648844,99	1294744,7			
<CV_1,V_1>		-379541,1								CV_1V_1^\top		627001,13	-170038	376213,75	750719,75			
												6451,1807	-1749,512	3870,843	7724,1148			
												-1081371	293259,66	-648845	-1294745			

Since features such as the gradients given in (4.17) and (4.18), the equations (4.19) and (4.20) that followed from setting these gradients to 0, and the Hessians as in (4.5) and (4.7) are implemented in the same way as in Appendix A, we refer the reader to the latter appendix for explanation about implementation. Again, by construction (i.e. by satisfying the equalities given in (4.19) and (4.20)) of the matrix `A` the gradients indicated by `Gradient <C,X0> =` and `Gradient <CV0,V0> =` will always equal the zero vector.

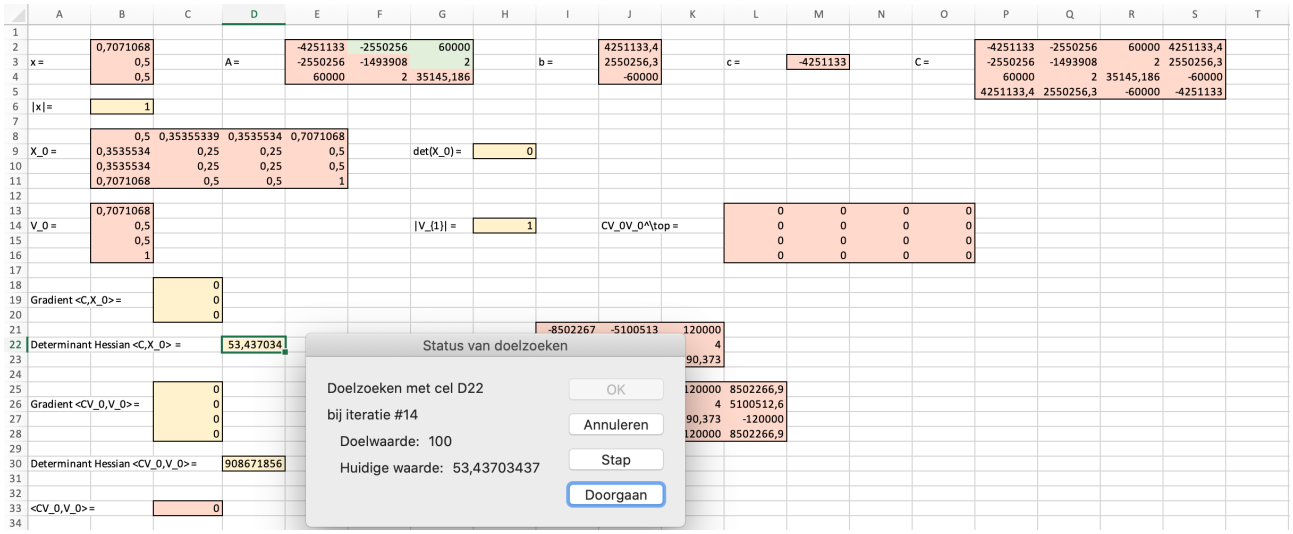
We will now directly turn to the generating process of bad cost matrices. Again, in the above spreadsheet, we allow ourselves to only change the values of the boxes in green, since all values in red-coloured boxes depend on these green-box coefficients. Using this, we apply the `Goal Seek`-feature of Excel, which can be found under the `Data` tab in the menu bar (see the screenshot of the Dutch Excel version below).



`Goal Seek` allows us to directly set one of the determinants of the Hessians equal to a desired positive or negative value by changing one of the upper-diagonal values of `A`. For example, in the figure below, we set `Determinant Hessian <C0,X0>` approximately equal to 100.



By setting the `Goal Seek`-command up like this, one lets Excel search for a value of `Determinant Hessian <C,X0>` (box `D22`) close to to 100 (or any other positive value, since we want $\det \mathrm{Hess}\langle C, X_0\rangle \succeq 0$) by changing the value of the upper-right coefficient of `A` (box `G2`), from which the result can be seen in the figure beneath. In the same way, we can deliberately set `Determinant Hessian <CV0,V0>` equal to some positive value (since we want $\det \mathrm{Hess}\langle CV_0, V_0\rangle \succeq 0$), if this was not already the case after applying `Goal Seek` to `Determinant Hessian <C,X0>`.



Now, since C consists of A, b and c as given in (4.3), the values of `C` depend on only `A` (since `b` and `c` also depend on solely values of `A`). After having used the `Goal Seek`-feature, we can jot down our `C` (and correspondingly C) for which it holds that `Gradient <C,X0>` is positive, while `Gradient <CV0,V0>` is negative, and so $\mathrm{Hess}\langle C, X_0\rangle \succeq 0$, together with $\mathrm{Hess}\langle CV_0, V_0\rangle \succeq 0$.

Bibliography

Bandeira, Afonso S., Nicolas Boumal, and Vladislav Voroninski (2016a). *On the low-rank approach for semidefinite programs arising in synchronization and community detection*. arXiv: 1602.04426 [math.OC].

— (2016b). *The non-convex Burer-Monteiro approach works on smooth semidefinite programs*. URL: https://papers.nips.cc/paper/6517-the-non-convex-burer-monteiro-approach-works-on-smooth-semidefinite-programs.

— (2020). "Deterministic Guarantees for Burer-Monteiro Factorizations of Smooth Semidefinite Programs". In: *Communications on Pure and Applied Mathematics* 73.3, pp. 581–608.

Barvinok, A.I. (1995). "Problems of distance geometry and convex properties of quadratic maps". In: *Discrete and Computational Geometry* 13.2, pp. 189–202.

Bhojanapalli, Srinadh et al. (2018). "Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form". In: *Proceedings of Machine Learning Research* 75, pp. 1–28.

Boumal, Nicolas (2016). *A Riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints*. arXiv: 1506.00575 [math.OC].

Burer, S. and R.D.C. Monteiro (2003). "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization". In: *Mathematical Programming* 95.2, pp. 427–444.

— (2005). "Local Minima and Convergence in Low-Rank Semidefinite Programming". In: *Mathematical Programming* 103.3, pp. 427–444.

Celis, M.R. (Sept. 1984). "A trust region strategy for nonlinear equality constrained optimization". PhD thesis. 6100 Main Street MS 134, Houston, Texas: Rice University.

Cifuentes, Diego (2020). *On the Burer-Monteiro method for general semidefinite programs*. arXiv: 1904.07147 [math.OC].

Cifuentes, Diego and Ankur Moitra (2019). *Polynomial time guarantees for the Burer-Monteiro method*. arXiv: 1912.01745 [math.OC].

Delorme, Charles and Svatopluk Poljak (1993). "Laplacian eigenvalues and the maximum cut problem". In: *Mathematical Programming* 62.1-3, pp. 557–574.

Ge, Rong, Jason D. Lee, and Tengyu Ma (2018). *Matrix Completion has No Spurious Local Minimum*. arXiv: 1605.07272 [cs.LG].

Goemans, Michel X. and David P. Williamson (1995). "Improved approximation algorithm for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM* 42.6, pp. 1115–1145.

Journée, M. et al. (2010). "Low-rank optimization for semidefinite convex problems". In: *SIAM Journal on Optimization* 20.5, pp. 2327–2351.

Pataki, G. (1998). "On the Rank of Extreme Matrices in Semidefinite Programs and the Multiplicity of Optimal Eigenvalues". In: *Mathematics of Operations Research* 23.2, pp. 339–358.

Pumir, Thomas, Samy Jelassi, and Nicolas Boumal (2018). *Smoothed analysis of the low-rank approach for smooth semidefinite programs*. arXiv: 1806.03763 [stat.ML].

Rosen, D.M. et al. (2019). "A certifiably correct algorithm for synchronization over the special Euclidean group". In: *International Journal of Robotics Research* 38.2-3, pp. 95–125.

Svatopluk Poljak, Franz Rendl (1995). "Nonpolyhedral Relaxations of Graph-Bisection Problems". In: *SIAM Journal on Optimization* 5.3, pp. 467–487.

Waldspurger, Irène and Alden Waters (2019). *Rank optimality for the Burer-Monteiro factorization*. arXiv: 1812.03046 [math.OC].