Numerical Optimization with Applications to Machine Learning

Nikitas Rontsis

Wadham College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2019

Acknowledgements

My deepest gratitude goes to my supervisor, Paul Goulart. I will be forever grateful for all the time Paul has spent on teaching me, correcting my writings, and offering me his valuable advice throughout my PhD. I particularly admire Paul's clarity of thought, research and writing style and I consider myself very lucky that he was my supervisor. I would also like to thank Prof. Mike Osborne for his supervision during the first year of my PhD.

Perhaps the most exciting time during the past years was my visit to Prof. Yuji Nakatsukasa in Tokyo. Visiting Yuji in Tokyo was an enlightening and enriching experience. I will always be inspired by Yuji's ingeniousness and enthusiasm and I will be forever grateful for the countless hours he has spent on me. Furthermore, he was an amazing host, and an excellent guide for Tokyo: his food suggestions have forever spoiled me on sushi and ramen.

I would not have been in Oxford without the advice and help of Prof. Vassily Hatzimanikatis. Vassily was very kind in offering both professional and personal advice in the uncertain period after my undergraduate degree. Thank you very much, Vassily! Similarly, I would like to thank Dr. Alice Cicirello for her valuable advice and for being an inspiration throughout my PhD.

I am also thankful to Bartolomeo Stellato, Idris Kempf, Michael Garstka, George Pantazis and Goran Banjac for being such a friendly group of people in the Control Lab. And, of course, Wendy for being such an amazing administrator.

During the last four years in Oxford, I have met numerous amazing people that I would like to thank. These include Rob, Angeliki, Yaron, Lena, Zhao, Kyriakos, Leo and Bernardo. Special thanks should also go to Antonis & Artemis, for hosting me countless times in London and Kostis for numerous Skype calls and for hosting me in Barcelona.

Finally, I am grateful to my family for emotional and financial support throughout my life, Leonidas, for being a lifelong friend, and my girlfriend for making me happy and staying by my side over the past two years.

This work was supported by the EPSRC AIMS CDT grant EP/L015987/1, Schlumberger and the NII International Internship Program in Tokyo.

Abstract

The aim of this thesis is to develop scalable numerical optimization methods that can be used to address large problems of practical interest in the Machine Learning (ML) community.

We begin with a motivating application, Bayesian Optimization (BO), which is a popular ML method employed for hyperparameter tuning of expensive, black-box functions. We show that significant improvements can be brought to the scaling and the computational resources required by batch BO by adopting a distributionally ambiguous framework which replaces expensive integrations with Semidefinite Programming problems (SDPs). Typically, thousands of semidefinite problems are solved when performing BO, which makes their efficient solution essential.

Driven by the above application, we adapt the popular Alternating Method of Multipliers (ADMM) to allow for faster solution of general semidefinite problems. In semidefinite programming, ADMM spends most of its time in projecting matrices into the semidefinite cone. We suggest using state-of-the-art eigensolvers for this task, which brings a significant, up to 20x, speedup to ADMM in standard problems and up to 10x for SDPs arising from our BO algorithm.

We show that useful bounds for the projection errors can be computed efficiently. This considerably improves standard eigenvalue perturbation theory bounds (e.g. Davis-Kahan bounds) that suggest the projection accuracy to be inversely proportional to the spectral gap. Our results allow to tightly control the projection errors, so that convergence of ADMM either to a solution, or (for the first time in an approximate ADMM framework) to a certificate of infeasibility is guaranteed.

Unlike SDP, in quadratic programming ADMM spends most of its time in solving linear systems. We show how these linear systems can be solved efficiently in the quadratic problem of doubly stochastic matrix approximation. Our analysis reveals a surprising structure in the doubly stochastic approximation problem which allows the scaling of very large, sparse matrices, arising, for example, from the 3D structure of the human genome, or spectral clustering.

Finally, we devise an active set algorithm for the solution of nonconvex quadratic problems that include a norm constraint. Our approach is based on repeated solutions of the famous Trust-Region Subproblem, for which we derive novel results regarding the existence and the computation of its local-nonglobal minimizer. The resulting algorithm was demonstrated to be useful in sparse PCA and sequential quadratic programming.

Contents

Notation

Sets

$\mathbb{N}$	the set of natural numbers $\{1, 2, \dots\}$
$\mathbb{C}$	the set of complex numbers
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_+$ ($\mathbb{R}_{++}$)	the nonnegative (positive) real numbers
$\mathbb{R}^n(\mathbb{C}^n)$	the set of real (complex) $n-$ dimensional vectors
$\mathbb{R}^{m\times n}(\mathbb{C}^{m\times n})$	the set of real (complex) $m \times n$ real matrices
$\mathbb{S}^n$	the set of real $n \times n$ symmetric matrices
$\mathbb{S}^n_{++}$ ($\mathbb{S}^n_+$)	the set of real symmetric positive (semi)definite matrices
$\mathbb{S}^n_{--}$ ($\mathbb{S}^n_-$)	the set of real symmetric negative (semi)definite matrices
$\mathbb{H}^n$	the set of complex $n \times n$ Hermitian matrices
$\mathbb{H}^n_+$	the set of complex $n \times n$ Hermitian positive semidefinite matrices
$\mathbb{H}^n_-$	the set of complex $n \times n$ Hermitian negative semidefinite matrices

Relations

$A := B$	A is defined by B
$A \succ (\succeq) B$	$A - B$ is positive (semi)definite
$A \prec (\preceq) B$	$A - B$ is negative (semi)definite
$x \approx_\epsilon y$	the vectors x, y satisfy $\|x - y\| \leq \epsilon$
$x \perp y$	x is orthogonal to y
$x \not\perp y$	x is not orthogonal to y
$O(\cdot)$	asymptotic upper bound on the growth of a function
$o(\cdot)$	asymptotic, non-tight upper bound on the growth of a function

Norms

$\|\cdot\|$	vector or matrix norm
$\|x\|_1$	$1-$norm of a vector x: $\|x\|_1 := \sum \|x_i\|$
$\|x\|_2$	$2-$norm of a vector x: $\|x\|_2 := \sqrt{\sum x_i^2}$
$\|x\|_\infty$	$\infty-$norm of a vector x: $\|x\|_\infty := \max \|x_i\|$
$\|X\|_2$	Spectral norm of a matrix X
$\|X\|_F$	Frobenius norm of a matrix X

Linear Algebra

$\mathbf{1}_n$	$n-$dimensional vector of ones
$\mathbf{1}_{n\times m}$	$n \times m$ matrix of ones
I	identity matrix of appropriate dimensions
A^T	transpose of a matrix A
A^H	adjoing of a matrix A
$A_\perp$	an orthonormal matrix that spans the nullspace of A^T
$\begin{bmatrix} x; & y \end{bmatrix}$	vertical concatenation of vectors (or matrices) x, y
$\begin{bmatrix} a_i \end{bmatrix}_{i\in\mathcal{E}}$	horizontal concatenation of the vectors a_i for all indices in $\mathcal{E}$.
$x_{(i)}$	$i-$th element of a vector x
$A_{(i,j)}$	element on the $i-$th row and $j-$th column of a matrix A
$\mathrm{vec}(A)$	vector composed by stacking columns of a matrix A [1, p. 12.3.4]
$\mathrm{vec}_u(A)$	vector composed by stacking the upper triangular elements of a matrix A in the same order as $\mathrm{vec}(A)$. The off-diagonal elements are scaled by $\sqrt{2}$.
$\mathrm{mat}(x)$	inverse operation of $\mathrm{vec}(\cdot)$
$\mathrm{mat}_u(x)$	inverse operation of $\mathrm{vec}_u(\cdot)$
$\mathrm{card}(A)$	cardinality of a vector or matrix A
$\mathcal{N}(A)$	nullspace of a matrix A
$\mathcal{R}(A)$	range of a matrix A
$\lambda_{\max}(A)$	rightmost eigenvalue of A
$\sigma_i(A)$	$i-$th largest singular value of A
$\otimes$	Kronecker product

$\oplus$	Kronecker sum
$\odot$	Hadamard (elementwise) product
$\mathrm{Re}(x)$	real part of a complex number x
$\mathrm{Im}(x)$	imaginary part of a complex number x
ϵ_u	Unit roundoff

Operator theory

$\mathcal{H}$	real Hilbert space
$\langle \cdot, \cdot \rangle$	inner product
Id	identity operator
$\Pi_{\mathcal{C}}$	projection onto a set $\mathcal{C}$: $\Pi_{\mathcal{C}}(y) := \arg\min_{x \in \mathcal{C}} \|y - x\|$
prox_f	proximal operator $\mathrm{prox}_f(\phi) := \arg\min_{\chi} \left(f(\chi) + \frac{1}{2}\|\chi - \phi\|^2 \right)$
$\mathrm{Fix}T$	fixed-point set of an operator T
$\mathcal{R}(T)$	range of an operator T
$\mathcal{I}_{\mathcal{C}}$	indicator function of a set $\mathcal{C}$
$\arg\min f$	the set of minimizers of a function f
$\arg\max f$	the set of maximizers of a function f
∇f	gradient of a function f
f'	derivative of a function f
$\nabla^2 f$	Hessian of a function f
∂f	(sub)differential of a function f
$\mathrm{conv}(\mathcal{C})$	convex hull of a set $\mathcal{C}$
$\mathrm{cl}(\mathcal{C})$	closure of a set $\mathcal{C}$
$\mathrm{int}(\mathcal{C})$	interior of a set $\mathcal{C}$
C^{∞}	recession cone of a set $\mathcal{C}$
$\mathcal{K}^{\circ}$	polar of a cone $\mathcal{K}$
$S_{\mathcal{C}}$	support function of a set $\mathcal{C}$

Probability theory

$\mathcal{N}(\mu, \Sigma)$	normal distribution with mean μ and variance Σ
$\mathcal{U}(l, u)$	uniform distribution defined in $[l, u] \subseteq \mathbb{R}$
$\mathbb{E}_{\mathbb{P}}$	expectation over the probability distribution $\mathbb{P}$
$x \sim \mathbb{P}$	the random variable x is distributed according to the distibution $\mathbb{P}$

Acronyms

ADMM	Alternating Directions Method of Multipliers
BLCB	Batch Lower Confidence Bound
CG	Conjugate Gradient algorithm
EI	Expected Improvement
EQP	Equality-constrained Quadratic Problem
GP	Gaussian Process
IID	Independent and Identically Distributed
KKT	Karush—Kuhn—Tucker (conditions)
LOBPCG	Locally Optimal Block Preconditioned Conjugate Gradient Method
L-BFBS-B	Limited memory Broyden-–Fletcher-–Goldfarb-–Shanno algorithm with Box constraints
LP	Linear Program
LP-EI	Local Penalization Expected Improvement
LICQ	Linear Independence Constraint Qualification
MINRES	Minimum Residuals algorithm
NP	Nondeterministic Polynomial time
OEI	Optimistic Expected Improvement
PPO	Proximal Policy Optimization
QEI	Multipoint Expected Improvement
QEI-CL	Multipoint Expected Improvement, Constant Liar heuristic
QP	Quadratic Program
SOCP	Second Order Conic Program
SDP	Semidefinite Program
SQP	Sequential Quadratic Programming
TRS	Trust-Region Subproblem

1 Introduction

1.1 Background

Numerical optimization is at the heart of virtually every decision making process. Machine learning, statistics, finance, manufacturing, operations research and control theory are among the many fields that rely on optimization. The wide adoption of numerical optimization in numerous scientific fields follows a century of extensive research in the field. The Simplex algorithm is a notable example of a constrained programming solver, able to efficiently solve Linear Programs. The Simplex algorithm is an early constraint optimization solver that been used in numerous fields, such as operations research, and is still competitive today [2]. However, Interior Point Methods have gradually gained attention, initially for solving Linear Problems, and eventually for generic, nonlinear (convex) problems. This has led to a surge in the use of a wider, nonlinear, class of optimization problems, including Quadratic and Semidefinite Programming.

Most currently available commercial optimization solvers rely on Interior Point methods, because of their excellent theoretical and practical performance [3]. Interior Point algorithms solve a constrained optimization problem by parametrizing the constraints as penalties in the objective resulting in an unconstrained optimization problem. The minimization is performed by Newton-like steps on the parametrized

unconstrained problem of the every iteration. It follows that, at every iteration, a matrix factorization has to be computed to perform the Newton step, which constitutes the main computational bottleneck of Interior Point Methods.

Every year, optimization algorithms are expected to address problems of increasing size. Notable recent examples include various machine learning applications, where models with billions of parameters have to be trained to very large datasets [4], and the field of Sum-of-Squares programming which requires the solution of Semidefinite Problems of increasing size [5].

The limitations of Interior Point methods become evident in these large problems, as factorizing "Hessian" matrices, arising from the Newton-like steps of the Interior Point methods, quickly become computationally intractable. For these reason, first-order methods have seen a resurgence over the past decade. As the name suggests, first-order methods iterate using only first-order information, i.e. gradients, without incorporating second order corrections. This makes them considerably more scalable, at the cost of decreased robustness (in the sense that first-order methods might require exceedingly many iterations to solve some problems), convergence rates and computed accuracy. Thus, first-order methods are suitable for cases where a modest accuracy solution is satisfactory for the problem. This is often the case in Machine Learning, where datasets can often be inexact or noisy and the objective, up to a point, arbitrary. In the case of convex relaxations, i.e. approximations to problems that are non-convex, which is very typical in semidefinite programming, solving a problem to high precision can be a waste of computational resources since it is only an approximation of the original problem.

A very popular first-order algorithm that has received widespread attention is the Alternating Directions Method of Multipliers (ADMM) [6] [7]. ADMM considers the solution of split optimization problems in the form

$$\text{minimize} \quad f(x) + g(x) \tag{1.1}$$

where f and g are convex functions. This formulations is very general and can include constrained optimization by defining one of the functions as the indicator

function of the feasible set. For a wide class of problems (including LPs, QPs, SDPs, and SOCPs) ADMM consists of relatively "simple" iterations such as solving linear systems (of fixed right hand side), and projecting into the feasible set. Each of these operations can be computed using standard linear algebra operations. This allows ADMM to be scalable and easy to implement.

In this thesis we focus on two ways that aim to scale ADMM to larger problems. Firstly, one can implement the ADMM operations using linear algebra operations that are more scalable, but approximate. A common example of this approach is the use of the celebrated Conjugate Gradient method for the approximate solution of linear systems. This can be very useful when using ADMM for quadratic problems, as most of the time is spent on solving linear systems. However, in semidefinite programming, ADMM spends most of its time on projecting into the feasible set, i.e. into the semidefinite cone. Fortunately, this relies on another standard linear algebra operation, the eigendecomposition. Using approximate eigensolvers in ADMM, while retaining convergence guarantees, was a key, novel contribution of this thesis.

Secondly, one can scale ADMM by exploiting the structure of particular problems. For, example in some special problems the Linear Systems embedded into ADMM admit a trivial closed form expression. This is highlighted, and exploited, in two problems considered in this thesis.

Finally, there are certain classes of problems that are neither convex, nor is it obvious that ADMM will lead to the most efficient solution. Although, nonconvexity might preclude convergence guarantees that commonly characterize convex algorithms, nonconvex optimization is a very important field of research. For these reasons, the last chapter of thesis is dedicated in how approximate eigensolvers can be helpful in a novel active-set algorithm addressing a nonconvex problem of practical interest.

1.2 Outline

We now proceed to outline the contributions of each of the remaining chapters in this thesis:

Chapter 2: In this chapter we propose a novel, theoretically-grounded, acquisition function for Batch Bayesian Optimization informed by insights from distributionally ambiguous optimization. Our acquisition function is a lower bound on the well-known Expected Improvement function, which requires evaluation of a Gaussian expectation over a multivariate piecewise affine function. Our bound is computed instead by evaluating the best-case expectation over all probability distributions consistent with the same mean and variance as the original Gaussian distribution. Unlike alternative approaches, including Expected Improvement, our proposed acquisition function avoids multi-dimensional integrations entirely, and can be computed exactly – even on large batch sizes – as the solution of a tractable convex optimization problem. Our suggested acquisition function can also be optimized efficiently, since first and second derivative information can be calculated inexpensively as by-products of the acquisition function calculation itself. We derive various novel theorems that ground our work theoretically and we demonstrate superior performance via simple motivating examples, benchmark functions and real-world problems. This chapter is based on:

- **N. Rontsis**, M. A. Osborne, P. J. Goulart. "Distributionally Ambiguous Optimization in Batch Bayesian Optimization". Accepted under minor corrections in: *Journal of Machine Learning Research (Special Issue on Bayesian Optimization)*.

Chapter 3: In this chapter, we demonstrate that tenfold speedups can be brought to ADMM for Semidefinite Programming with virtually no decrease in robustness and provable convergence simply by projecting approximately to the Semidefinite cone. Instead of computing the projections via "exact" eigendecompositions that scale cubically with the matrix size and cannot be warm-started, we suggest using state-of-the-art factorization-free, approximate eigensolvers thus achieving almost quadratic scaling and the crucial ability of warm-starting. Based on a novel theoretical result, presented in Chapter 4, we are able to circumvent the numerical instability of the eigendecomposition and thus maintain a tight control on the

projection accuracy, which in turn guarantees convergence, either to a solution or a certificate of infeasibility, of the ADMM algorithm. To achieve this, we extend recent results from [8] to prove that reliable infeasibility detection can be performed with ADMM even in the presence of approximation errors. The suggested Algorithm can bring more than a tenfold speedup for the Bayesian Optimization process of Chapter 2. Furthermore, in all of the considered problems of SDPLIB that "exact" ADMM can solve in a few thousand iterations, our approach brings a significant, up to 20x, speedup without a noticeable increase on ADMM's iterations. This chapter is based on:

- **N. Rontsis**, P. J. Goulart, Y. Nakatsukasa. "Efficient Semidefinite Programming with Approximate ADMM". In: *arXiv:1912.02767* (2019).

Chapter 4: In this Chapter we derive a key theoretical result used in the convergence proofs of the Approximate ADMM of Chapter 3. When a projection of a symmetric or Hermitian matrix to the positive semidefinite cone is computed approximately (or to working precision on a computer), a natural question is to quantify its accuracy. A straightforward bound invoking standard eigenvalue perturbation theory (e.g. Davis-Kahan and Weyl bounds) suggests that the accuracy would be inversely proportional to the spectral gap, implying it can be poor in the presence of small eigenvalues. This work shows that a small gap is not a concern for projection onto the semidefinite cone, by deriving error bounds that are gap-independent. This chapter is based on:

- P. J. Goulart, Y. Nakatsukasa, **N. Rontsis**. "Accuracy of approximate projection to the semidefinite cone". In: *arXiv:1908.01606* (2019).

Chapter 5: In this chapter, we consider the least-squares approximation of a matrix C in the set of doubly stochastic matrices with the same sparsity pattern as C. Our approach is based on applying ADMM to a reformulation of the original problem. Our resulting algorithm requires an initial Cholesky factorization of a positive definite matrix that has the same sparsity pattern as $C + I$, followed by

simple iterations whose complexity is linear in the number of nonzeros in C, thus ensuring excellent scalability and speed. We demonstrate the advantages of our approach in a series of experiments on problems with up to 82 million nonzeros; these include normalizing large scale matrices arising from the 3D structure of the human genome, clustering applications, and the SuiteSparse matrix library. Overall, our experiments illustrate the outstanding scalability of our algorithm; matrices with millions of nonzeros can be approximated in a few seconds on modest desktop computing hardware. This chapter is based on:

- **N. Rontsis**, P. J. Goulart. "Optimal Approximation of Doubly Stochastic Matrices". In: *arXiv:1910.05295* (2019).

Chapter 6: In this chapter, we present an algorithm for the minimization of a nonconvex quadratic function subject to linear inequality constraints and a two-sided bound on the 2-norm of its solution. The algorithm minimizes the objective using an active-set method by solving a series of Trust-Region Subproblems (TRS). Underpinning the efficiency of this approach is that the global solution of the TRS has been widely studied in the literature, resulting in remarkably efficient algorithms and software. We extend these results by proving that nonglobal minimizers of the TRS, or a certificate of their absence, can also be calculated efficiently by computing the two rightmost eigenpairs of an eigenproblem. We demonstrate the usefulness and scalability of the algorithm in a series of experiments that often outperform state-of-the-art approaches; these include calculation of high-quality search directions arising in Sequential Quadratic Programming on problems of the `CUTEst` collection, and Sparse Principal Component Analysis on a large text corpus problem (70 million nonzeros) that can help organize documents in a user interpretable way. This chapter is based on:

- **N. Rontsis**, P. J. Goulart, Y. Nakatsukasa. "An active-set algorithm for norm constrained quadratic problems". In: *arXiv:1906.04022* (2019).

Chapter 7: This chapter concludes the thesis and presents some directions for future research.

2

Distributionally Ambiguous Optimization for Batch Bayesian Optimization

2.1 Introduction

In this chapter we introduce an SDP relaxation for Batch Bayesian Optimization, a popular machine learning algorithm that we proceed to describe. Our approach significantly improves the scaling of the overall machine algorithm[1] while relying on state-of-the-art SDP solvers. Further scaling will be achieved with the novel semidefinite solver that we will introduce in the next chapter.

When dealing with numerical optimization problems in engineering applications, one is often faced with the optimization of a process or function that is expensive to evaluate and depends on a number of tuning parameters. Examples include the training of machine learning algorithms [9], algorithms for robotic tasks [10] or reinforcement learning [11]. Given the cost of evaluating the process, we wish to select the parameters at each stage of evaluation carefully so as to optimize the process using as few experimental evaluations as possible. We are concerned with problem instances wherein there is the capacity to speed up optimization by performing k experiments in *parallel* and, if needed, repeatedly select further batches with cardinality k as part of some sequential decision making process.

[1] With respect to the batch size which we proceed to define.

It is common to assume a surrogate model for the outcome $f : \mathbb{R}^n \to \mathbb{R}$ of the process to be optimized. This model, which is based on both prior assumptions and past function evaluations, is used to determine a collection of k input points for the next set of evaluations. Bayesian Optimization provides an elegant surrogate model approach and has been shown to outperform other state-of-the-art algorithms on a number of challenging benchmark functions [12]. It models the unknown function f with a Gaussian Process (GP) [13], a probabilistic function approximator which can incorporate prior knowledge such as smoothness, trends, etc.

A comprehensive introduction to GPs can be found in [13]. In short, modeling a function with a GP amounts to modeling the function as a realization of a stochastic process. In particular, we assume that the outcomes of function evaluations are normally distributed random variables with known *prior* mean function $m : \mathbb{R}^n \to \mathbb{R}$ and *prior* covariance function $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Prior knowledge about f, such as smoothness and trends, can be incorporated through judicious choice of the covariance function κ, while the mean function m is commonly assumed to be zero. Note that the choice of κ can often be ad hoc. A training dataset $\mathcal{D} = (X^{\mathcal{D}}, y^{\mathcal{D}})$ of ℓ past function evaluations $y_i^{\mathcal{D}} = f(X_i^{\mathcal{D}})$ for $i = 1 \dots \ell$, with $y^{\mathcal{D}} \in \mathbb{R}^{\ell}, X^{\mathcal{D}} \in \mathbb{R}^{\ell \times n}$ is then used to calculate the *posterior* of f.

The celebrated GP regression equations [13] give the posterior

$$y|\mathcal{D} \sim \mathcal{N}(\mu(X), \Sigma(X)) \tag{2.1}$$

on a batch of k test locations $X \in \mathbb{R}^{k \times n}$ as a multi-variate normal distribution in a closed form formula. The posterior mean value μ and variance Σ depend also on the dataset $\mathcal{D}$, but we do not explicitly indicate this dependence in order to simplify the notation. Likewise, the posterior $y|\mathcal{D}$ is a normally distributed random variable whose mean $\mu(X)$ and covariance $\Sigma(X)$ depend on X, $y^{\mathcal{D}}$, and $X^{\mathcal{D}}$, but we do not make this explicit.

Given the surrogate model, we wish to identify some selection criterion for choosing the next batch of points to be evaluated. Such a selection criterion is known as an *acquisition function* in the terminology of Bayesian Optimization. Ideally, such

an acquisition function would take into account the number of remaining evaluations that we can afford, e.g. by computing a solution via dynamic programming to construct an optimal sequence of policies for future batch selections. However, a probabilistic treatment of such a criterion is computationally intractable, involving multiple nested maximization-marginalization steps [14].

To avoid this computational complexity, myopic acquisition functions that consider only the one-step return are typically used instead. For example, one could choose to maximize the one-step *Expected Improvement* (described more fully in §2.1.1) over the best evaluation observed so far, or maximize the probability of having an improvement in the next batch over the best evaluation. Other criteria use ideas from the bandit [15] and information theory [16] literature. In other words, the intractability of the multistep lookahead problem has spurred instead the introduction of a wide variety of myopic heuristics for batch selection.

2.1.1 Expected improvement

We will focus on the (one-step) Expected Improvement criterion, which is a standard choice and has been shown to achieve good results in practice [9]. In order to give a formal description we first require some definitions related to the optimization procedure of the original process. At each step of the optimization procedure, define $y^{\mathcal{D}} \in \mathbb{R}^{\ell}$ as the vector of ℓ past function values evaluated at the points $X^{\mathcal{D}} \in \mathbb{R}^{\ell \times n}$, and $X \in \mathbb{R}^{k \times n}$ as a candidate set of k points for the next batch of evaluations. Then the classical Expected Improvement acquisition function is defined as

$$\begin{aligned} \alpha(X) &= \underline{y^{\mathcal{D}}} - \mathbb{E}[\min(y_1, \ldots, y_k, \underline{y^{\mathcal{D}}})|\mathcal{D}] \\ &\text{with } y|\mathcal{D} \sim \mathcal{N}\Big(\mu(X), \Sigma(X)\Big), \end{aligned} \tag{2.2}$$

where $\underline{y^{\mathcal{D}}}$ is the element-wise minimum of $y^{\mathcal{D}}$, i.e., the minimum value of the function f achieved so far by any known function input. In the above definition we assume perfect knowledge of $\underline{y^{\mathcal{D}}}$, which implies a noiseless objective. A noisy objective requires the introduction of heuristics discussed in detail in [17]. For the purposes of clarity, a noiseless objective is assumed for the rest of the document. This is

not constraining, as most of the heuristics discussed in [17] are compatible with the theoretical analysis presented in the rest of the chapter.

Selection of a batch of points to be evaluated with optimal Expected Improvement amounts to finding some $X \in \arg\max[\alpha(X)]$. Unfortunately, direct evaluation of the acquisition function α requires the k–dimensional integration of a piecewise affine function; this is potentially a computationally expensive operation. This is particularly problematic for gradient-based optimization methods, wherein $\alpha(X)$ may be evaluated many times when searching for a maximizer. Regardless of the optimization method used, such a maximizer must also be computed again for every step in the original optimization process, i.e., every time a new batch of points is selected for evaluation. Therefore a tractable acquisition function should be used. In contrast to (2.2), the acquisition function we will introduce in Section 2.2 avoids expensive integrations and can be calculated efficiently with standard software tools.

2.1.2 Related work

Despite the intractability of (2.2), [18] presented an efficient way of calculating α and its derivative $d\alpha/dX$ [19] by decomposing it into a sum of q–dimensional Gaussian Cumulative Distributions, which can be calculated efficiently using the seminal work of [20]. However, the number of calls to the q–dimensional Gaussian Cumulative Distribution grows quadratically with respect to the batch size q. To avoid this issue, [21] and [22] rely on heuristics to derive a multi-point criterion. Both methods choose the batch points in a greedy, sequential way, which restricts them from exploiting the interactions between the batch points in a probabilistic manner. Finally Parallel Predictive Entropy Search (PPES), a non-greedy information theoretic acquisition function was derived by [16].

2.2 Distributionally ambiguous optimization for Bayesian Optimization

We now proceed to the main contribution of the chapter. We draw upon ideas from the Distributionally Ambiguous Optimization community to derive a novel,

tractable, acquisition function that lower bounds the expectation in (2.2). Our acquisition function:

- is theoretically grounded;
- is numerically reliable and consistent, unlike Expected Improvement-based alternatives (see §2.3);
- can be evaluated efficiently, even for large batch sizes; and
- provides first and second order derivative information inexpensively.

In particular, we use the GP posterior (2.1) derived from the GP to determine the mean $\mu(X)$ and variance $\Sigma(X)$ of $y|\mathcal{D}$ given a candidate batch selection X, but we thereafter ignore the Gaussian assumption and consider only that $y|D$ has a distribution embedded within a family of distributions $\mathcal{P}$ that share the mean $\mu(X)$ and covariance $\Sigma(X)$ calculated by the standard GP regression equations. In other words, we define

$$\mathcal{P}(\mu, \Sigma) = \left\{ \mathbb{P} \text{ is a p.d.f. on } \mathbb{R}^k \;\middle|\; \mathbb{E}_{\mathbb{P}}[\xi] = \mu, \mathbb{E}_{\mathbb{P}}[\xi\xi^T] = \Sigma + \mu\mu^T \right\},$$

where ξ is defined as the random variable of the output y conditioned on the dataset observed so far $\mathcal{D}$. We will omit the dependence of μ and Σ on X, and will denote the set $\mathcal{P}(\mu, \Sigma)$ simply as $\mathcal{P}$, where the context is clear. Note in particular that $\mathcal{N}(\mu, \Sigma) \in \mathcal{P}(\mu, \Sigma)$ for any choice of mean μ or covariance Σ.

One can then construct lower and upper bound for the Expected Improvement by minimizing or maximizing over the set $\mathcal{P}$ respectively, i.e., by writing

$$\inf_{\mathbb{P}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}}[g(\xi)] \leq \mathbb{E}_{\mathcal{N}(\mu,\Sigma)}[g(\xi)] \leq \sup_{\mathbb{P}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}}[g(\xi)], \tag{2.3}$$

where the random vector $\xi \in \mathbb{R}^k$ and the function $g : \mathbb{R}^k \to \mathbb{R}$ are chosen according to the definition (2.2) of the Expected Improvement i.e., $\xi = y|\mathcal{D}$, and

$$g(\xi) = \underline{y^{\mathcal{D}}} - \min(\xi_1, \ldots, \xi_k, \underline{y^{\mathcal{D}}}) \tag{2.4}$$

so that $\mathbb{E}_{\mathcal{N}(\mu(X),\Sigma(X))}[g(\xi)] = \alpha(X)$. Thus, the middle term in (2.3) is equivalent to the Expected Improvement.

Perhaps surprisingly, both of the bounds in (2.3) are computationally tractable even though they seemingly require optimization over the infinite-dimensional (but convex) set of distributions $\mathcal{P}$. For either case, these bounds can be computed exactly via transformation of the problem to a tractable SDP using distributionally ambiguous optimization techniques [23].

We will focus on the upper bound $\sup_{\mathbb{P}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}}[g(\xi)]$ in (2.3), hence adopting an *optimistic* modeling approach. The lower bound turns out to be of limited use, and we show in Proposition 2.A.4 of Appendix 2.A.4 that it is trivial to evaluate and independent of Σ. Hence, we informally assume that the distribution of function values is such that it yields the largest possible Expected Improvement compatible with the mean and covariance computed by the GP, which we put together in the second order moment matrix Ω of the posterior as

$$\Omega := \begin{bmatrix} \Sigma + \mu\mu^T & \mu \\ \mu^T & 1 \end{bmatrix}. \tag{2.5}$$

We will occasionally write this explicitly as $\Omega(X)$ to highlight the dependency of the second order moment matrix on X.

The following result says that the upper (i.e., optimistic) bound in (2.3) can be computed via the solution of a convex semidefinite optimization problem whose objective function is linear in Ω. Semidefinite Problems (SDPs) are convex optimization problems with matrices as decision variables that are constrained to be positive semidefinite. SDPs enjoy strong theoretical properties which guarantee that they can be solved globally in polynomial time, as well as a variety of mature software tools [24], [25], [26]. The reader can refer to [3] and [27] for an introduction to SDPs.

Theorem 2.2.1. *For any $\Sigma \succ 0$ the optimal value of the semi-infinite optimization problem*

$$\sup_{\mathbb{P}\in\mathcal{P}(\mu,\Sigma)} \mathbb{E}_{\mathbb{P}}[g(\xi)]$$

coincides with the optimal value of the following semidefinite program:

$$\begin{aligned} p(\Omega) := \; & -\sup \quad \langle \Omega, M \rangle \\ & s.t. \quad M - C_i \preceq 0, \quad i = 0, \ldots, k, \end{aligned} \tag{P}$$

where $M \in \mathbb{S}^{k+1}$ *is the decision variable, and* $C_0 := 0$,

$$C_i := \begin{bmatrix} 0 & e_i/2 \\ e_i^T/2 & -\underline{y^{\mathcal{D}}} \end{bmatrix}, \qquad i = 1, \ldots, k, \tag{2.6}$$

are auxiliary matrices defined using $\underline{y^{\mathcal{D}}}$ *and the standard basis vectors* e_i *in* $\mathbb{R}^k$.

Proof. See Appendix 2.A.1. □

We therefore propose the computationally tractable acquisition function

$$\bar{\alpha}(X) := p\Big(\Omega(X)\Big) \geq \alpha(X) \quad \forall X \in \mathbb{R}^{k \times n},$$

which we will call *Optimistic Expected Improvement (OEI)*, as it is an optimistic variant of the Expected Improvement function in (2.2).

This computational tractability comes at the cost of inexactness in the bounds (2.3), which is a consequence of maximizing over a set of distributions containing the Gaussian distribution as just one of its members. Indeed, we prove in Theorem 2.A.6 of Appendix 2.A.4 that the maximizing distribution is discrete with $k+1$ possible outcomes that can be constructed by the Lagrange multipliers of (P). We show in §2.3 that this inexactness is of limited consequence in practice and it mainly renders the acquisition function more explorative. In particular, we show in Figure 2.1 that the qualitative behavior of OEI closely matches that of Multi-point Expected Improvement (QEI). Nevertheless, there remains significant scope for tightening the bounds in (2.3) via imposition of additional convex constraints on the set $\mathcal{P}$, e.g. by restricting $\mathcal{P}$ to symmetric or unimodal distributions [28]. Most of the results in this work would still apply, *mutatis mutandis*, if such structural constraints were to be included.

Although the value of $p(\Omega)$ for any fixed Ω is computable via solution of an SDP, the non-convexity of the GP posterior (2.1) that defines the mapping $X \to \Omega(X)$ means that $\bar{\alpha}(X) = p\Big(\Omega(X)\Big)$ is still non-convex in X. This is unfortunate, since we ultimately wish to maximize $\bar{\alpha}(X)$ in order to identify the next batch of points to be evaluated experimentally.

However we can still optimize $\bar{\alpha}$ locally via non-linear programming. We will establish that a second order method is applicable by showing that $\bar{\alpha}(X)$ is twice

differentiable under mild conditions. Such an approach would also be efficient as the Hessian of $\bar{\alpha}$ can be calculated inexpensively. To show these results we will begin by considering the differentiability of p as a function of Ω.

Theorem 2.2.2. *The optimal value function $p : \mathbb{S}^{k+1}_{++} \to \mathbb{R}$ defined in problem (P) is differentiable on its domain with $\partial p(\Omega)/\partial\Omega = -\bar{M}(\Omega)$, where $\bar{M}(\Omega)$ is the unique optimal solution of (P) at Ω.*

Proof. See Appendix 2.A.2. □

The preceding result shows that $\partial p(\Omega)/\partial\Omega$ is produced as a byproduct of evaluation of $\sup_{\mathbb{P}\in\mathcal{P}} \mathbb{E}_{\mathbb{P}}[g(\xi)]$, since it is simply $-\bar{M}(\Omega)$, the negation of the unique optimizer of (P). The simplicity of this result suggests consideration of second derivative information of $p(\Omega)$, i.e., derivatives of $-\bar{M}(\Omega)$. The following result proves that this is well defined and tractable for any $\Omega \succ 0$:

Theorem 2.2.3. *The optimal solution function $\bar{M} : \mathbb{S}^{k+1}_{++} \to \mathbb{S}^{k+1}$ in problem (P) is differentiable on $\mathbb{S}^{k+1}_{++}$. Every directional derivative of $\bar{M}(\Omega)$ is the unique solution to a sparse linear system with $O(k^3)$ nonzeros.*

Proof. See Appendix 2.A.3. □

We can now consider the differentiability of $\bar{\alpha} = p \circ \Omega$. The following Corollary establishes this under certain conditions.

Corollary 2.2.4. *$\bar{\alpha} : \mathbb{R}^{k\times n} \to \mathbb{R}$ is twice differentiable on any X for which $\Sigma(X) \succ 0$ and the mean and kernel functions of the underlying GP are twice differentiable.*

Proof. By examining the GP Regression equations [13] and Equation (2.5), we conclude that $\Omega(X)$ is twice differentiable on $\mathbb{R}^{k\times n}$ if the kernel and mean functions of the underlying Gaussian Process are twice differentiable. Hence, $\bar{\alpha}(X) = p(\Omega(X))$ is twice differentiable for any $\Omega(X) \succ 0$ as a composition of twice differentiable functions. Examining (2.5) reveals that $\Omega(X) \succ 0$ is equivalent to $\Sigma(X) \succ 0$, which concludes the proof. □

A rank deficient $\Sigma(X)$ implies perfectly correlated outcomes. At these points both OEI and QEI can be shown to be non-differentiable. However, this is not constraining in practice as both QEI and OEI can be calculated by considering a smaller, equivalent problem. It is also not an issue for ascent based methods for maximizing $\bar{\alpha}$, as an ascent direction can be obtained by an appropriate perturbation of the perfectly correlated points.

We are now in a position to derive expressions for the gradient and the Hessian of $\bar{\alpha} = p \circ \Omega$. For simplicity of notation we consider derivatives over $\bar{x} = \mathrm{vec}(X)$. Application of the chain rule to $\bar{\alpha}(\bar{x}) = p(\Omega(\bar{x}))$ gives:

$$\frac{\partial \bar{\alpha}(\bar{x})}{\partial \bar{x}_{(i)}} = \left\langle \frac{\partial p(\Omega)}{\partial \Omega}, \frac{\partial \Omega(\bar{x})}{\partial \bar{x}_{(i)}} \right\rangle = -\left\langle \bar{M}(\Omega), \frac{\partial \Omega(\bar{x})}{\partial \bar{x}_{(i)}} \right\rangle. \tag{2.7}$$

Note that the second term in the rightmost inner product above depends on the particular choice of covariance function κ and mean function m. It is straightforward to compute (2.7) in modern graph-based `autodiff` frameworks, such as the `TensorFlow`-based `GPflow`. Differentiating again (2.7) gives the Hessian of $\bar{\alpha}$:

$$\begin{aligned} \frac{\partial^2 \bar{\alpha}(\bar{x})}{\partial \bar{x}_{(i)} \partial \bar{x}_{(j)}} &= -\frac{\partial}{\partial x_{(i)}} \left\langle \bar{M}(\Omega), \frac{\partial \Omega(\bar{x})}{\partial \bar{x}_{(j)}} \right\rangle \\ &= -\left\langle \bar{M}(\Omega), \frac{\partial^2 \Omega(\bar{x})}{\partial \bar{x}_{(i)} \partial \bar{x}_{(j)}} \right\rangle - \left\langle \frac{\partial \bar{M}(\Omega(\bar{x}))}{\partial \bar{x}_{(i)}}, \frac{\partial \Omega(\bar{x})}{\partial \bar{x}_{(j)}} \right\rangle, \end{aligned} \tag{2.8}$$

where $\partial \bar{M} / \partial \bar{x}_{(i)}$ is the directional derivative of $\bar{M}(\Omega)$ across the perturbation $\partial \Omega(\bar{x}) / \partial \bar{x}_{(i)}$. According to Theorem 2.2.3, each of these directional derivatives exists and can be computed via solution of a sparse linear system.

2.3 Empirical analysis

In this section we demonstrate the effectiveness of our acquisition function against a number of state-of-the-art alternatives. The acquisition functions we consider are listed in Table 2.1.

We show that our acquisition function OEI achieves better performance than alternatives and highlight simple "failure" cases exhibited by competing methods. In making the following comparisons, extra care should be taken in the setup used.

Table 2.1: List of acquisition functions

Key	Description
OEI	Optimistic Expected Improvement *(Our novel algorithm)*
QEI	Multi-point Expected Improvement [19]
QEI-CL	Constant Liar (*"mix"* strategy) [29]
LP-EI	Local Penalization Expected Improvement [21]
BLCB	Batch Lower Confidence Bound [15]

This is because Bayesian Optimization is a multifaceted procedure that depends on a collection of disparate elements (e.g. kernel/mean function choice, normalization of data, acquisition function, optimization of the acquisition function) each of which can have a considerable effect on the resulting performance [9, 11]. For this reason we test the different algorithms on a unified testing framework, based on `GPflow`, available online at:

`https://github.com/oxfordcontrol/Bayesian-Optimization`

Our acquisition function is evaluated via solution of a semidefinite program, and as such it benefits from the huge advances of the convex optimization field. A variety of standard tools exist for solving such problems, including `MOSEK` [25], `SCS` [30] and `CDCS` [31]. We chose the first-order [3], freely-available solver `SCS`, which scales well with batch size and allows for solver warm-starting between acquisition function evaluations.

Warm starting allows for a significant speedup since the acquisition function is evaluated repeatedly at nearby points by the non-linear solver. This results in solving (P) repeatedly for similar Ω. Warm-starting the SDP solver with the previous solution reduces `SCS`'s iterations by 77% when performing the experiments of Figure 2.3. Moreover, Theorem 2.2.2 provides the means for first-order warm starting. Indeed, the derivative of the solution across the change of the cost matrix Ω can be calculated, allowing us to take a step in the direction of the gradient and warm start from that point. This reduces `SCS`'s iterations by a further 43%, resulting in a 87% total speedup compared with no warm starting at all.

Indicative timing results for the calculation of OEI, QEI and their derivatives are listed in Table 2.2. The dominant operation for calculating OEI and its gradient is solving (P). This makes OEI much faster than QEI, which is in line with the complexity of the dominant operation in SDP solvers based on first-order operator splitting methods such as `SCS` or `CDCS` which, for our problem, is $O\left(k^4\right)$. In the next chapter, we will introduce novel SDP solution techniques with a complexity of their dominant operation would be $O\left(k^3\right)$.

On the other hand, assume that, given the solution of (P), we want to also calculate the Hessian of OEI. This would entail the following two operations:

Calculating $\partial\bar{M}/\partial X_{(i,j)}$ given $\partial\Omega(X)/\partial X_{(i,j)}$. According to Lemma 2.2.3 this can be obtained as a solution to a sparse linear system. We used `Intel MKL PARDISO` to solve these linear systems efficiently.

Calculate $\partial\Omega(X)/\partial X_{(i,j)}$ and apply chain rules of (2.8) to get the Hessian of the acquisition function $\bar{\alpha} = p \circ \Omega$ given the gradient and Hessian of p. We used `Tensorflow`'s Automatic Differentiation for this part, without any effort to optimize its performance. Considerable speedups can be brought by e.g. running this part on GPU, or automatically generating low-level code optimized specifically for this operation.

Note that the computational tractability of the Hessian is only made possible due to the novelty of Theorem 2.2.2 which exploits the "structure" of (P)'s optimizer.

We chose the `KNITRO v10.3` [32] Sequential Quadratic Optimization (SQP) non-linear solver with the default parameters for the optimization of OEI. Explicitly providing the Hessian on the experiments of Figure 2.3 reduces `KNITRO`'s iterations by 49% relative to estimating the Hessian via the symmetric-rank-one (SR1) update method included in the `KNITRO` suite. Given the inexpensiveness of calculating the Hessian and the fact that `KNITRO` requests the calculation of the Hessian less than a third as often as it requires the objective function evaluation we conclude that including the Hessian is beneficiary.

Table 2.2: Average execution time of the acquisition function, its gradient and Hessian when running BO in the Eggholder function on an Intel E5-2640v3 CPU. For batch size 40, QEI fails, i.e. it always returns 0 without any warning message. For the execution time of the Hessian we assume knowledge of the solution of (P). Its timing is split into two parts as described in the main text. Note that these results present a qualitative picture as OEI and QEI are coded in different languages and use different underlying libraries.

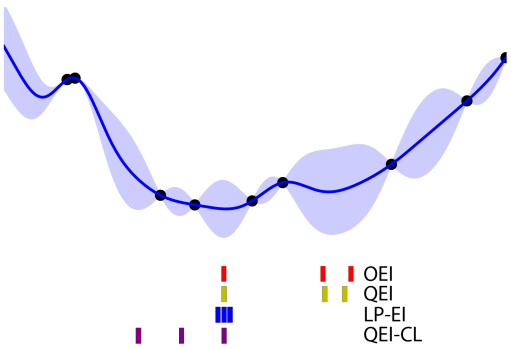
Batch Size	QEI: $\alpha(X), \nabla\alpha(X)$	OEI: $\bar{\alpha}(X), \nabla\bar{\alpha}(X)$	$\nabla^2\bar{\alpha}(X)$	
		Solve (P)	$\partial\bar{M}$	`Tensorflow` Part
2	$5.6 \cdot 10^{-3}$	$2.1 \cdot 10^{-4}$	$5.3 \cdot 10^{-4}$	$1.0 \cdot 10^{-2}$
3	$1.2 \cdot 10^{-2}$	$3.8 \cdot 10^{-4}$	$7.5 \cdot 10^{-4}$	$1.4 \cdot 10^{-2}$
6	$1.1 \cdot 10^{-1}$	$1.5 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	$2.0 \cdot 10^{-2}$
10	1.1	$8.2 \cdot 10^{-3}$	$5.9 \cdot 10^{-3}$	$3.2 \cdot 10^{-2}$
20	$2.1 \cdot 10^{1}$	$4.7 \cdot 10^{-2}$	$2.3 \cdot 10^{-2}$	$8.7 \cdot 10^{-2}$
40	–	$3.4 \cdot 10^{-1}$	$1.4 \cdot 10^{-1}$	$3.5 \cdot 10^{-1}$

We next present simulation results to demonstrate the performance of OEI in various scenarios.

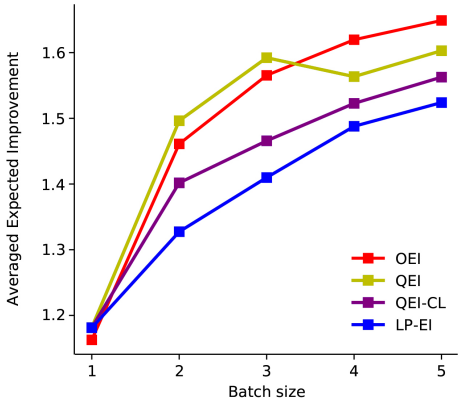
2.3.1 Perfect modeling assumptions

We first demonstrate that the competing Expected Improvement based algorithms can produce clearly suboptimal choices in a simple 1–dimensional example. We consider a 1–d Gaussian Process on the interval $[-1, 1]$ with a squared exponential kernel [13] of lengthscale $1/10$, variance 10, noise 10^{-6}[2] and a mean function $m(x) = (5x)^2$. An example posterior of 9 observations is depicted in Figure 2.1(a). Given the GP and the 9 observations, we depict the optimal 3-point batch chosen by maximizing each acquisition function. Note that in this case we assume perfect modeling assumptions – the GP is completely known and representative of the actual process. We can observe in Figure 2.1(a) that the OEI choice is very close to the one proposed by QEI while being slightly more explorative, as OEI allows for the possibility of more exotic distributions than the Gaussian.

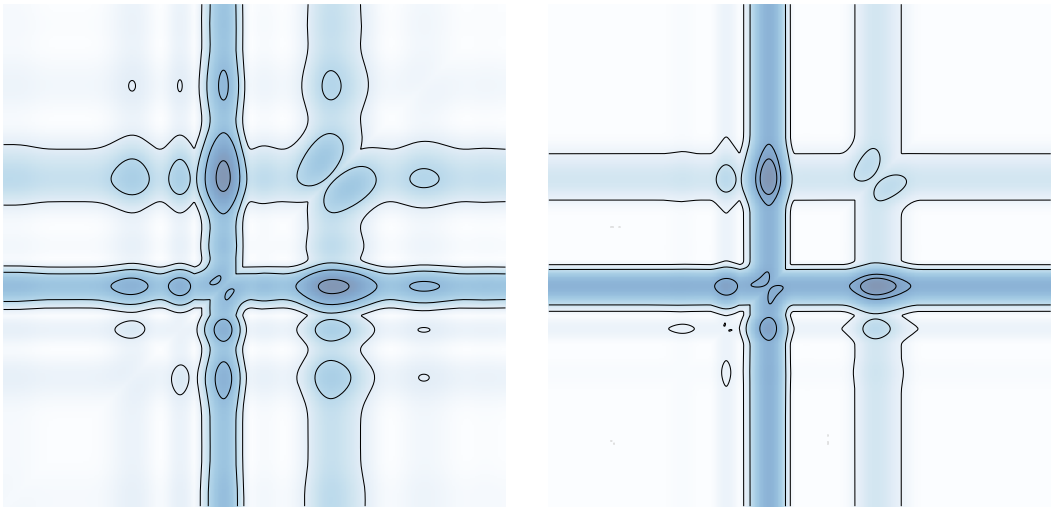
[2] Although in our theoretical analysis we assume a noiseless objective, we include a small, non-zero noise in the implementation for numerical purposes [13, Appendix A.2].



(a) Suggested 3-point batches of different algorithms for a GP posterior, depicted on $[-1, 1]$, given 10 observations. The thick blue line depicts the GP mean, the light blue shade the uncertainty intervals ($\pm$ sd) and the black dots the observations. The locations of the batch chosen by each algorithm are depicted with colored vertical lines at the bottom of the figure.



(b) Averaged one-step Expected Improvement on 200 GP posteriors of sets of 10 observations with the same generative characteristics (kernel, mean, noise) as the one in Figure 2.1 for different algorithms across varying batch size.



(c) Contour plots of evaluating 2-point batch selections on the GP posterior of Figure 2.1(a) across $[-1, 1]^2$ with OEI (left) and QEI (right). OEI closely approximates QEI in the sense that their optimizers nearly coincide and their level sets are similar. Note that the level sets of the two figures correspond to different values.

Figure 2.1: Simulation results on Gaussian Process draws.

In contrast the LP-EI heuristic samples almost at the same point all three times. This can be explained as follows: LP-EI is based on a Lipschitz criterion to penalize areas around previous choices. However, the Lipschitz constant for this function is dominated by the end points of the function (due to the quadratic trend), which is clearly not suitable for the area of the minimizer (around zero), where the Lipschitz constant is approximately zero. On the other hand, QEI-CL favors suboptimal regions. This is because QEI-CL postulates outputs equal to the mean value of the observations which significantly alter the GP posterior.

Testing the algorithms on 200 different posteriors, generated by creating sets of 10 observations drawn from the previously defined GP, suggests OEI as the clear winner. For each of the 200 posteriors, each algorithm chooses a batch, the performance of which is evaluated by sampling the multipoint Expected Improvement (2.2). The averaged results are depicted in Figure 2.1(b). For a batch size of 1 all of the algorithms perform the same, except for OEI which performs slightly worse due to the relaxation of Gaussianity. For batch sizes 2-3, QEI is the best strategy, while OEI is a very close second. For batch sizes 4-5 OEI performs significantly better. Figure 2.1(c) explains the very good performance of OEI. Although always different from the sampled estimate, it closely approximates the actual Expected Improvement in the sense that their optimizers and level sets are in close agreement. The deterioration of the performance for QEI in Figure 2.1(b) on batch sizes 4 and 5 might be related with software issues of the `R` package `DiceOptim` as there appear to exist points that `DiceOptim`'s results of the multipoint Expected Improvement differ considerably from sampled estimates.

2.3.2 Synthetic functions

Next, we evaluate the performance of OEI in minimizing synthetic benchmark functions. The functions considered are: the Six-Hump Camel function defined on $[-2, 2] \times [-1, 1]$, the Hartmann 6 dimensional function defined on $[0, 1]^6$ and the Eggholder function, defined on $[-512, 512]^2$. We compare the performance of OEI against QEI, LP-EI and BLCB as well as random uniform sampling. The

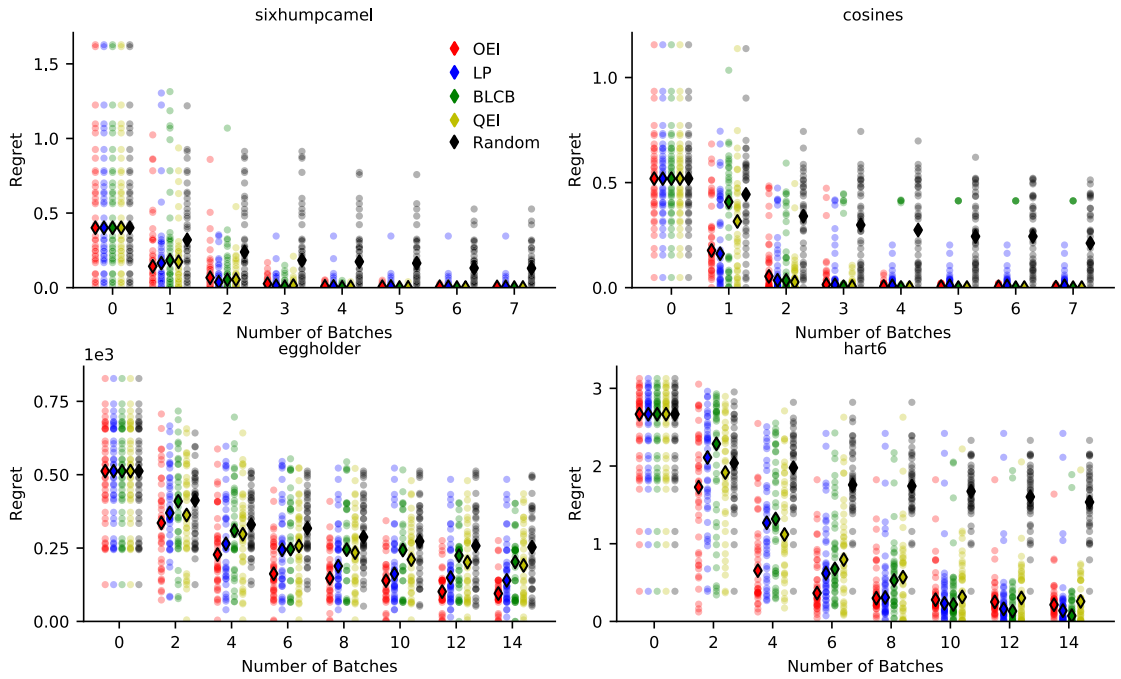


Figure 2.2: BO of *batch size 5* on synthetic functions. Red, blue, green, yellow and black dots depict runs of OEI, LP-EI, BLCB, QEI and Random algorithms respectively. Diamonds depict the median regret for each algorithm. Regret is defined as the difference of the best evaluation of each algorithm and the global minimum of the function under minimization.

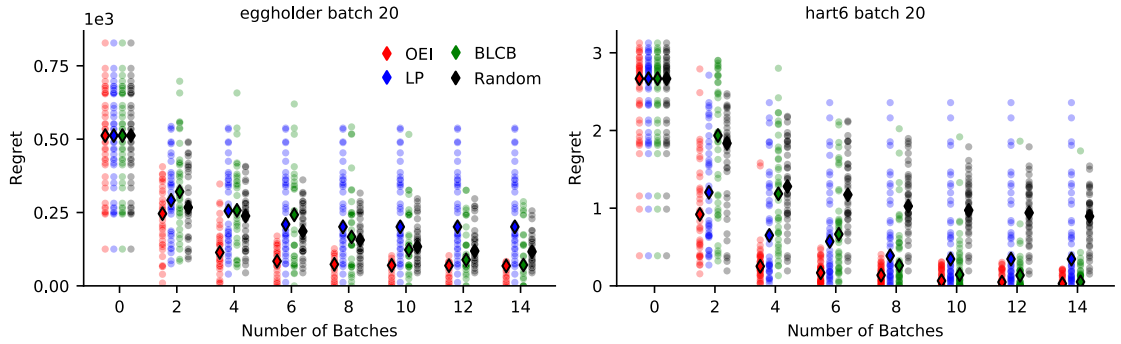


Figure 2.3: BO of *batch size 20* on the challenging Hartmann 6-d Eggholder 2-d functions where OEI exhibits clearly superior performance. Red, blue, green and black dots depict runs of OEI, LP-EI, BLCB and Random algorithms respectively. Diamonds depict the median regret for each algorithm. Regret is defined as the difference of the best evaluation of each algorithm and the global minimum of the function under minimization. Compare the above results with Figure 2.2 for the same runs with a smaller batch size. Note that QEI is not included in this case, as it does not scale to large batch sizes (see Table 2.2).

initial dataset consists of 10 random points for all the functions. A Matern 3/2 kernel is used for the GP modeling [13]. As all of the considered functions are noiseless, we set the likelihood variance to a fixed small number 10^{-6} for numerical

reasons. For the purpose of generality, the input domain of every function is scaled to $[-0.5, 0.5]^n$ while the observation dataset y_d is normalized at each iteration, such that $\text{Var}[y^{\mathcal{D}}] = 1$. The same transformations are applied to QEI, LP-EI and BLCB for reasons of consistency. All the acquisition functions except OEI are optimized with the quasi-Newton L-BFGS-B algorithm [33] with 20 random restarts. We use point estimates for the kernel's hyperparameters obtained by maximizing the marginal likelihood via L-BFGS-B restarted on 20 random initial points.

First, we consider a small-scale scenario of *batch size 5*. The results of 40 runs of Bayesian Optimization on a mixture of Cosines, the Six-Hump Camel, Eggholder, and 6-d Hartmann functions are depicted in Figure 2.2. In these functions, OEI approaches the minimizers faster than QEI and BLCB while considerably outperforming LP-EI, which exhibits outliers with bad behavior. The explorative nature of OEI can be observed in the optimization of the Hartmann function. OEI quickly reaches the vicinity of the minimum, but then decides not to refine the solution further but explore instead the rest of the 6-d space. Increasing the *batch size* to *20* for the challenging Eggholder and Hartmann functions shows a further advantage for OEI. Indeed, as we can observe in Figure 2.3, OEI successfully exploits the increased batch size. BLCB also improves its performance though not to the extent of OEI. In contrast, LP-EI fails to manage the increased batch size. This is partially expected due to the heuristic based nature of LP-EI: the Lipschitz constant estimated by LP-EI is rarely suitable for all the 20 locations of the suggested batch. Even worse, LP-EI's performance is *deteriorated* as compared to smaller batch sizes. LP-EI is plagued by numerical issues in the calculation of its gradient, and suggests multiple re-evaluations of the same points. This multiple re-samplinging affects the GP modeling, resulting in an inferior overall BO performance.

2.3.3 Real world example: Tuning a Reinforcement Learning Algorithm on various tasks

Finally we perform Bayesian Optimization to tune Proximal Policy Optimization (PPO), a state-of-the-art Deep Reinforcement Learning algorithm that has been

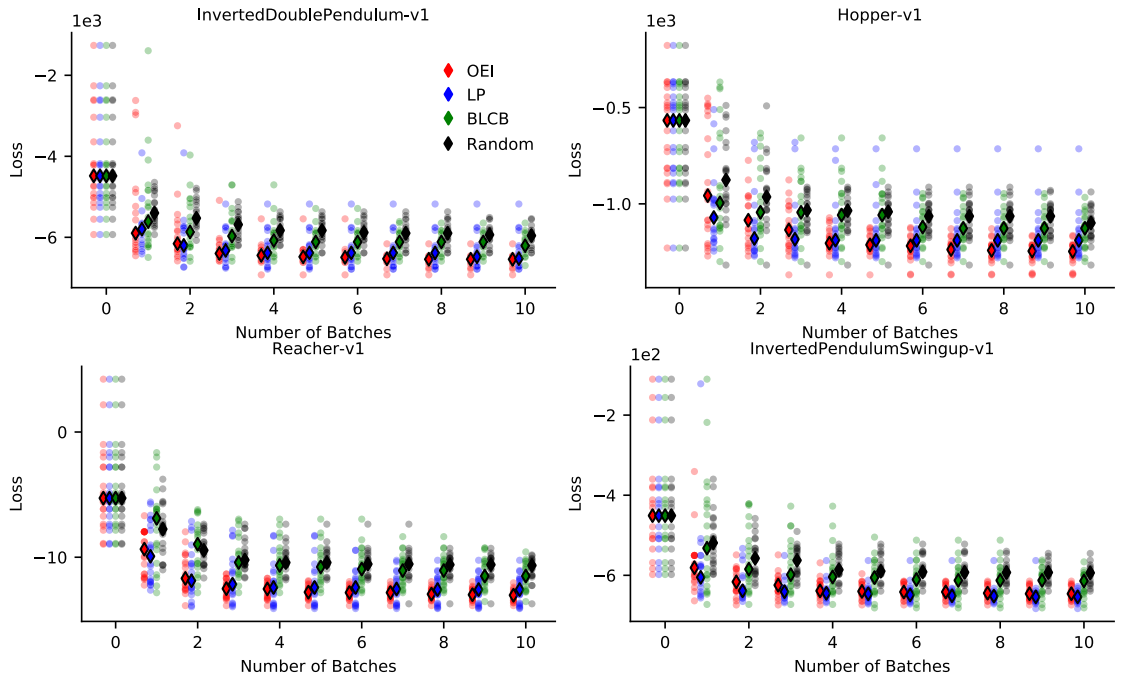


Figure 2.4: BO of batch size 20 on tuning PPO on a variety of robotic tasks. Red, blue, green and black dots depict runs of OEI, LP-EI, BLCB and Random algorithms respectively. Diamonds depict the median loss of all the runs for each algorithm.

shown to outperform several policy gradient reinforcement learning algorithms [34]. The problem is particularly challenging, as deep reinforcement learning can be notoriously hard to tune, without any guarantees about convergence or performance. We use the implementation [35] published by the authors of PPO and tune the reinforcement algorithm on 4 `OpenAI Gym` tasks (Hopper, InvertedDoublePendulum, Reacher and InvertedPendulumSwingup) using the `Roboschool` robot simulator. We tune a set of 5 hyper-parameters which are listed in Table 2.3. We define as objective function the negative average reward per episode over the entire training period ($4 \cdot 10^5$ timesteps), which favors fast learning [34]. All of the other parameters are the same as the original implementation [34].

We run Bayesian Optimization with batch size of 20, with the same modeling, preprocessing and optimization choices as the ones used in the benchmark functions. The results of 20 runs are depicted in Figure 2.4. OEI outperforms BLCB, which performs comparably to Random search, and LP-EI, which exhibits high variance with occasional outliers stuck in inferior solutions.

Table 2.3: List of PPO's Hyperparameters used for tuning. Items with asterisk are tuned in the log-space.

Hyperparameter	Range
Adam step-size	$[10^{-5}, 10^{-3}]^*$
Clipping parameter	$[0.05, 0.5]$
Batch size	$24, \ldots, 256$
Discount Factor (γ)	$1 - [10^{-3}, 10^{-3/2}]^*$
GAE parameter (λ)	$1 - [10^{-2}, 10^{-1}]^*$

2.4 Conclusions

We have introduced a new acquisition function that is a tractable, probabilistic relaxation of the multi-point Expected Improvement, drawing ideas from the Distributionally Ambiguous Optimization community. Novel theoretical results allowed inexpensive calculation of first and second derivative information resulting in efficient Bayesian Optimization on large batch sizes. Further directions include applying our distributionally agnostic approach to non-Gaussian setups e.g. to Bayesian Optimization with Student t-processes [36], scaling to larger batch sizes by exploiting recent advances in first-order convex optimization techniques [26], testing performance in noisy setups and examining the asymptotic properties of OEI. Finally, we believe that the distributionally ambiguous techniques used in this chapter can be applied to other machine learning fields such as Gaussian Process based Reinforcement Learning [37].

2.A Proofs

2.A.1 Value of the Expected Improvement Lower Bound

In this section we provide a proof of the first of our main results, Theorem 2.2.1, which establishes that for any $\Sigma \succ 0$ one can compute the value of our optimistic upper bound function

$$\sup_{\mathbb{P}\in\mathcal{P}(\mu,\Sigma)} \mathbb{E}_{\mathbb{P}}[g(\xi)] \tag{2.9}$$

via solution of a convex optimization problem in the form of a semidefinite program.

Proof of Theorem 2.2.1:

Recall that the set $\mathcal{P}(\mu, \Sigma)$ is the set of all distributions with mean μ and covariance Σ. Following the approach of [23, Lemma 1], we first remodel problem (2.9) as:

$$\begin{array}{lll} -\inf\limits_{\nu\in\mathcal{M}_+} & \displaystyle\int_{\mathbb{R}^k} -g(\xi)\nu(d\xi) & \\ \text{s.t.} & \displaystyle\int_{\mathbb{R}^k} \nu(d\xi) = 1 & \\ & \displaystyle\int_{\mathbb{R}^k} \xi\nu(d\xi) = \mu & \\ & \displaystyle\int_{\mathbb{R}^k} \xi\xi^T\nu(d\xi) = \Sigma + \mu\mu^T, & \end{array} \tag{2.10}$$

where $\mathcal{M}_+$ represents the cone of nonnegative Borel measures on $\mathbb{R}^k$. The optimization problem (2.10) is a semi-infinite linear program, with infinite dimensional decision variable ν and a finite collection of linear equalities in the form of moment constraints. As such, it belongs to the well-studied family of moment problems [38].

As shown by [23], the dual of problem (2.10) has instead a finite dimensional set of decision variables and an infinite collection of constraints, and can be written as

$$\begin{array}{ll} -\sup & \langle \Omega, M\rangle \\ \text{s.t.} & \left[\xi^T\ 1\right] M \left[\xi^T\ 1\right]^T \leq -g(\xi) \quad \forall \xi \in \mathbb{R}^k, \end{array} \tag{2.11}$$

with $M \in \mathbb{S}^{k+1}$ the decision variable and $\Omega \in \mathbb{S}^{k+1}$ the second order moment matrix of ξ (see (2.5)). Strong duality holds between problems (2.10) and (2.11) for any $\Omega \succ 0 \Leftrightarrow \Sigma \succ 0$, i.e., there is zero duality gap and their optimal values coincide.

The dual decision variables in (2.11) form a matrix M of Lagrange multipliers for the constraints in (2.10) that is block decomposable as

$$M = \begin{pmatrix} M_{11} & m_{12} \\ m_{12}^T & m_{22} \end{pmatrix},$$

where $M_{11} \in \mathbb{S}^k$ are multipliers for the second moment constraint, $m_{12} \in \mathbb{R}^k$ multipliers for the mean value constraint, and m_{22} a scalar multiplier for the constraint that $\nu \in \mathcal{M}_+$ should integrate to 1, i.e., that ν should be a probability measure.

For our particular problem, we have:

$$\begin{aligned} -g(\xi) &= \min(\xi_{(1)}, \ldots, \xi_{(k)}, \underline{y^{\mathcal{D}}}) - \underline{y^{\mathcal{D}}} \\ &= \min(\xi_{(1)} - \underline{y^{\mathcal{D}}}, \ldots, \xi_{(k)} - \underline{y^{\mathcal{D}}}, 0), \end{aligned}$$

as defined in (2.4), so that (2.11) can be rewritten as

$$\begin{aligned} -\sup \quad & \langle \Omega, M \rangle \\ \text{s.t.} \quad & \begin{bmatrix} \xi^T & 1 \end{bmatrix} M \begin{bmatrix} \xi^T & 1 \end{bmatrix}^T \leq \xi_{(i)} - \underline{y^{\mathcal{D}}}, \qquad \forall \xi \in \mathbb{R}^k \\ & \begin{bmatrix} \xi^T & 1 \end{bmatrix} M \begin{bmatrix} \xi^T & 1 \end{bmatrix}^T \leq 0 \qquad i = 1, \ldots, k. \end{aligned} \tag{2.12}$$

The infinite dimensional constraints in (2.12) can be replaced by the equivalent conic constraints

$$M - \begin{bmatrix} 0 & e_i/2 \\ e_i^T/2 & -\underline{y^{\mathcal{D}}} \end{bmatrix} \preceq 0 \qquad i = 1, \ldots, k,$$

and $M \preceq 0$ respectively, where e_i are the standard basis vectors in $\mathbb{R}^k$. Substituting the above constraints in (2.12) results in (P), which proves the claim.

2.A.2 Gradient of the Expected Improvement Lower Bound

In this section we provide a proof of our second main result, Theorem 2.2.2, which shows that the gradient[3] $\partial p / \partial \Omega$ of our upper bound function (2.3) with respect to Ω coincides with the optimal solution of the semidefinite program (P). We will find it useful to exploit also the dual of this SDP, which we can write as

$$\begin{aligned} -\inf \quad & \sum_{i=1}^{k} \langle Y_i, C_i \rangle \\ \text{s.t.} \quad & Y_i \succeq 0, \quad i = 0, \ldots, k \\ & \sum_{i=0}^{k} Y_i = \Omega, \end{aligned} \tag{2.13}$$

[3]Technically, the gradient is not defined, since Ω is by definition symmetric. We can get around this technicality by a slight abuse of notation allowing for a non-symmetric Ω such that $\Omega + \Omega^T \in \mathbb{S}_{++}^{k+1}$.

and the Karush-Kuhn-Tucker conditions for the pair of primal and dual solutions $\bar{M},\ \{\bar{Y}_i\}$:

$$C_i - \bar{M} \succeq 0 \tag{2.14}$$

$$\bar{Y}_i \succeq 0 \tag{2.15}$$

$$\langle \bar{Y}_i, \bar{M} - C_i \rangle = 0 \Rightarrow \bar{Y}_i(\bar{M} - C_i) = 0 \tag{2.16}$$

$$\left.\frac{\partial L(M,\Omega)}{\partial M}\right|_{\bar{M}} = 0 \Rightarrow \sum_{i=0}^{k} \bar{Y}_i = \Omega, \tag{2.17}$$

where L denotes the Lagrangian of (P).

Before proving Theorem 2.2.2 we require three ancillary results. The first of these results establishes that any feasible point M for the optimization problem (P) has strictly negative definite principal minors in the upper left hand corner.

Lemma 2.A.1. *For any feasible $M \in \mathbb{S}^{k+1}$ of (P) the upper left $k \times k$ matrix M_{11} is negative definite.*

Proof. Let

$$M = \begin{bmatrix} M_{11} & m_{12} \\ m_{12}^T & m_{22} \end{bmatrix},$$

where $M_{11} \in \mathbb{S}^k, m_{12} \in \mathbb{R}^k$ and $m_{22} \in \mathbb{R}$.

From (2.6) we can infer that $m_{22} + \underline{y^{\mathcal{D}}} < 0$, otherwise (2.6) would require $m_{12} + e_i/2 = 0\ \forall i = 1, \dots k$, which is impossible.

Since $M \preceq 0$, we have $m_{22} \leq 0$. Assume though, for now, that $m_{22} < 0$. Applying then a standard Schur complement identity in (2.6) results in:

$$M_{11} \preceq (m_{12} - e_i)(m_{22} + \underline{y^{\mathcal{D}}})^{-1}(m_{12} - e_i)^T$$

$$M_{11} \preceq m_{12} m_{22}^{-1} m_{12}^T \quad i = 1, \dots, k.$$

Summing the above results in

$$M_{11} \preceq \frac{(m_{22} + \underline{y^{\mathcal{D}}})^{-1}}{k+1} \sum_{i=1}^{k} (m_{12} - e_i)(m_{12} - e_i)^T + \frac{m_{22}^{-1}}{k+1} m_{12} m_{12}^T,$$

which results in $M_{11} \prec 0$, since $\operatorname{span}\Big(\{m_{12}, \{m_{12} - e_i\}_{i=i,\dots,k}\}\Big) \supseteq \operatorname{span}\Big(\{m_{12} - m_{12} + e_i\}_{i=i,\dots,k}\}\Big) = \mathbb{R}^k$.

Finally, in the case where $m_{22} = 0$ we have $m_{12} = 0$, since $M \preceq 0$. Applying the above results in $M_{11} \preceq (m_{22} + \underline{y^{\mathcal{D}}})^{-1}/k \sum_{i=1}^{k} \frac{1}{k} e_i e_i^T \prec 0$. □

The second auxiliary results lists some useful properties of the dual solution:

Lemma 2.A.2. *The optimal Lagrange multipliers of* (P) *are of rank one with* $\mathcal{R}(\bar{Y}_i) = \mathcal{N}(\bar{M} - C_i)$, $\forall i = 0, \ldots, k$, *where* $\mathcal{N}(\cdot)$ *and* $\mathcal{R}(\cdot)$ *denote the nullspace and the range of a matrix.*

Proof. Lemma 2.A.1 implies that $[x^T 0](\bar{M} - C_i)[x^T 0]^T = [x^T 0]\bar{M}[x^T 0]^T < 0, \forall x \in \mathbb{R}^k$ (recall that C_i is nonzero only in the last column or the last row), which means that $\operatorname{rank}(\bar{M} - C_i) \geq k$. Due to the complementary slackness condition (2.16), the span of $\bar{Y}_i$ is orthogonal to the span of $\bar{M} - C_i$ and consequently $\operatorname{rank}(\bar{Y}_i) \leq 1$. However, according to (2.17) we have

$$\operatorname{rank} \sum_{i=0}^{k} \bar{Y}_i = \operatorname{rank}(\Omega) \overset{\Omega \succ 0}{\Longrightarrow} \sum_{i=0}^{k} \operatorname{rank}(\bar{Y}_i) \geq k + 1$$

which results in

$$\operatorname{rank}(\bar{M} - C_i) = k, \quad \operatorname{rank}(\bar{Y}_i) = 1,$$

and, using (2.16):

$$\mathcal{R}(\bar{Y}_i) = \mathcal{N}(\bar{M} - C_i), \quad i = 0, \ldots, k.$$

□

Our final ancillary result considers the (directional) derivative of the function p when its argument is varied linearly along some direction $\bar{\Omega}$.

Lemma 2.A.3. *Given any* $\bar{\Omega} \in \mathbb{S}^{k+1}$ *and any moment matrix* $\Omega \in \mathbb{S}^{k+1}_{++}$, *define the scalar function* $q(\cdot\,; \Omega) : \mathbb{R} \to \mathbb{R}$ *as*

$$q(\gamma; \Omega) := p(\Omega + \gamma\bar{\Omega}).$$

Then $q(\cdot\,; \Omega)$ *is differentiable at* 0 *with* $\partial q(\gamma; \Omega)/\partial\gamma|_{\gamma=0} = -\langle \bar{\Omega}, \bar{M}(\Omega) \rangle$, *where* $\bar{M}(\Omega)$ *is the optimal solution of* (P) *at* Ω.

Proof. Define the set Γ_Ω as

$$\Gamma_\Omega := \{\gamma \mid \gamma \in \operatorname{dom} q(\cdot\,; \Omega)\} = \left\{\gamma \;\middle|\; \left(\Omega + \gamma\bar{\Omega}\right) \in \operatorname{dom} p\right\},$$

i.e., the set of all γ for which problem (P) has a bounded solution given the moment matrix $\Omega + \gamma\bar{\Omega}$. In order to prove the result it is then sufficient to show:

(i) $0 \in \operatorname{int} \Gamma_\Omega$, and

(ii) The solution of (P) at Ω is unique.

The remainder of the proof then follows from [39, Lemma 3.3], wherein it is shown that $0 \in \operatorname{int} \Gamma_\Omega$ implies that $-\langle \bar{\Omega}, \bar{M}(\Omega) \rangle$ is a subgradient of $q(\cdot\,; \Omega)$ at 0, and subsequent remarks in [39] establish that uniqueness of the solution $M(\Omega)$ ensure differentiability.

We will now show that both of the conditions (i) and (ii) above are satisfied.

(i): Proof that $0 \in \operatorname{int} \Gamma_\Omega$*:*

It is well-known that if both of the primal and dual problems (P) and (2.13) are strictly feasible then their optimal values coincide, i.e., Slater's condition holds and we obtain strong duality; see [3, Section 5.2.3] and [40].

For (P) it is obvious that one can construct a strictly feasible point. For (2.13), $Y_i = \Omega/(k+1)$ defines a strictly feasible point for any $\Omega \succ 0$. Hence (P) is solvable for any $\Omega + \gamma\bar{\Omega}$ with γ sufficiently small. As a result, $0 \in \operatorname{int} \Gamma_\Omega$.

(ii): Proof that the solution to (P) *at* Ω *is unique:*

We will begin by showing that the range of the dual variables $\mathcal{R}(Y_i), i = 0, \dots k$ remain the same for every primal-dual solution. Assume that there exists another optimal primal-dual pair denoted by $\tilde{M} = \bar{M} + \delta M$, and $\tilde{Y}_i$. Due to Lemma 2.A.2, there exist $\bar{y}_i, \tilde{y}_i \in \mathbb{R}^{k+1}$ such that

$$\bar{Y}_i = \bar{y}_i \bar{y}_i^T, \quad \tilde{Y}_i = \tilde{y}_i \tilde{y}_i^T \quad \forall i = 0, \dots, k. \tag{2.18}$$

Obviously, $\tilde{y}_i \in \mathcal{R}(\tilde{Y}_i) = \mathcal{N}(\tilde{M} - C_i)$ and, by definition, we have

$$\tilde{y}_i^T (\tilde{M} - C_i) \tilde{y}_i = 0 \quad i = 0, \dots, k. \tag{2.19}$$

Moreover, as $\bar{M}$ is feasible we have $\tilde{y}_i^T (\bar{M} - C_i) \tilde{y}_i \leq 0$, resulting in

$$\tilde{y}_i^T \delta M \tilde{y}_i \geq 0, \quad i = 0, \dots, k. \tag{2.20}$$

Since $\bar{M}$ and $\tilde{M}$ have the same objective value we conclude that $\langle \Omega, \delta M \rangle = 0$. Moreover, according to (2.17) and (2.18) we can decompose Ω as $\sum_{i=0}^{k} \tilde{y}_i \tilde{y}_i^T$. Hence

$$\begin{aligned}
\text{Trace}(\Omega \delta M) = 0 \Longrightarrow \quad & \text{Trace}(\delta M \sum_{i=0}^{k} \tilde{y}_i \tilde{y}_i^T) = 0 \Longrightarrow \quad \sum_{i=0}^{k} \text{Trace}(\delta M \tilde{y}_i \tilde{y}_i^T) = 0 \\
\Longrightarrow \quad & \sum_{i=0}^{k} \tilde{y}_i^T \delta M \tilde{y}_i = 0 \overset{(2.20)}{\Longrightarrow} \quad \tilde{y}_i^T \delta M \tilde{y}_i = 0 \quad \forall i = 0, \ldots, k \\
\overset{(2.19)}{\Longrightarrow} \quad & \tilde{y}_i (\bar{M} - C_i) \tilde{y}_i^T = 0 \quad \forall i = 0, \ldots, k.
\end{aligned}$$

Hence, $\tilde{y}_i$ is, like $\bar{y}_i$, a null vector of $\bar{M} - C_i$. Since the null space of $\bar{M} - C_i$ is of rank one, we get $\tilde{y}_i = \lambda_i \bar{y}_i$ for some $\lambda_i \in \mathbb{R}$, resulting in, $\bar{Y}_i = \lambda_i^2 \tilde{Y}_i$.

Now we can show that the dual solution is unique. Assume that $\bar{Y}_i \neq \tilde{Y}_i$, i.e. $\lambda_i^2 \neq 1$ for some $i = \{1, \ldots, k\}$. Feasibility of $\tilde{Y}_i$ and $\bar{Y}_i$ gives

$$\left.\begin{aligned} \sum_{i=0}^{k} \bar{Y}_i = \Omega \Leftrightarrow \sum_{i=0}^{k} \lambda_i^2 \tilde{Y}_i = \Omega \\ \sum_{i=0}^{k} \tilde{Y}_i = \Omega \end{aligned} \right| \Rightarrow \sum_{i=0}^{k} (1 - \lambda_i^2) \tilde{Y}_i = 0,$$

i.e. $\{\mathcal{R}(\tilde{Y}_i)\}$ are linearly dependent. This contradicts Lemma 2.A.2 and (2.17) which suggest linear independence, as each $\mathcal{R}(Y_i)$ is of rank one and together they span the whole space $\mathbb{R}^{k+1}$. Hence, $\bar{Y}_i = \tilde{Y}_i \; \forall i = \{0, \ldots, k\}$, i.e. the dual solution is unique.

Finally, the uniqueness of the primal solution can be established by the uniqueness for the dual solution. Indeed, summing (2.16) gives

$$\sum_{i=0}^{k} \bar{Y}_i (\bar{M} - C_i) = 0 \overset{(2.17)}{\Leftrightarrow} \Omega \bar{M} = \sum_{i=0}^{k} \bar{Y}_i C_i \Leftrightarrow \bar{M} = \Omega^{-1} \sum_{i=0}^{k} \bar{Y}_i C_i.$$

□

Proof of Theorem 2.2.2:

Given the preceding support results of this section, we are now in a position to prove Theorem 2.2.2.

First, we will show that $p : \mathbb{S}_{++}^{k+1} \to \mathbb{R}$ is differentiable on its domain. First, note that p is concave due to [41, Corollary 3.32] and hence continuous on $\text{int}(\text{dom } p) = S_{++}^{k+1}$ [41, Theorem 2.35]. Also, note that due to Lemma 2.A.3 the regular directional

derivatives [41, Theorem 8.22] of p exist and are a linear map of the direction. Hence, according to [41, Theorem 9.18 (a, f)] p is differentiable on $\mathbb{S}^{k+1}_{++}$.

Consider now the derivative of the solution of (P) when perturbing Ω across a specific direction $\bar{\Omega}$, i.e. $\partial q(\gamma;\Omega)/\partial\gamma$ with $q(\gamma;\Omega) = p(\Omega + \gamma\bar{\Omega})$. Lemma 2.A.3 shows that $\partial q(\gamma;\Omega)/\partial\gamma|_0 = -\langle\bar{\Omega}, \bar{M}\rangle$ when $\Omega \succ 0$. The proof then follows elementwise from Lemma 2.A.3 by choosing $\bar{\Omega}$ a sparse symmetric matrix with $\bar{\Omega}_{(i,j)} = \bar{\Omega}_{(j,i)} = 1/2$ the only nonzero elements.

2.A.3 Derivative of the Optimal Solution

In this section we will provide a constructive proof of Theorem 2.2.3, and show in particular that $\dot{\bar{M}}$, the directional derivative of $\bar{M}(\Omega)$ when perturbing Ω linearly across a direction $\bar{\Omega} \in \mathbb{S}^{k+1}$, can be computed by solution of the following linear system

$$\begin{bmatrix} \bar{S}_0 & & 0 & (\bar{y}_0^T \otimes I)\bar{H} \\ & \ddots & & \vdots \\ 0 & & \bar{S}_k & (\bar{y}_k^T \otimes I)\bar{H} \\ H(\bar{y}_0 \oplus \bar{y}_0) & \dots & H(\bar{y}_0 \oplus \bar{y}_0) & 0 \end{bmatrix} \begin{bmatrix} \dot{\bar{y}}_0 \\ \vdots \\ \dot{\bar{y}}_k \\ \mathrm{vec_u}(\dot{\bar{M}}) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \mathrm{vec_u}(\bar{\Omega}) \end{bmatrix},$$

where

- $\bar{S}_i = \bar{M}_i - C_i, \quad i = 0, \dots, k$
- y_i is defined such that $\bar{Y}_i = \bar{y}_i \bar{y}_i^T$, i.e. the non-zero eigenvector of the Lagrange multiplier $\bar{Y}_i$, which was shown to be unique in Lemma (2.A.2).
- $\mathrm{vec}(\cdot)$, is the operator that stacks the columns of a matrix to a vector, and $\mathrm{vec_u}(\cdot)$ the operator that stacks only the upper triangular elements in a similar fashion.
- H, is the matrix performing the map $\mathrm{vec}(Z) \to \mathrm{vec_u}(Z)$ where $Z \in \mathbb{S}$, and $\bar{H}$ a matrix that performs the inverse operation.
- $\otimes$ and $\oplus$ denote the Kronecker product and sum respectively.

Proof. Lemma 2.A.3 of Appendix 2.A.2, guarantees that the solutions of (P) and (2.13) are unique for any $\Omega \succ 0$. Hence, according to [42], the directional derivatives $\dot{\bar{M}}, \dot{\bar{Y}}_i$ of $\bar{M}, \bar{Y}$ along the perturbation $\bar{\Omega}$ exist and are given as the unique solution to the following overdetermined system:

$$\begin{aligned} \sum_{i=0}^{k} \dot{\bar{Y}}_i &= \bar{\Omega} \\ \dot{\bar{Y}}_i \bar{S}_i - \bar{Y}_i \dot{\bar{M}} &= 0 \\ \dot{\bar{M}}, \dot{\bar{Y}}_i &\in \mathbb{S}^{k+1} \quad i = 0, \dots, k. \end{aligned} \tag{2.21}$$

The above linear system is over-determined, and has symmetric constraints. This renders standard solution methods, such as LU decomposition, inapplicable. Expressing the above system in a standard matrix form results in a matrix with $O(k^4)$ zeros, which makes its solution very costly.

To avoid theses issues, we will exploit Lemma 2.A.2 of Appendix 2.A.2 to express the dual solution $\bar{Y}_i$ compactly as $\bar{Y}_i = \bar{y}_i \bar{y}_i^T$. One can choose a differentiable mapping $\bar{Y}_i(t) \to \bar{y}_i(t)$, e.g. $\bar{y}_i(t) = \sqrt{\lambda_i(t)} u_i(t)$ where $\lambda_i(t)$ is the only positive eigenvalue of $\bar{Y}_i(t)$ and $u_i(t)$ its corresponding unit-norm eigenvector. Differentiability of $\bar{y}_i(t)$ comes from differentiability of $\bar{Y}_i(t)$, $\lambda_i(t)$, $u_i(t)$ [43] and positivity of $\lambda_i(t)$ due to Lemma 2.A.2. The chain rule then applies for $\dot{\bar{Y}}_i = \dot{\bar{y}}_i \bar{y}_i^T + \bar{y}_i \dot{\bar{y}}_i^T$. Hence (2.21) can be expressed as

$$\sum_{i=0}^{k} \dot{\bar{y}}_i \bar{y}_i^T + \bar{y}_i \dot{\bar{y}}_i^T = \bar{\Omega} \tag{2.22}$$

$$(\dot{\bar{y}}_i \bar{y}_i^T + \bar{y}_i \dot{\bar{y}}_i^T) \bar{S}_i - \bar{y}_i \bar{y}_i^T \dot{\bar{M}} = 0, \quad i = 0, \dots, k. \tag{2.23}$$

Exploiting $\bar{y}_i^T \bar{S}_i = 0$ from (2.16) and that $y_i \neq 0$ gives

$$(2.23) \Leftrightarrow \bar{y}_i (\dot{\bar{y}}_i^T \bar{S}_i - \bar{y}_i^T \dot{\bar{M}}) = 0 \overset{y_i \neq 0}{\Longleftrightarrow} \dot{\bar{y}}_i^T \bar{S}_i - \bar{y}_i^T \dot{\bar{M}}, \quad i = 0, \dots, k \tag{2.24}$$

We can express equations (2.22) and (2.24) into the standard matrix form by using the vec operator and the identity $\mathrm{vec}(AXB) = (B^T \otimes A)\,\mathrm{vec}(X)$, which gives

$$\begin{aligned} \sum_{i=0}^{k} (\bar{y}_i \otimes I + I \otimes \bar{y}_i) \dot{\bar{y}}_i &= \mathrm{vec}(\bar{\Omega}) \\ S_i \dot{\bar{y}}_i - (\bar{y}_i^T \otimes I) \mathrm{vec}(\dot{\bar{M}}) &= 0, \quad i = 0, \dots, k. \end{aligned}$$

Finally, eliminating the symmetric constraint via $\mathrm{vec_u}(\cdot)$, H and H^+ gives:

$$\begin{aligned} \sum_{i=0}^{k} H(\bar{y}_i \otimes I + I \otimes \bar{y}_i)\dot{\bar{y}}_i &= \mathrm{vec_u}(\bar{\Omega}) \\ S_i\dot{\bar{y}}_i - (\bar{y}_i^T \otimes I)H^+\mathrm{vec_u}(\dot{\bar{M}}) &= 0, \quad i = 0, \dots, k, \end{aligned} \tag{2.25}$$

leading to the suggested linear system. The system is square and non-singular since it is equivalent to (2.21) which has a unique solution.

Finally, it remains to show that $M(\Omega)$ is a differentiable for any $\Omega \succ 0$. First note that M is outer semicontinuous [41, Definition 5.4] on $\mathbb{S}^{k+1}_{++}$ as, according to Theorem (2.2.2), p is continuous on $\mathbb{S}^{k+1}_{++}$. Since M is unique on $\mathbb{S}^{k+1}_{++}$ it is also continuous. Finally, note that due to Equation (2.25) the regular directional derivatives [41, Theorem 8.22] of M exist and are a linear map of the direction. Hence, according to [41, Theorem 9.18 (a, f)] M is differentiable on $\mathbb{S}^{k+1}_{++}$. □

2.A.4 Construction of worst and best-case distributions

In this appendix we construct distributions in $\mathcal{P}(\mu, \Sigma)$ that achieve the bounds of (2.3). First, regarding the lower bound $\inf_{\mathbb{P}\in\mathcal{P}(\mu,\Sigma)} \mathbb{E}_{\mathbb{P}}[g(\xi)]$, where g is the *convex piecewise affine function* defined by (2.4), the following Proposition implies that it is trivial, in the sense that it is independent of the variance Σ.

Proposition 2.A.4. *For any convex piecewise affine function* $\max_{i=1,\dots,l}(a_i + b_i^T\xi)$ *the worst case expectation* $\inf_{\mathbb{P}\in\mathcal{P}(\mu,\Sigma)} \mathbb{E}_{\mathbb{P}}[\max_{i=1,\dots,l}(a_i + b_i^T\xi)]$ *is equal to* $\max_{i=1,\dots,l}(a_i + b_i^T\mu)$.

Proof. First note that the worst case expectation is bounded below

$$\mathbb{E}\Big[\max_{i=1,\dots,l}(a_i + b_i^T\xi)\Big] \geq \max_{i=1,\dots,l} \mathbb{E}[a_i + b_i^T\xi] = \max_{i=1,\dots,l}(a_i + b_i^T\mu). \tag{2.26}$$

We will construct a sequence of parametric distributions that, in the limit, achieve the above mentioned lower bound.

Assume the one-dimensional, uncorrelated random variables z, w

$$z \sim \mathcal{U}\Big(-\frac{1}{\epsilon}, \frac{1}{\epsilon}\Big), \quad w \sim \mathcal{N}(0, \epsilon),$$

i.e. z is uniformly distributed in $(-\epsilon^{-1}, \epsilon^{-1})$, and w is a zero mean Gaussian with variance ϵ, where $\epsilon \in \mathbb{R}_{++}$.

Now, assuming $0 < \epsilon \leq \sqrt{\frac{1}{3}}$, consider the random variable x with the mixture distribution

$$x = \begin{cases} z & \text{with probability } 3\epsilon^2 \\ w & \text{with probability } 1 - 3\epsilon^2. \end{cases}$$

Since both of the mixing distributions are zero mean, the resulting distribution is zero mean with variance

$$\mathbb{E}(x^2) = 3\epsilon^2\mathbb{E}(z^2) + (1 - 3\epsilon^2)\mathbb{E}(w^2) = 1 + \epsilon(1 - 3\epsilon^2).$$

In the limit $\epsilon \to 0$ the random variable x has zero mean and variance one, but its probability distribution function is infinitesimal everywhere outside the origin.

Assuming that y is a vector of independent variables distributed identically to x for $\epsilon \to 0$, the random vector $\xi = \Sigma^{1/2}y + \mu$ has covariance matrix Σ and mean value μ, with its probability distribution being infinitesimal everywhere expect in μ. For this random vector (2.26) is tight. □

Thus, according to Proposition 2.A.4 the worst case expectation $\inf_{\mathbb{P}\in\mathcal{P}(\mu,\Sigma)} \mathbb{E}_{\mathbb{P}}[g(\xi)]$ does not depend on Σ and is equal to $\underline{y^{\mathcal{D}}} - \min(\mu_1, \ldots, \mu_k, \underline{y^{\mathcal{D}}})$.

Next, we will construct a distribution that achieves the upper bound of (2.3). In particular we will show that for any $\Omega \succ 0$ the optimal distribution $\bar{\mathbb{P}}_{\mu,\Sigma}(\Omega)$ of problem (P) is discrete and can be given in closed form by the solution of (2.13), i.e. the dual of (P). To show this, we will need the following ancillary result:

Proposition 2.A.5. *For any $\Omega \succ 0$ we have $\langle \bar{Y}_i(\Omega), C_i \rangle < 0 \ \forall i = 1, \ldots, k$ where $\bar{Y}_i(\Omega)$ is the dual solution of (P).*

Proof. Assume that there exists $i \in \{1, \ldots, k\}$ such that $\langle \bar{Y}_i, C_i \rangle \geq 0$. Then, the following set of dual variables

$$\begin{aligned} \tilde{Y}_0 &= \bar{Y}_0 + \bar{Y}_i \\ \tilde{Y}_i &= 0 \\ \tilde{Y}_j &= \bar{Y}_j, \quad j \in \{1, \ldots, k\} \cap j \neq i, \end{aligned} \tag{2.27}$$

is also optimal, as $\sum_{i=1}^{k}\langle\tilde{Y}_i, C_i\rangle = \sum_{i\neq j}\langle\bar{Y}_i, C_i\rangle \leq \sum_{i=1}^{k}\langle\bar{Y}_i, C_i\rangle$. This is a contradiction to Lemma (2.A.3). □

The following Theorem will now construct the best-case distribution:

Theorem 2.A.6. *For any $\Omega \in \mathbb{S}_{++}^{k+1}$ the probability distribution $\bar{\mathbb{P}}(\Omega)$ that minimizes* (2.12) *is discrete with probability mass function*

$$f_y(\xi) = \begin{cases} p_i, & \xi = \xi_i, \qquad i = 0, \ldots, k \\ 0, & \text{otherwise,} \end{cases} \tag{2.28}$$

where the points ξ_i and the probability masses p_i are uniquely defined by the solution of (2.13) *and $\Omega = \left[\begin{smallmatrix} \Sigma+\mu\mu^T & \mu \\ \mu^T & 1 \end{smallmatrix}\right]$ as following*

$$\bar{Y}_i = p_i \begin{bmatrix} \xi_i \\ 1 \end{bmatrix} \begin{bmatrix} \xi_i^T & 1 \end{bmatrix}, \quad i = 1, \ldots k$$
$$p_0 = 1 - \sum_{i=1}^{k} p_i \quad \xi_0 = \mu - \sum_{i=1}^{k} p_i \xi_i.$$

Proof. Consider the dual of (2.11) defined in (2.12) which can be reformulated as follows:

$$\begin{aligned} -\sup \quad & \langle \Omega, M \rangle \\ \text{s.t.} \quad & \begin{bmatrix} \xi^T & 1 \end{bmatrix} (M - C_i) \begin{bmatrix} \xi^T & 1 \end{bmatrix}^T \leq 0 \\ & \forall \xi \in \mathbb{R}^k, \quad i = 0, \ldots k. \end{aligned} \tag{2.29}$$

According to Lemma 2.A.2, $\mathcal{N}(\bar{M} - C_i) = 1 = \mathcal{R}(\bar{Y}_i),\ i = 0, \ldots, k$. Hence, at optimality, the infinite dimensional constraint of (2.29) is active in at most $k+1$ points. Due to slack complementarity between the primal-dual pair (2.11)-(2.12) these are the only points for which the optimal probability measure of (2.11) is nonzero. As a result, the optimal distribution $\bar{P}_{\mu,\Sigma}$ is discrete with at most $k+1$ points.

Denote as ξ_i the possible outcomes of the optimal probability distribution $\bar{P}_{\mu,\Sigma}$, with ξ_i corresponding to the i-th constraint of (2.28). Each ξ_i has a nonzero probability if and only if there exists a (unique) $\begin{bmatrix} \xi_i^T & 1 \end{bmatrix} \in \mathcal{N}(\bar{M} - C_i) = \mathcal{R}(\bar{Y}_i)$. This is always the case for $i = 1, \ldots, k$ as otherwise the last row and column of Y_i

would be zero and, due to the special structure of C_i, we would get $\langle \bar{Y}_i, C_i \rangle = 0$ contradicting Proposition 2.27. Decomposing $\bar{Y}_i$ as

$$\bar{Y}_i = p_i \begin{bmatrix} \xi_i \\ 1 \end{bmatrix} \begin{bmatrix} \xi_i^T & 1 \end{bmatrix}, \quad i = 1, \dots k$$

results in the following dual optimal cost

$$-\sum_{i=1}^{k} \langle \bar{Y}_i, C_i \rangle = -\sum_{i=1}^{k} p_i (\xi_{i(i)} - \underline{y_d}),$$

from which we can verify that the suggested probability distribution (2.28) achieves the optimal cost. Uniqueness of the probability distribution comes from uniqueness and linear independence of $\bar{Y}_i$ (Lemma (2.A.3)).

Finally, it is worth noting that a similar result that concerns a broader class of distributionally robust optimization problems is given in [44, Theorem 5.1] which shows that the optimal distribution $\bar{\mathbb{P}}_{\mu,\Sigma}$ has a discrete support. However, [44] does not show how to construct the optimal distribution nor does he prove that it consists of exactly $k + 1$ points. □

3

Efficient Semidefinite Programming with approximate ADMM

3.1 Introduction

The scaling of the SDP relaxation introduced in the previous chapter was dependent on the limitations of state-of-the-art semidefinite solvers. In this chapter we introduce a novel approach that can lead to significant speedups when solving semidefinite programs with ADMM. Semidefinite Programming is of central importance in many scientific fields. Areas as diverse as kernel-based learning [45], dimensionality reduction [46] analysis and synthesis of state feedback policies of linear dynamical systems [47], sum of squares programming [48] and fluid mechanics [49] rely on Semidefinite Programming as a crucial component. Further examples include optimal power flow [50] and the distributionally ambiguous optimization that was used in the machine learning application of chapter 2.

The wide adoption of Semidefinite Programming was facilitated by reliable algorithms that can solve semidefinite problems with polynomial worst case complexity [3]. For small to medium sized problems it is widely accepted that primal-dual Interior Point methods are efficient and robust and often the method of choice. Several open-source solvers, like `SDPT3` [51] and `SDPA` [52], as well as the commercial solver `MOSEK` [25] exist that follow this approach. However, the limitations of

interior point methods become evident in large problems, as each iteration requires factorizations of large "Hessian matrices". First-order methods avoid this bottleneck thus scaling better to large problems, with the ability to provide modest-accuracy solutions for many large scale problems of practical interest.

We will focus on the Alternating Directions Method of Multipliers (ADMM), a popular first-order algorithm that has been the method of choice for several popular optimization solvers both for Semidefinite Programming [30], [31], [26] and other types of convex optimization problems such as Quadratic Programming (QP) [53]. Following an initial factorization of an $m \times m$ matrix, every iteration of ADMM entails a solution of a linear system via forward/backward substitution and a projection to the Semidefinite Cone. Typically, in SDPs, the projection operation takes the majority (sometimes 90% or more) of the solution time. Thus, reducing the per-iteration time of ADMM is directly linked to computing projections in a time-efficient manner.

The projection of a symmetric matrix $n \times n$ matrix A to the Semidefinite Cone is defined as

$$\Pi_{\mathbb{S}_+}(X) := \arg\min_X \|A - X\|_F,$$

and it can be computed in "closed form" as a function of the eigendecomposition of X. Indeed, assuming $\begin{bmatrix} V_+ & V_- \end{bmatrix} \begin{bmatrix} \Lambda_+ & \\ & \Lambda_- \end{bmatrix} \begin{bmatrix} V_+ & V_- \end{bmatrix} := X$ where V_+ (V_-) is an orthonormal matrix containing the positive (nonnegative) eigenvectors and Λ_+ (Λ_-) is a diagonal matrix that contains the positive (nonnegative) eigenvalues of A then

$$\Pi_{\mathbb{S}_+}(A) = V_+ \Lambda_+ V_+^T = A - V_- \Lambda_- V_-^T. \tag{3.1}$$

Thus the computation of $\Pi_{\mathbb{S}_+}$ entails the (partial) eigendecomposition of A followed by a scaled matrix-matrix product.

The majority of optimization solvers, e.g. `SCS` [30] and `COSMO.jl` [26], calculate $\Pi_{\mathbb{S}_+}$ by computing the full eigendecomposition using `LAPACK`'s `syevr` routine[1]. There are two important limitations associated with computing full-eigendcompositions.

[1]Detailed in https://software.intel.com/mkl-developer-reference-c-syevr

Namely, eigendecomposition has cubic complexity with respect to the matrix size n [1, §8], and it cannot be warm started. This has prompted research on methods for the approximate computation of a few eigenpairs in an iterative fashion [54], [55], [56]. This has been associated with a set of relevant software such as the widely used `ARPACK` [57], and the more recent `BLOPEX` [58] and `PRIMME` [59]. The reader can find surveys of relevant software in [60] and [59, §2]

However, the use of iterative eigensolvers in the Semidefinite optimization community has been very limited. To the best of our knowledge, approximate eigensolvers have not been used inside an ADMM algorithm in any widely available implementation. In the wider area of first-order methods, [61, §3.1] considered `ARPACK` but disregarded it on the basis that it does not allow efficient warm starting suggesting that it should only be used when the problem is known a priori to have low rank. Wen's suggestion of using `ARPACK` for SDPs whose solution are expected to be low rank has been demonstrated recently by [62]. At every iteration, [62] uses `ARPACK` to compute the r largest eigenvalues/vectors and then uses the approximate projection $\tilde{\Pi}(A) = \sum_{i=1}^{r} \max(\lambda_i, 0) v_i v_i^T$. The projection error can then be bounded by

$$\left\|\Pi(A) - \tilde{\Pi}(A)\right\|_F^2 = \left\| \sum_{i=r+1}^{n} \max(\lambda_i, 0) v_i v_i^T \right\|_F^2 \leq (n-r) \max(\lambda_r, 0)^2.$$

The parameter r is chosen in a decreasing manner across iterations so that the projection errors are summable. The summability of the projection errors is important, as it has been shown to ensure convergence of averaged non-expansive operators [63, Proposition 5.34] and for ADMM in particular [7, Theorem 8].

However, the analysis of [62] depends on the assumption that the iterative eigensolver will indeed compute the r largest eigenpairs "exactly". This is both practically and theoretically problematic, because the computation of eigenvectors is numerically unstable, as it depends on the spectral gap (defined as the distance between the corresponding eigenvalue and its nearest neighboring eigenvalue; refer to §3.5.2 for a concise definition), and therefore no useful bounds can be given when repeated eigenvalues exist.

In contrast, our approach relies on a novel bound that characterizes the projection accuracy independently of the spectral gaps, depending only on the residual norms. The derived bounds do not require that the eigenpairs have been computed "exactly", but hold for *any* set of approximate eigenpairs obtained via the Rayleigh-Ritz process. This allows us to compute the eigenpairs with a relatively loose tolerance while still retaining convergence guarantees. Furthermore, unlike [62], our approach has the ability of warm-starting of the eigensolver, which typically results in speedups.

On the theoretical side, we extend recent results regarding the detection of primal or dual infeasibility. It is well known that if the problem is infeasible, then the iterates of ADMM will diverge [7]. This is true even when the iterates of ADMM are computed approximately with summable approximation errors. Hence, infeasibility can be detected in principle by stopping the ADMM algorithm when the iterates exceed a certain bound. This is, however, unreliable both in practice, because it depends on the choice of the bound, and theoretically, because it does not provide certificates of infeasibility [3]. Recently, [8] has shown that the successive differences of ADMM's iterates, which always converge regardless of feasibility, can be used to reliably detect infeasibility and construct infeasibility certificates. This approach has been used successfully in the optimization solver `OSQP` [53]. We extend Banjac's results to show that they hold even when ADMM's iterates are computed approximately, under the assumption that the approximation errors are summable.

3.2 Approximate ADMM

Although the focus of this chapter is on Semidefinite Programming, our analysis holds for more general convex optimization problems that allow for combinations of Semidefinite Problems, Linear Programs (LPs), Quadratic Programs (QPs), Second Order Cone Programs (SOCPs) among others. In particular, the problem form we consider is defined as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & Ax = z \\ & z \in \mathcal{C}, \end{array} \tag{P}$$

where $x \in \mathbb{R}^k$ and $z \in \mathcal{C}$ are the decision variables, $P \in \mathbb{S}^k_+$, $q \in \mathbb{R}^k$, $A \in \mathbb{R}^{m \times n}$ and $\mathcal{C}$ is a nonempty, closed convex, set. The set $\mathcal{C}$ is restricted to be the Cartesian product of a convex compact set and a translated closed convex cone.

We suggest solving (P), i.e. finding a solution $(\bar{x}, \bar{z}, \bar{y})$ where $\bar{y}$ is a Lagrange multiplier for the equality constraint of (P), with an approximate version of ADMM, described in Algorithm 3.1. As expected from an ADMM algorithm, our Algorithm

Algorithm 3.1: Solving (P) with approximate ADMM

1 **given** initial values x^0, y^0, z^0, parameters $\rho > 0, \sigma > 0, \alpha \in (0, 2)$, the summable sequences $(\mu^k)_{k\in\mathbb{N}}, (\nu^k)_{k\in\mathbb{N}}$ and $x \approx_\epsilon y$ denoting that the vectors x, y satisfy $\|x - y\| \leq \epsilon$; **for** $k = 0, \ldots$ *until convergence* **do**

2 $\quad \begin{bmatrix} \tilde{x}^{k+1} \\ \tilde{z}^{k+1} \end{bmatrix} \approx_{\mu^k} \begin{bmatrix} P + \sigma I & \rho A^T \\ \rho A & -\rho I \end{bmatrix} \backslash \left(\begin{bmatrix} \sigma I & \rho A^T \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x^k \\ z^k - y^k/\rho \end{bmatrix} - \begin{bmatrix} q \\ 0 \end{bmatrix} \right)$;

3 $\quad x^{k+1} = \alpha \tilde{x}^{k+1} + (1 - \alpha) x^k$;

4 $\quad z^{k+1} \approx_{\nu^k} \Pi_{\mathcal{C}}(\alpha \tilde{z}^{k+1} + (1 - \alpha) z^k + y^k/\rho)$;

5 $\quad y^{k+1} = y^k + \rho(\alpha \tilde{x}^{k+1} + (1 - \alpha) z^k - z^{k+1})$

6 **end**

consists of repeated solutions of linear systems (Line 3) and projections to $\mathcal{C}$ (Line 5). These are the drivers of efficiency of ADMM. Usually, these operations are computed to machine precision via matrix factorizations. Indeed, Algorithm 3.1 has been first introduced by [53] in the absence of approximation errors. However, "exact" computations can be prohibitively expensive for large problems (and indeed impossible in finite-precision arithmetic), and the user may have to rely on approximate methods for their computation. For example, [6, §4.3] suggests using the Conjugate Gradient method for approximately solving the linear systems embedded in ADMM. In Section 3.5, we suggest specific methods for the approximation computation of ADMM steps with a focus in the operation of line 5. Before moving into particular methods, we will first discuss the convergence properties of Algorithm 3.1.

Indeed, our analysis explicitly accounts for approximation errors and provides convergence guarantees, either to solutions or certificates of infeasibility, in their presence. In general when ADMM's steps are computed approximately, ADMM might lose its convergence properties. Indeed, when the approximation errors

are not controlled appropriately, the Fejér monotonicity [63] of the iterates and any convergence rates of ADMM lost. In the worst case, the iterates diverge. However, the following Theorem, which constitutes the main theoretical result of this chapter, shows that Algorithm 3.1 converges to a solution, or a certificate of infeasibility of (P) due to the requirement that the approximation errors are summable across the Algorithm's iterations.

Theorem 3.2.1. *Consider the iterates x^k, z^k, and y^k of Algorithm 3.1. If a KKT point exists for* (P)*, then $\lim_{k\to\infty}(x^k, z^k, y^k)$ converges to a KKT point, i.e. a solution of* (P)*. Otherwise, the successive differences*

$$\delta x := \lim_{k\to\infty} x^{k+1} - x^k, \quad \textit{and} \quad \delta y := \lim_{k\to\infty} y^{k+1} - y^k$$

still converge and can be used to detect infeasibility as follows:

(i) If $\delta y \neq 0$ then (P) *is primal infeasible and δy is a certificate of primal infeasibility [8, Proposition 3.1] in that it satisfies*

$$A^T \delta y = 0 \quad \textit{and} \quad S_{\mathcal{C}}(\delta y) < 0 \tag{3.2}$$

(ii) If $\delta x \neq 0$ then (P) *is dual infeasible and δx is a certificate of dual infeasibility [8, Proposition 3.1] in that it satisfies*

$$P\delta x = 0, \quad A\delta x \in \mathcal{C}^\infty, \quad \textit{and} \quad q^T \delta x < 0 \tag{3.3}$$

(iii) If both $\delta x \neq 0$ and $\delta y \neq 0$ then (P) *is both primal and dual infeasible and $\delta x, \delta y$ are certificates of primal and dual infeasibility as above.*

In order to prove Theorem 3.2.1 we first have to discuss some key properties of ADMM. This will provide the theoretical background that will allow us to present the proof in Chapter 3.4. Then, in Chapter 3.5 we will discuss particular methods for the approximate computation of ADMM's steps that can lead to significant speedups.

3.3 The asymptotic behaviour of approximate ADMM

In this section we present ADMM in a general setting, express it as an iteration over an averaged operator, and then consider its convergence when this operator is computed only approximately.

ADMM is used to solve split optimization problems of the following form

$$\begin{array}{ll} \text{minimize} & f(\chi) + g(\psi) \\ \text{subject to} & \chi = \psi \end{array} \tag{S}$$

where χ, ψ denote the decision variables on $\mathbb{R}^l$ which is equipped with an inner product induced norm $\|\cdot\| = \langle \cdot, \cdot \rangle$. The functions $f : \mathbb{R}^l \to [-\infty, +\infty]$, $g : \mathbb{R}^l \to [-\infty, +\infty]$ are proper, lower-semicontinuous, and convex.

ADMM works by alternatingly minimizing the augmented Lagrangian of (S), defined as

$$L(\chi, \psi, \omega) := f(\chi) + g(\psi) + \langle \omega, \chi - \psi \rangle + \frac{1}{2}\|\chi - \psi\|^2 \tag{3.4}$$

over χ and ψ. That is, ADMM consists of the following iterations

$$\chi^{k+1} = \arg\min_{\chi} L(\chi, \psi^k, \omega^k) \tag{ADMM1}$$
$$\psi^{k+1} = \arg\min_{\psi} L(\bar{\chi}^{k+1}, \psi, \omega^k) \tag{ADMM2}$$
$$\omega^{k+1} = \omega^k + (\bar{\chi}^{k+1} - \psi^{k+1}) \tag{ADMM3}$$

where $\bar{\chi}^{k+1}$ is a relaxation of χ^{k+1} with $\bar{\chi}^{k+1} = \alpha\chi^{k+1} + (1-\alpha)\psi^k$ for some relaxation parameter $\alpha \in (0, 2)$.

Although (ADMM1)-(ADMM3) are useful for implementing ADMM, theoretical analyses of the algorithm typically consider ADMM as an iteration over an averaged operator. To express ADMM in operator form, note that (ADMM1) and (ADMM2) can be expressed via the *proximity operators* [63, §24]

$$\mathrm{prox}_f(\phi) := \arg\min_{\chi} \Big(f(\chi) + \frac{1}{2}\|\chi - \phi\|^2 \Big), \tag{3.5}$$

and the similarly defined $prox_g$, as

$$\chi^{k+1} = \mathrm{prox}_f(\psi^k - \omega^k), \quad \psi^{k+1} = \mathrm{prox}_g(\bar{\chi}^{k+1} + \omega^k)$$

respectively. Now, using the *reflections* of prox_f and prox_g, i.e. $R_f := 2\text{prox}_f - Id$ and $R_g := 2\text{prox}_g - Id$, we can express ADMM as an iteration over the following $\frac{1}{2}\alpha$-averaged operator (see Appendix 3.A for details)

$$T := \Big(1 - \frac{1}{2}\alpha\Big)Id + \frac{1}{2}\alpha R_f R_g \tag{3.6}$$

on the variable $\phi^k := \bar{\chi}^k + \omega^{k-1}$. The variables ψ, χ, ω of (ADMM1)-(ADMM3) can then be obtained from ϕ as

$$\chi^{k+1} = \text{prox}_f R_g \phi^k, \quad \psi^k = \text{prox}_g \phi^k, \quad \text{and} \quad \omega^k = (Id - \text{prox}_g)\phi^k. \tag{3.7}$$

We are interested in the convergence properties of ADMM when the operators $\text{prox}_f, \text{prox}_g$, and thus T, are computed inexactly. In particular, we consider that the iterates are generated as

$$(\forall k \in \mathbb{N}) \quad \phi^{k+1} = \Big(1 - \frac{1}{2}\alpha\Big)\phi_k + \frac{1}{2}\alpha\Big(R_f\Big(R_g\phi^k + \epsilon_g^k\Big) + \epsilon_f^k\Big) \tag{3.8}$$

for some error sequences $\epsilon_f^k, \epsilon_g^k \in \mathbb{R}^l$. Our convergence results will depend on an assumption that $\left\|\epsilon_f^k\right\|$ and $\left\|\epsilon_g^k\right\|$ are summable. This implies that ϕ^k can be considered as an approximate iteration over T, i.e.

$$\phi^{k+1} \approx_{\epsilon^k} T\phi^k, \tag{3.9}$$

for some summable error sequence (ϵ^k). Indeed, since R_g and R_f are nonexpansive, we have

$$\begin{aligned}\left\|R_f\Big(R_g\phi^k + \epsilon_g^k\Big) + \epsilon_f^k - R_f R_g \phi^k\right\| &\le \left\|R_g\phi^k + \epsilon_g^k - R_g\phi^k\right\| + \left\|\epsilon_f^k\right\| \\ &\le \left\|\epsilon_f^k\right\| + \left\|\epsilon_g^k\right\|,\end{aligned}$$

or $\left\|\phi^{k+1} - T\phi^k\right\| \le \alpha\left\|\epsilon_f^k\right\|/2 + \alpha\left\|\epsilon_g^k\right\|/2$, from which the summability of ϵ^k follows easily.

It is well known that, when $\left\|\epsilon_f^k\right\|, \left\|\epsilon_g^k\right\|$ are summable, (3.8) converges to a solution of (S), obtained by ϕ according to (3.7), provided that (S) has a KKT point [7, Theorem 8]. We will show that, under the summability assumption, $\delta\phi = \lim \phi^{k+1} - \phi^k$ always converges, regardless whether (S) has a KKT point:

Theorem 3.3.1. *The successive differences* $\delta\phi^k = \phi^{k+1} - \phi^k$ *of* (3.8) *converge to the minimal element of* $\operatorname{cl}\mathcal{R}(Id - T)$ *provided that* $\sum\left\|\epsilon_f^k\right\| < \infty$ *and* $\sum\left\|\epsilon_g^k\right\| < \infty$.

Proof. This is a special case of Proposition 3.B.1 of Appendix 3.B. □

Theorem 3.3.1 will prove useful in detecting infeasibility, as we will show in the following section.

3.4 Proof of Theorem 3.2.1

We now turn our attention to proving Theorem 3.2.1. To this end, note that (P) can be considered as a special case of (S) [8] and [53]. This becomes clear if we set $\chi = (\tilde{x}, \tilde{z}), \psi = (x, z)$,

$$f(\tilde{x}, \tilde{z}) := \frac{1}{2}\tilde{x}^T P\tilde{x} + q^T\tilde{x} + \mathcal{I}_{A\tilde{x}=\tilde{z}}(\tilde{x}, \tilde{z}), \tag{3.10}$$

$$g(x, z) := \mathcal{I}_{\mathcal{C}}(z), \tag{3.11}$$

where $\mathcal{I}_{\mathcal{C}}(z)$ denotes the indicator function of $\mathcal{C}$. Furthermore, according to the analysis of the previous section, and using the norm

$$\|(x, z)\| = \sqrt{\sigma\|x\|_2^2 + \rho\|z\|_2^2}. \tag{3.12}$$

we get that Algorithm 3.1 is equivalent to iteration (3.8).

First, we show that if (P) has a KKT point then Algorithm 3.1 converges to its primal-dual solution. Due to (3.10), (3.11) and (3.12) every KKT point $(\bar{x}, \bar{z}, \bar{y})$ of (P) gives a KKT point

$$(\bar{\chi}, \bar{\psi}, \bar{\omega}) = ((\bar{x}, \bar{z}), (\bar{x}, \bar{z}), (0, \bar{y}/\rho)) \tag{3.13}$$

for (S) and, vice versa, every KKT point of (S) is in the form of (3.13) (right) and gives a KKT point $(\bar{x}, \bar{z}, \bar{y})$ for (P). Thus, according to [7, Theorem 8], Algorithm 3.1 converges to a KKT point of (P), assuming that a KKT point exists.

It remains to show points $(i) - (iii)$ of Theorem 3.2.1. These are a direct consequence of [8, Theorem 5.1] and the following proposition:

Proposition 3.4.1. *The following limits*

$$\delta x := \lim_{k\to\infty} x^{k+1} - x^k, \quad \delta y := \lim_{k\to\infty} y^{k+1} - y^k,$$

defined by the iterates of Algorithm 3.1, converge to the respective limits defined by the iterates of Algorithm 3.1 with $\mu^k = \nu^k = 0\ \forall k \in \mathbb{N}$.

Proof. According to Appendix 3.A we can rewrite Algorithm 3.1 as follows

$$z^k \approx_{\nu^{k-1}} \Pi_{\mathcal{C}}(\upsilon^k) \tag{3.14a}$$

$$(\tilde{x}^{k+1}, \tilde{z}^{k+1}) \approx_{\mu^k} \operatorname{prox}_f((x^k, 2z^k - \upsilon^k)) \tag{3.14b}$$

$$x^{k+1} = x^k - \alpha(\tilde{x}^{k+1} - x^k) \tag{3.14c}$$

$$\upsilon^{k+1} = \upsilon^k + \alpha(\tilde{z}^{k+1} - z^k) \tag{3.14d}$$

where $(x^k, \upsilon^k) := \phi^k$ and y^k can be obtained as $y^k = \rho(\upsilon^k - z^k)$.

Define $\delta x^k := x^{k+1} - x^k$, $\forall k \in \mathbb{N}$ and δz^k, $\delta \upsilon^k$, $\delta \tilde{x}^k$, $\delta \tilde{z}^k$ in a similar manner. Due to Theorem 3.3.1 and [8, Lemma 5.1] we conclude that $\lim_{k\to\infty} \delta x^k$ and $\lim_{k\to\infty} \delta \upsilon^k$ defined by the iterates of Algorithm 3.1 converge to the respective limits defined by the iterates of Algorithm 3.1 with $\mu^k = \nu^k = 0\ \forall k \in \mathbb{N}$.

To show the same result for δy, recall that $y^k = \rho(\upsilon^k - z^k)$, thus, it suffices to show the desired result for $\lim_{k\to\infty} \delta z^k$. We show this using similar arguments with [8, Proposition 5.1 (iv)]. Indeed, note that due to (3.14c)-(3.14d) we have

$$-(\delta x^{k+1} - \delta x^k)/\alpha = \delta x^k - \delta \tilde{x}^{k+1}$$
$$-(\delta \upsilon^{k+1} - \delta \upsilon^k)/\alpha = \delta z^k - \delta \tilde{z}^{k+1}$$

and thus $\lim_{k\to\infty} \delta x^k = \lim_{k\to\infty} \delta \tilde{x}^k$ and $\lim_{k\to\infty} \delta z^k = \lim_{k\to\infty} \delta \tilde{z}^k$. Furthermore, due to (3.10) we have $A\tilde{x}^{k+1} - \tilde{z}^{k+1} = e^k$ for some sequence (e^k) with summable norms, thus

$$\lim_{k\to\infty} \delta \tilde{z}^k = A \lim_{k\to\infty} \delta \tilde{x}^k = A \lim_{k\to\infty} \delta x^k$$

and the claim follows due to [8, Proposition 5.1 (i) and (iv)]. □

3.5 Krylov-Subspace Methods for ADMM

In this Section, we suggest suitable methods for calculating the individual steps of Algorithm 3.1. We will focus on Semidefinite Programming, i.e., when $\mathcal{C}$ is the Semidefinite Cone. After an initial presentation of state-of-the-art methods used for solving linear systems approximately, we will describe (in §3.5.1) LOBPCG, the suggested method for projecting into the semidefinite cone.

We begin with a discussion of the Conjugate Gradient method, a widely used method for the solution of the linear systems embedded in Algorithm 3.1. Through CG's presentation we will introduce the Krylov Subspace which is a critical component of LOBPCG. Finally, we will show how we can assure that the approximation errors are summable across ADMM iterations, thus guaranteeing convergence of the algorithm.

The linear systems embedded in Algorithm 3.1 are in the following form

$$\underbrace{\begin{bmatrix} P+\sigma I & \rho A^T \\ \rho A & -\rho I \end{bmatrix}}_{:=Q} \begin{bmatrix} \tilde{x}^{k+1} \\ \tilde{z}^{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \sigma I & \rho A^T \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x^k \\ z^k - y^k/\rho \end{bmatrix} - \begin{bmatrix} q \\ 0 \end{bmatrix}}_{:=b^k}. \tag{3.15}$$

The linear system (3.15) belongs to the widely explored class of symmetric quasidefinite systems [64], [65]. Standard scientific software packages, such as the Intel Math Kernel Library and the Pardiso Linear Solver implement methods that can solve (3.15) approximately. Since the approximate solution (3.15) can be considered standard in the Linear Algebra community, we will only discuss the popular class of Krylov Subspace methods, which includes the celebrated Conjugate Gradient method[2]. Although CG has been used in ADMM extensively [§4.3.4][6], [30], its presentation will be useful for introducing some basic concepts that are shared with the main focus of this section, i.e. the approximate projection to the semidefinite cone.

From the optimization perspective, Krylov subspace algorithms for solving linear systems can be considered as an improvement of gradient methods. Indeed,

[2]The Conjugate Gradient Method is only suitable for Positive Definite Linear Systems. However, (3.15) can be solved with CG via a variable reduction which yields a smaller positive linear system [65, §1].

solving $Ax = b$, where $A \in \mathbb{S}^n_{++}$ via gradient descent on the objective function $c(x) := \frac{1}{2}x^T Ax - x^T b$ amounts to the following iteration

$$(\forall k \in \mathbb{N}) \quad x^{k+1} = x^k - \beta^k \nabla c(x) = x^k - \beta^k \underbrace{(Ax^k - b)}_{:=r^k} \tag{3.16}$$

where β^k is the step size at iteration k. Note that

$$x^{k+1} \in x_0 + \underbrace{\text{span}(r_0, Ar_0, \cdots A^k r_0)}_{:=\mathcal{K}_k(A,r^0)},$$

where $\mathcal{K}_k(A, r_0)$ is known as the *Krylov Subspace*. As a result, the following algorithm,

$$(\forall k \in \mathbb{N}) \quad x^{k+1} = \underset{x \in x^0 + \mathcal{K}_k(A,r^0)}{\arg\min} \frac{1}{2}x^T Ax - x^T b \tag{CG}$$

is guaranteed to yield no-worse results than gradient descent. What is remarkable is that (CG) can be implemented efficiently in the form of two-term recurrences, resulting in the celebrated *Conjugate Gradient* (CG) Algorithm [1, §11.3].

We now turn our attention to the projection to the Semidefinite cone. The projection of a symmetric matrix A to the Semidefinite cone, i.e.

$$\Pi_{\mathbb{S}_+}(A) := \arg\min_X \|A - X\|_F,$$

is a function of another basic operation of Linear Algebra, the eigendecomposition. To see this, define

$$\begin{bmatrix} V_+ & V_- \end{bmatrix} \begin{bmatrix} \Lambda_+ & \\ & \Lambda_- \end{bmatrix} \begin{bmatrix} V_+ & V_- \end{bmatrix} := A, \tag{3.17}$$

where V_+ (V_-) is an orthonormal matrix containing the positive (nonnegative) eigenvectors and Λ_+ (Λ_-) is a diagonal matrix that contains the positive (nonnegative) eigenvalues of A. We then have

$$\Pi_{\mathbb{S}_+}(A) = V_+ \Lambda_+ V_+^T = A - V_- \Lambda_- V_-^T. \tag{3.18}$$

Thus, the projection to the semidefinite cone can be computed via either the positive or the negative eigenpairs of A. As we will see, the cost of approximating eigenpairs of a matrix depends on their cardinality, thus computing $\Pi_{\mathbb{S}_+}(A)$ with the positive

eigenpairs of A is preferable when A has mostly negative eigenvalues, and vice versa. In the following discussion we will focus on methods that compute the positive eigenpairs of A, thus assuming that A has mostly negative eigenvalues. The opposite case can be easily handled by considering $-A$.

Similarly to CG, the class of Krylov Subspace methods is very popular for the computation of "extreme" eigenvectors of an $n \times n$ symmetric matrix A and can be considered as an improvement to gradient methods. In the subsequent analysis we will make frequent use of the real eigenvalues of A, which we denote with $\lambda_1 \geq \cdots \geq \lambda_n$ and a set of corresponding orthogonal eigenvectors $\upsilon_1, \ldots \upsilon_n$. The objective to be maximized in this case is the Rayleigh Quotient,

$$r(x) := \frac{x^T A x}{x^T x}. \tag{3.19}$$

due to the fact that the maximum and the minimum values of $r(x)$ are λ_1 and λ_n respectively with υ_1 and υ_n as corresponding maximizers [1, Theorem 8.1.2]. Thus, we end up with the following gradient ascent iteration

$$\begin{aligned} (\forall k \in \mathbb{N}) \quad x^{k+1} &= \alpha^k x^k - \beta^k \nabla r(x^k) \\ &= \alpha^k x^k - 2\beta^k \Big(A x^k - r(x^k) x^k \Big) \end{aligned} \tag{3.20}$$

where the "stepsizes" α^k and β^k and the initial point x^0 are chosen so that all the iterates lie on the unit sphere. Although $r(x)$ is nonconvex, (3.20) can be shown to converge when appropriate stepsizes are used. For example, if we choose $\alpha_k = -2\beta^k r(x^k) \Rightarrow x^{k+1} \propto A x^k \ \forall k \in \mathbb{N}$, then (3.20) is simply the *Power Method*, which is known to converge linearly to an eigenvector associated with $\max |\lambda_i|$. Other stepsize choices can also assure convergence to an eigenvector associated with $\max \lambda_i$ [55, p. 11.3.4], [66, Theorem 3].

Algorithm 3.2: The Rayleigh-Ritz Procedure

1 **given** $A \in \mathbb{S}^n$ and an $n \times m$ thin matrix S that spans the trial subspace;
2 orthonormalize S;
3 $(\tilde{\Lambda}, \tilde{W}) \leftarrow$ Eigendecomposition of $S^T A S$ with $\tilde{\Lambda}_{(1,1)} \leq \cdots \leq \tilde{\Lambda}_{(m,m)}$;
4 **return** the *Ritz vectors* $S\tilde{W}$ and *Ritz values* $\tilde{\Lambda}$ of A on span(S);

Similarly to the gradient descent method for linear systems, the iterates of (3.20) lie in the Krylov subspace $\mathcal{K}_k(A, x_0)$. As a result, the following Algorithm

$$(\forall k \in \mathbb{N}) \quad x^{k+1} = \begin{array}{ll} \text{argmax} & r(x) \\ \text{subject to} & x \in \mathcal{K}_k(A, x_0) \\ & \|x\|_2 = 1, \end{array} \tag{3.21}$$

is guaranteed to yield no worse results than any variant of (3.20) in finding an eigenvector associated with $\max \lambda_i$, and in practice the difference is often remarkable. But how can the Rayleigh Quotient be maximized over a subspace? This can be achieved with the *Rayleigh-Ritz* Procedure, defined in Algorithm 3.2, which computes approximate eigenvalues/vectors (called *Ritz values/vectors*) that are restricted to lie on a certain subspace and are, under several notions, optimal [56, p. 11.4] (see discussion after Theorem 3.5.1). Indeed, every iterate x^{k+1} of (3.21) coincides with the last column of X^{k+1}, i.e. the largest Ritz vector, of the following Algorithm [56, Theorem 11.4.1]

$$(\forall k \in \mathbb{N}) \quad (\Lambda^{k+1}, X^{k+1}) = \text{Rayleigh Ritz of } A \text{ on the trial subspace } [x^0 \; Ax^0 \; \ldots \; A^k x^0]. \tag{3.22}$$

Note that unlike (3.21), Algorithm (3.22) provides approximations to not only one, but k eigenpairs, with the extremum ones exhibiting a faster rate of convergence.

Remarkably, similarly to the Conjugate Gradient algorithm, (3.22), and thus (3.21), also admit an efficient implementation in the form of three-term recurrences, known as the *Lanczos Algorithm* [1, §10.1]. In fact, the Lanczos Algorithm produces a sequence of orthonormal vectors that tridiagonalize A. Given this sequence of vectors, the computation of the associated Ritz pairs is inexpensive [1, p. 8.4]. The Lanczos Algorithm is usually the method of choice for computing a few extreme eigenpairs. However, although the Lanczos Algorithm is computationally efficient, the Lanczos process can suffer from lack of orthogonality, with the issue becoming particularly obvious when a Ritz pair is close to converging to some (usually an extremal) eigenpair [67]. Occasional re-orthogonalizations, with a cost of $O(n^2 l^k)$ where l^k is the dimension of the $k-$th trial subspace, are required to mitigate the

effects of the numerical instability. To avoid such a computational cost, the Krylov subspace is restarted or shrunk so that l^k, and thus the computational costs of re-othogonalizations, are bounded by an acceptable amount. The Lanczos Algorithm with occasional restarts is the approach employed by the popular eigensolver `ARPACK` [57] for symmetric matrices.

However, there are two limitations of the Lanczos Algorithm. Namely, it does not allow for efficient warm starting of multiple eigenvectors since its starting point is a single eigenvector, and, in finite precision, it cannot detect the multiplicity of the approximated eigenvalues as it normally provides a single approximate eigenvector for every invariant subspace of A.

Block Lanczos addresses both of these issues. Similarly to the standard Lanczos Algorithm, Block Lanczos computes Ritz pairs on the trial block Krylov Subspace $\mathcal{K}_k(A, X_0) := \text{span}(X^0, AX^0, \ldots, A^kX^0)$ where X_0 is an $n \times m$ matrix that contains a set of initial eigenvector guesses. Thus, Block Lanczos readily allows for the warm starting of multiple Ritz pairs. Furthermore, block methods handle clustered and multiple eigenvectors (of multiplicity up to m) well. However, these benefits comes at the cost of higher computational costs, as the associated subspace is increased by m at every iteration. This, in turn, requires more frequent restarts, particularly for the case where m is comparable to n.

In our experiments we observed that a single block iteration often provides Ritz pairs that give good enough projections for Algorithm 3.1. This remarkably good performance motivated us to use the *Locally Optimal Block Preconditioned Conjugate Gradient Method* (LOBPCG) which is presented in the following subsection.

3.5.1 LOBPCG: The suggested eigensolver

LOBPCG [58] is a block Krylov method that, after the first iteration, uses the trial subspace $\text{span}(X^k, AX^k, \Delta X^k)$, where $\Delta X^k := X^k - X^{k-1}$, and sets X^{k+1} to Ritz vectors corresponding to the m largest eigenvalues. Thus, the size of the trial subspace is fixed to $3m$. As a result, LOBPCG keeps its computational costs bounded and is particularly suitable for obtaining Ritz pairs of modest accuracy,

as it not guaranteed to exhibit the super-linear convergence of Block Lanczos [55] which might only be observed after a large number of iterations. Algorithm 3.3 presents LOBPCG for computing the positive eigenpairs of a symmetric matrix[3]. Note that the original LOBPCG Algorithm [58, Algorithm 5.1] is more general in the sense that it allows for the solution of generalized eigenproblems and supports preconditioning. We do not discuss these features of LOBPCG as they are not directly relevant to Algorithm 3.1. On the other hand [58] assumes that the number of desired eigenpairs is known a priori. However, this is not the case for $\Pi_{\mathbb{S}_+}$, where the computation of all *positive* eigenpairs is required.

In order to allow the computation of all the positive eigenpairs, X^k is expanded when more than m positive eigenpairs are detected in the Rayleigh-Ritz Procedure in Line 6 of Algorithm 3.3. Note that the Rayleigh-Ritz Procedure produces $3m$ approximate eigenpairs (or $2m$ in the first iteration of LOBPCG) and the number of positive Ritz values are always no more than the positive eigenvalues of A [56, p. 10.1.1], thus the subspace X^k must be expanded when more than m positive Ritz values are found.

It might appear compelling to expand the subspace to include *all* the positive Ritz pairs computed by Rayleigh-Ritz. However, this can lead to ill-conditioning, as we proceed to show. Indeed, consider the case where we perform LOBPCG starting from an initial matrix X^0. In the first iteration, Rayleigh Ritz is performed on $\text{span}(X^0, AX^0)$. Suppose that all the Rayleigh values are positive and we thus decide to include all of the Ritz vectors in X^1, setting $X^1 = [X^0 \ AX^0]W$ for some orthogonal W. In the next iteration we perform Rayleigh Ritz on the subspace spanned by

$$\begin{bmatrix} X^1 & AX^1 & \Delta X^1 \end{bmatrix} = \begin{bmatrix} X^0 & AX^0 & AX^0 & A^2X^0 & \Delta X^1 \end{bmatrix} \begin{bmatrix} W & & \\ & W & \\ & & I \end{bmatrix}.$$

The problem is that the above matrix is rank deficient, thus one has to rely on a numerically stable Algorithm, like Householder QR, for its orthonormalization

[3]Note that Algorithm 3.3 performs Rayleigh-Ritz on the subspace spanned by $[X^k \ AX^k - X^k\Lambda^k \ \Delta X^k]$. Since Λ^k is diagonal, this is mathematically the same as using $[X^k \ AX^k \ \Delta X^k]$ but using $AX^k - X^k\Lambda^k$ improves the conditioning of the Algorithm.

(required by the Rayleigh-Ritz Procedure) instead of the more efficient Cholesky QR algorithm [68, page 251]. Although, for this example, one can easily reduce columns from the matrix so that it becomes full column rank, the situation becomes more complicated when not all of the Rayleigh values are positive. In order to avoid this numerical instability, and thus be able to use Cholesky QR for othonormalizations, we expand X^k whenever necessary by a fixed size (equal to a small percentage of n (the size of A), e.g. $n/50$) with a set of randomly generated vectors.

Algorithm 3.3: The LOBPCG Algorithm for Computing the Positive Eigenpairs of a Symmetric Matrix

1 **given** $A \in \mathbb{S}^n$ and the $n \times m$ thin matrix X^0 that spans the initial trial subspace;
2 $(\Lambda^0, X^0) \leftarrow$ Rayleigh-Ritz for A on the trial subspace span(X^0);
3 $\Delta X^0 \leftarrow$ empty $n \times 0$ matrix;
4 **for** $k = 0, \ldots$ *until convergence* **do**
5 $\quad R^k \leftarrow AX^k - X^k\Lambda^k$;
6 $\quad (\Lambda^{k+1}, X^{k+1}) \leftarrow$ Apply Rayleigh-Ritz for A on the trial subspace span$(X^k, R^k, \Delta X^k)$ and return the m largest eigenpairs;
7 $\quad \Delta X^{k+1} \leftarrow X^{k+1} - X^k$;
8 $\quad$ Expand Λ^{k+1}, X^{k+1} with randomly generated elements and set $m = \text{size}(X^{k+1}, 2) = \text{size}(\Lambda^{k+1}, 2)$ if the positive Ritz values of line 6 were more than m.
9 **end**
10 **return** X^k, Λ^k containing m Ritz pairs that approximate the positive eigenpairs of A

When is projecting to $\mathbb{S}_+$ with LOBPCG most efficient? Recall that there exist two ways to project a matrix A into the semidefinite cone. The first is to compute all the positive eigenpairs Λ_+, V_+ of A and set $\Pi_{\mathbb{S}_+}(A) = V_+\Lambda_+V_+^T$. The opposite approach is to compute all the *negative* eigenpairs Λ_-, V_- of A and set $\Pi_{\mathbb{S}_+}(A) = I - V_-\Lambda_-V_-^T$. The per-iteration cost of LOBPCG is $O(n^2m)$ where m is the number of computed eigenpairs. Thus, when most of the eigenvalues are negative, then the positive eigenpairs should be approximated, and vice versa.

As a result, LOBPCG is most efficient when the eigenvalues of the matrix under projection are either almost all positive or almost all negative, in which case LOBPCG exhibits an almost quadratic complexity, instead of the cubic

complexity of the full eigendecomposition. This is the case when ADMM converges to a low rank primal or dual solution of (P). Fortunately, low rankness is often present or desirable in practical problems [69]. On the other hand, the worst case scenario is when half of the eigenpairs are negative and half positive, in which case LOBPCG exhibits worse complexity than the exact eigendecomposition and thus the latter should be preferred.

3.5.2 Error Analysis & Stopping Criteria

Algorithm 3.1 requires that the approximation errors in lines 3 and 5 are bounded by a summable sequence. As a result, bounds on the accuracy of the computed solutions are necessary to assess when the approximate algorithms (CG and LOBPCG) can be stopped.

For the approximate solution of the Linear System (3.15) one can easily devise such bounds. Indeed, note that the left hand matrix of (3.15) is fixed across iterations and is full rank. We can check if an approximate solution $[\bar{x}^{k+1};\ \bar{x}^{k+1}]$ satisfies the condition

$$\left\| \begin{bmatrix} \tilde{x}^{k+1} \\ \tilde{z}^{k+1} \end{bmatrix} - \begin{bmatrix} \bar{x}^{k+1} \\ \bar{x}^{k+1} \end{bmatrix} \right\|_2 \leq \mu^k \tag{3.23}$$

of Algorithm 3.1 easily, since (recalling Q is the KKT matrix defined in 3.15)

$$\left\| \begin{bmatrix} \tilde{x}^{k+1} \\ \tilde{z}^{k+1} \end{bmatrix} - \begin{bmatrix} \bar{x}^{k+1} \\ \bar{x}^{k+1} \end{bmatrix} \right\|_2 \leq \left\| Q^{-1} \right\| \underbrace{\left\| b^k - Q \begin{bmatrix} \bar{x}^{k+1} \\ \bar{x}^{k+1} \end{bmatrix} \right\|_2}_{:=r^k}. \tag{3.24}$$

Since $\|Q^{-1}\|$ is constant across iterations, it can be ignored when considering the summability of the approximation errors 3.23. Thus, we can terminate CG (or any other iterative linear system solver employed) when the *residual* r^k of the approximate solution $[\bar{x}^{k+1};\ \bar{x}^{k+1}]$ becomes less than a summable sequence e.g. $1/k^2$.

On the other hand, controlling the accuracy of the projection to the Semidefinite Cone requires a closer examination. Recall that, given a symmetric matrix A that is to be projected[4], our approach uses LOBCPG to compute a set of positive

[4]Note that the matrices under projection depend on the iteration number of ADMM. We do not make this dependence explicit in order to keep the notation uncluttered.

Ritz pairs $\tilde{V}, \tilde{\Lambda}$ approximating V_+, Λ_+ of (3.17) which we then use to approximate $\Pi_{\mathbb{S}^n_+}(A) = V_+\Lambda_+V_+^T$ as $\tilde{V}\tilde{\Lambda}\tilde{V}^T$[5]. A straightforward approach would be to quantify the projection's accuracy with respect to the accuracy of the Ritz pairs. Indeed, if we assume that our approximate positive eigenspace is "sufficiently rich" in the sense that $\lambda_{\max}(\tilde{V}_\perp A\tilde{V}_\perp) \leq 0$, then we get $m = \tilde{m}$ [56, Theorem 10.1.1], thus we can define $\Delta\Lambda = \Lambda_+ - \tilde{\Lambda}$, $\Delta V = V_+ - \tilde{V}$ which then gives the following bound

$$\left\|V_+\Lambda_+V_+^T - \tilde{V}\tilde{\Lambda}\tilde{V}^T\right\| \leq 2\left\|\Delta V\Lambda_+V_+^T\right\| + \left\|V_+\Delta\Lambda V_+^T\right\| + O(\|\Delta\|^2) \tag{3.25}$$

with $\|\Delta\| := \max(\|\Delta V\|, \|\Delta\Lambda\|)$. Standard results of eigenvalue perturbation theory can be used to bound the error in the computation of the eigenvalues, i.e. by $\|\Delta\Lambda\|_F^2 \leq 2\|R\|_F^2$ [56, Theorem 11.5.2][6] where

$$R := A\tilde{V} - \tilde{V}\Lambda.$$

In contrast, $\|\Delta V\|$ is ill-conditioned, as the eigenvectors are not uniquely defined in the presence of multiple (i.e. clustered) eigenvalues. At best, eigenvalue perturbation theory can give $\|\Delta V\| \lessapprox \|R\|/\text{gap}$ [70, Theorem 3.1 and Remark 3.1] where

$$\text{gap} := \min_{i,j}\left(\tilde{\Lambda}_{(i,i)} - \Lambda_{-(j,j)}\right).$$

This implies that the projection accuracy depends on the separation of the spectrum and can be very poor in the presence of small eigenvalues. Note that unlike R that is readily computable from $(\tilde{V}, \tilde{\Lambda})$, "gap" is, in general, unknown and non-trivial to compute, thus further complicating the analysis.

To overcome these issues, we derive a novel bound that shows that, although the accuracy of the Ritz pairs depends on the separation of eigenvalues, the approximate projection does not:

Theorem 3.5.1. *Assume that $\tilde{V}$ and $\tilde{\Lambda}$ are such that $\tilde{\Lambda} = \tilde{V}^TA\tilde{V}$. Then*

$$\left\|\tilde{V}\tilde{\Lambda}\tilde{V}^T - \Pi_{\mathbb{S}_+}(A)\right\|_F^2 \leq 2\|R\|_F^2 + \left\|\Pi_{\mathbb{S}_+}(\tilde{V}_\perp^TA\tilde{V}_\perp)\right\|_F^2$$

[5]When LOBPCG approximates the negative eigenspace (because the matrix under projection is believed to be almost positive definite), then all of the results of this section hold mutatis mutandis. Refer to chapter 4 for more details.

[6]Note that following [56, Theorem 11.5.1], $\lambda_{\max}(\tilde{V}_\perp A\tilde{V}_\perp) \leq 0$ implies that the indices of α can coincide with the indices of θ in [56, Theorem 11.5.2].

Proof. This is a restatement of 4.2.3 presented in the following chapter. □

Note that the above result does not depend on the assumption that $\lambda_{\max}(\tilde{V}_\perp A\tilde{V}_\perp)$ is nonpositive or that $m = \tilde{m}$. Nevertheless, with a block Krylov subspace method it is often expected that $\lambda_{\max}(\tilde{V}_\perp A\tilde{V}_\perp)$ will be either small or negative, thus the bound of Theorem 3.5.1 will be dominated by $\|R\|$. The assumption $\tilde{\Lambda} = \tilde{V}^T A\tilde{V}$ is satisfied when $\tilde{V}$ and $\tilde{\Lambda}$ are generated with the Rayleigh Ritz Procedure and thus holds for Algorithm 3.3. In fact, the use of the Rayleigh-Ritz, which is employed by Algorithm 3.3, is strongly suggested by Theorem 3.5.1 as it minimizes $\|R\|_F$ [56, Theorem 11.4.2].

We suggest terminating Algorithm 3.3 when every positive Ritz pair has a residual with norm bounded by a sequence that is summable across ADMM's iterations. Then, excluding the effect of $\left\|\Pi_{\mathbb{S}_+}(\tilde{V}_\perp^T A\tilde{V}_\perp)\right\|_F^2$, which appears to be negligible according to the results of the next section, Theorem 3.5.1 implies that the summability requirements of Algorithm 3.3 will be satisfied. Either way, the term $\left\|\Pi_{\mathbb{S}_+}(\tilde{V}_\perp^T A\tilde{V}_\perp)\right\|_F^2$ can be bounded by $(n-\tilde{m})\lambda_{\max}^2(\tilde{V}_\perp A\tilde{V}_\perp)$, where $\lambda_{\max}^2(\tilde{V}_\perp A\tilde{V}_\perp)$ can be estimated with a projected Lanczos methods.

3.6 Experiments and Software

In this section we provide numerical results for Semidefinite Programming with Algorithm 3.1, where the projection to the Semidefinite Cone is performed with Algorithm 3.3. Our implementation is essentially a modification of the optimization solver `COSMO.jl`. `COSMO.jl` is an open-source Julia implementation of Algorithm 3.1 which allows for the solution of problems in the form (P) for which $\mathcal{C}$ is a composition of translated cones $\{\mathcal{K}_i + b_i\}$. Normally, `COSMO.jl` computes ADMM's steps to machine precision and supports any cone $\mathcal{K}_i$ for which a method to calculate its projection is provided[7]. `COSMO.jl` provides default implementations for

[7]Operations for testing if a vector belongs to $\mathcal{K}_i$, its polar and its recession must be provided. These operations might be used to check for termination of the Algorithm, which, by default, is checked every 40 iterations. For the Semidefinite Cone, both of these tests can be implemented via the Cholesky factorization.

various cones, including the Semidefinite cone, where `LAPACK`'s `syevr` function is used for its projection.

We compared the default version of `COSMO.jl` with a version where the operation `syevr` for the Semidefinite Cone is replaced with Algorithm 3.3. We have reimplemented `BLOPEX`, the original `MATLAB` implementation of LOBPCG [58], in Julia. For the purposes of simplicity, our implementation supports only symmetric standard eigenproblems without preconditioning. For these problems, our implementation was tested against `BLOPEX` to assure that exactly the same results (up to machine precision) are returned for identical problems. Furthermore, according to §3.5.1 we provide the option to compute all eigenvalues that are larger or smaller than a given bound.

At every iteration k of Algorithm 3.1 we compute approximate eigenpairs of every matrix that is to be projected onto the semidefinite cone. If, at the previous iteration of ADMM, a given matrix were estimated to have less than a third of its eigenvectors positive, then LOBPCG is used to compute its *positive* eigenpairs, according to (3.18) (middle). If it had less than a third of its eigenvectors negative, then LOBPCG computes its negative eigenpairs according to (3.18) (right). Otherwise, a full eigendecomposition is used.

In every case, LOBPCG is terminated when all of the Ritz pairs has a residual with norm less than $10/k^{1.01}$. According to §3.5.2, this implies that the projection errors are summable across ADMM's iterations, assuming that the rightmost term of Theorem 3.5.1 is negligible. Indeed, in our experiments, these terms were found to converge to zero very quickly, and we therefore ignored them. A more theoretically rigorous approach would require the consideration of these terms, a bound of which can obtained using e.g. a projected Lanczos algorithm, as discussed in §3.5.2.

The linear systems of Algorithm 3.1 are solved to machine precision via an LDL factorization [2, §16.2]. We did not rely on an approximate method for the solution of the linear system because, in the problems that we considered, the projection to the Semidefinite Cone required the majority of the total time of Algorithm 3.1.

Nevertheless, the analysis of presented in Sections 3.2-3.4 allows for the presence of approximation errors in the solution of the linear systems.

3.6.1 Results for the SDPLIB collection

We first consider problems of the SDPLIB collection, in their dual form, i.e.

$$\begin{array}{ll} \text{maximize} & \langle F_0, Y \rangle \\ \text{subject to} & \langle F_i, Y \rangle = c_i \\ & Y \in \mathbb{S}^n_+. \end{array} \tag{3.26}$$

The problems are stored in the sparse SDPA form, which was designed to efficiently represent SDP problems in which the matrices $F_i, i = 0, \dots m$ are block diagonal with sparse blocks. If the matrices F_i consist of ℓ diagonal blocks, then the solution of (3.26) can be obtained by solving

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{l} \langle F_{0,j}, Y_j \rangle \\ \text{subject to} & \sum_{j=1}^{l} \langle F_{i,j}, Y_j \rangle = c_i \\ & Y_j \in \mathbb{S}^{n_j}_+ \quad j = 1, \dots, \ell. \end{array} \tag{3.27}$$

where $F_{i,j}$ denotes the $j-$th diagonal block of F_i and Y_j the respective block of Y. Note that (3.27) has more but smaller semidefinite variables than (3.26); thus it is typically solved by solvers like COSMO.jl more efficiently than (3.26). As a result, our results refer to the solution of problems in the form (3.27).

Table 3.1 shows the results on all the problems of SDPLIB problems for which the largest semidefinite variable is of size at least 50. We observe that using approximate projections in `COSMO.jl` can lead to a significant speedup of up to 20x. At the same time, the robustness of the solver is not affected, in the sense that the number of iterations to reach convergence is not, on average, increased by using approximate projections. It is remarkable that for every problem that the original `COSMO.jl` implementation converges within 2500 iterations (i.e. the default maximum iteration limit), our approach also converges with a faster overall solution time.

Table 3.1: Results of `COSMO.jl` on the `SDPLIB` collection when computing the semidefinite projections with "exact" eigendecompositions as compared to LOBPCG. The column $n_{\max}$ denotes the dimensions of the largest semidefinite variable at each problem while the column "rank" denotes the maximum number of computed ritz pairs by LOBPCG in the last iteration of Algorithm 3.3. The columns $t^{\text{exact}}, t^{\text{exact}}_{\text{proj}}, \text{Iter}^{\text{exact}}, f^{\text{exact}}$ denote the solution time (in seconds), the time spent in computing projections to the semidefinite cone, the ADMM iterations and the resulting objective when computing the projections exactly. Similarly, the columns Iter and f denote respectively identical metrics when computing projections approximately. The columns Speedup and $\text{Speedup}_{\text{proj}}$ denote the speedups achieved for the full ADMM algorithm and only the project part respectively when using LOBPCG. Finally, the column f^* denotes the optimal objective of each problem. Hardware used: Intel Gold 5120 with 192GB of memory.

Name	$n_{\max}$	rank	t^{exact}	Speedup	$t^{\text{exact}}_{\text{proj}}$	$\text{Speedup}_{\text{proj}}$	$\text{Iter}^{\text{exact}}$	Iter	f^{exact}	f	f^*
arch0	161	5	2.14×10^1	1.00	1.98×10^1	0.99	2500	2500	2.35×10^5	2.35×10^5	5.67×10^{-1}
arch2	161	4	2.24×10^1	0.83	2.08×10^1	0.82	2500	2500	8.49×10^4	7.88×10^4	6.72×10^{-1}
arch4	161	5	2.24×10^1	0.65	2.05×10^1	0.62	2500	2500	3.30×10^5	3.30×10^5	9.73×10^{-1}
arch8	161	6	2.23×10^1	0.64	2.07×10^1	0.62	2500	2500	1.49×10^6	1.49×10^6	7.06
control10	100	26	3.61×10^1	0.98	1.05×10^1	1.00	2500	2500	7.31×10^2	7.51×10^2	3.85×10^1
control11	110	17	5.23×10^1	0.93	1.25×10^1	0.96	2500	2500	9.51×10^2	8.28×10^2	3.20×10^1
control6	60	16	7.92	0.76	4.37	0.65	2500	2500	6.44×10^2	6.44×10^2	3.73×10^1
control7	70	15	1.10×10^1	0.72	5.60	0.59	2500	2500	6.79×10^2	6.79×10^2	2.06×10^1
control8	80	14	1.63×10^1	0.77	6.75	0.62	2500	2500	6.97×10^2	7.08×10^2	2.03×10^1
control9	90	14	2.35×10^1	0.74	8.07	0.57	2500	2500	7.00×10^2	8.48×10^2	1.47×10^1
equalG11	801	104	3.49×10^1	3.11	3.03×10^1	3.80	160	120	6.25×10^2	6.25×10^2	6.29×10^2
equalG51	1001	258	1.05×10^2	2.14	9.27×10^1	2.17	280	160	3.99×10^3	3.98×10^3	4.01×10^3
gpp100	100	8	1.27	3.15	1.17	4.08	360	440	-4.49×10^1	-4.49×10^1	-4.49×10^1
gpp124-1	124	8	3.00	4.56	2.76	6.18	520	520	-7.28	-7.28	-7.34
gpp124-2	124	7	1.72	2.86	1.57	3.34	320	320	-4.68×10^1	-4.68×10^1	-4.69×10^1
gpp124-3	124	9	1.14	3.04	9.36×10^{-1}	3.74	200	280	-1.53×10^2	-1.53×10^2	-1.53×10^2
gpp124-4	124	9	2.24	7.23	2.09	10.44	360	240	-4.19×10^2	-4.18×10^2	-4.19×10^2
gpp250-1	250	10	1.37×10^1	5.03	1.23×10^1	8.22	720	720	-1.54×10^1	-1.54×10^1	-1.54×10^1
gpp250-2	250	12	1.88×10^1	6.69	1.72×10^1	13.21	840	800	-8.19×10^1	-8.18×10^1	-8.19×10^1

Table 3.1: Continued.

Name	$n_{\max}$	rank	t^{exact}	Speedup	$t^{\text{exact}}_{\text{proj}}$	$\text{Speedup}_{\text{proj}}$	$\text{Iter}^{\text{exact}}$	Iter	f^{exact}	f	f^*
gpp250-3	250	13	1.96×10^{1}	8.90	1.80×10^{1}	15.11	840	600	-3.03×10^{2}	-3.03×10^{2}	-3.04×10^{2}
gpp250-4	250	13	8.03	6.35	7.37	11.64	360	360	-7.47×10^{2}	-7.47×10^{2}	-7.47×10^{2}
gpp500-1	500	20	4.67×10^{1}	3.48	4.24×10^{1}	4.92	560	680	-2.61×10^{1}	-2.60×10^{1}	-2.53×10^{1}
gpp500-2	500	19	5.46×10^{1}	2.70	4.96×10^{1}	5.66	640	1720	-1.56×10^{2}	-1.56×10^{2}	-1.56×10^{2}
gpp500-3	500	21	8.17×10^{1}	14.45	7.44×10^{1}	29.97	960	440	-5.13×10^{2}	-5.13×10^{2}	-5.13×10^{2}
gpp500-4	500	22	2.71×10^{1}	5.50	2.42×10^{1}	10.77	320	360	-1.57×10^{3}	-1.57×10^{3}	-1.57×10^{3}
maxG11	800	43	1.31×10^{2}	6.94	1.18×10^{2}	13.21	640	600	6.29×10^{2}	6.29×10^{2}	6.29×10^{2}
maxG32	2000	70	1.23×10^{3}	9.01	1.11×10^{3}	22.47	760	760	1.57×10^{3}	1.57×10^{3}	1.57×10^{3}
maxG51	1000	54	7.22×10^{1}	3.17	6.43×10^{1}	5.25	200	360	4.00×10^{3}	4.00×10^{3}	4.00×10^{3}
maxG55	5000	105	1.27×10^{4}	8.97	1.15×10^{4}	17.65	840	960	1.28×10^{4}	1.28×10^{4}	10.00×10^{3}
maxG60	7000	178	4.67×10^{4}	10.75	4.31×10^{4}	15.71	880	920	1.52×10^{4}	1.51×10^{4}	1.52×10^{4}
mcp100	100	8	1.37	3.25	1.25	3.98	280	320	2.26×10^{2}	2.26×10^{2}	2.26×10^{2}
mcp124-1	124	20	2.88	2.29	2.63	2.37	400	400	1.42×10^{2}	1.42×10^{2}	1.42×10^{2}
mcp124-2	124	8	1.51	3.13	1.38	4.26	240	240	2.70×10^{2}	2.70×10^{2}	2.70×10^{2}
mcp124-3	124	9	1.49	3.21	1.35	4.06	240	400	4.68×10^{2}	4.68×10^{2}	4.68×10^{2}
mcp124-4	124	8	1.60	4.62	1.47	5.93	200	240	8.64×10^{2}	8.64×10^{2}	8.64×10^{2}
mcp250-1	250	30	1.68×10^{1}	2.69	1.52×10^{1}	2.88	680	680	3.17×10^{2}	3.17×10^{2}	3.17×10^{2}
mcp250-2	250	14	1.17×10^{1}	6.73	1.05×10^{1}	10.60	560	520	5.32×10^{2}	5.32×10^{2}	5.32×10^{2}
mcp250-3	250	13	9.16	5.14	8.20	6.99	440	400	9.81×10^{2}	9.81×10^{2}	9.81×10^{2}
mcp250-4	250	14	4.39	2.90	3.93	4.43	200	360	1.68×10^{3}	1.68×10^{3}	1.68×10^{3}
mcp500-1	500	72	9.89×10^{1}	1.26	9.11×10^{1}	1.50	920	2400	5.98×10^{2}	5.98×10^{2}	5.98×10^{2}
mcp500-2	500	27	1.20×10^{2}	12.40	1.09×10^{2}	22.08	1160	600	1.07×10^{3}	1.07×10^{3}	1.07×10^{3}
mcp500-3	500	21	4.78×10^{1}	4.10	4.25×10^{1}	7.26	400	760	1.85×10^{3}	1.85×10^{3}	1.85×10^{3}
mcp500-4	500	22	7.31×10^{1}	9.82	6.67×10^{1}	17.94	680	480	3.57×10^{3}	3.57×10^{3}	3.57×10^{3}
qap10	101	29	1.21	2.07	1.07	2.09	280	160	-1.08×10^{3}	-1.08×10^{3}	-1.09×10^{1}
qap8	65	20	3.70×10^{-1}	1.13	3.04×10^{-1}	1.06	160	160	-7.42×10^{2}	-7.42×10^{2}	-7.57×10^{2}
qap9	82	24	4.84×10^{-1}	1.16	4.10×10^{-1}	1.10	160	160	-1.39×10^{3}	-1.39×10^{3}	-1.41×10^{3}

Table 3.1: Continued.

Name	$n_{\max}$	rank	t^{exact}	Speedup	$t^{\text{exact}}_{\text{proj}}$	$\text{Speedup}_{\text{proj}}$	$\text{Iter}^{\text{exact}}$	Iter	f^{exact}	f	f^*
qpG11	1600	38	7.10×10^2	5.71	6.11×10^2	13.14	920	880	2.45×10^3	2.45×10^3	2.45×10^3
qpG51	2000	20	3.64×10^3	10.13	3.20×10^3	38.71	2500	2500	1.18×10^4	1.19×10^4	1.18×10^3
ss30	294	8	9.70×10^1	0.95	8.87×10^1	0.92	2500	2500	1.94×10^6	1.94×10^6	2.02×10^1
theta2	100	18	1.04×10^1	2.67	9.56	2.82	2240	2240	3.29×10^1	3.29×10^1	3.29×10^1
theta3	150	26	5.05	3.12	4.60	3.85	440	560	4.23×10^1	4.22×10^1	4.22×10^1
theta4	200	34	1.13×10^1	3.99	1.01×10^1	4.58	720	680	5.05×10^1	5.04×10^1	5.03×10^1
theta5	250	17	1.16×10^1	0.88	1.03×10^1	1.09	480	2500	5.74×10^1	9.43×10^1	5.72×10^1
theta6	300	13	1.68×10^1	1.02	1.46×10^1	1.33	480	2500	6.36×10^1	-2.39	6.35×10^1
thetaG11	801	26	7.71×10^2	20.79	7.37×10^2	47.91	1280	1280	3.99×10^2	3.99×10^2	4.00×10^2
thetaG51	1001	116	1.31×10^3	5.40	1.19×10^3	7.00	2500	2500	3.40×10^2	3.43×10^2	3.49×10^2

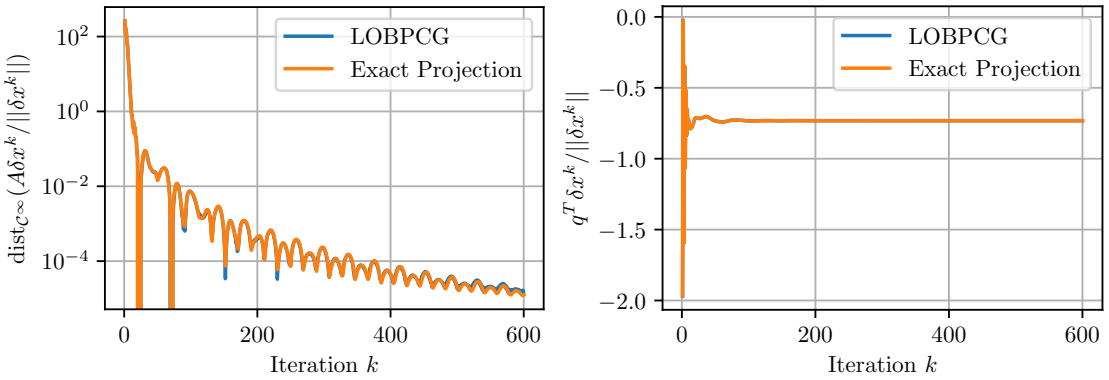


Figure 3.1: Convergence of $\left(\delta x^k / \left\|\delta x^k\right\|\right)_{k\in\mathbb{N}}$ to a certificate of dual infeasibility for problem `infd1` from the SDPLIB. A fixed value of $\rho = 10^{-3}$ is used in Algorithm 3.1.

3.6.2 Infeasible Problems

Next, we demonstrate the asymptotic behavior of Algorithm 3.1 on the problem `infd1` of the SDPLIB collection. This problem can be expressed in the form (P) with $\mathcal{C} = \{\text{vec}_u(X) \mid X \in \mathbb{S}^{30}\}$ (the set of vectorized 30×30 positive semidefinite matrices), and $x \in \mathbb{R}^{10}$.

As the name suggests, `infd1` is dual infeasible. Following [8, §5.2], `COSMO` detects dual infeasibility in conic problems when the certificate (3.3) holds approximately, that is when $\delta x^k \neq 0$ and

$$\text{dist}_{C^\infty}\left(A\bar{x}^k\right) < \epsilon_{\text{dinf}}, \quad \text{and} \quad q^T\bar{x}^k < \epsilon_{\text{dinf}},$$

where $\bar{x}^k := \delta x^k/||\delta x^k||$, for a positive tolerance ϵ_{dinf}. Figure 3.1, depicts the convergence of these quantities both for the case where the projection to the semidefinite cone are computed approximately and when LOBPCG is used. The convergence of the successive differences to a certificate of dual infeasibility is practically identical.

To demonstrate the detection of primal infeasibility we consider the dual of `infd1`. Following [8, §5.2], `COSMO` detects primal infeasibility in conic problems when the certificate (3.2) is satisfied approximately, that is when $\delta y^k \neq 0$ and

$$\left\|P\bar{y}^k\right\| < \epsilon_{\text{pinf}}, \quad \left\|A^T\bar{y}^k\right\| < \epsilon_{\text{pinf}}, \quad \text{dist}_{C^\circ}\left(\bar{y}^k\right) < \epsilon_{\text{pinf}}, \quad b^T\bar{y}^k < \epsilon_{\text{pinf}},$$

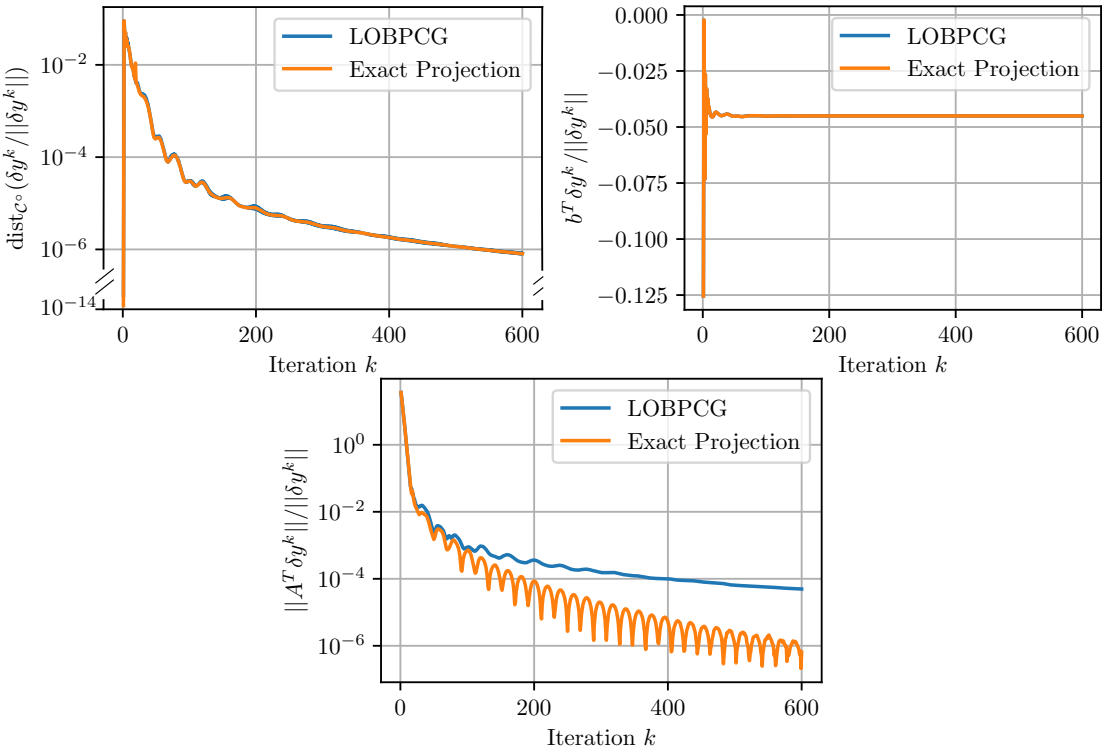


Figure 3.2: Convergence of $\left(\delta y^k/\left\|\delta y^k\right\|\right)_{k\in\mathbb{N}}$ to a certificate of primal infeasibility for the dual of the problem `infd1` from the SDPLIB. A fixed value of $\rho = 10^3$ is used in Algorithm 3.1.

where $\bar{y}^k := \delta y^k/||\delta y^k||$, for a positive tolerance ϵ_{pinf}. Note that, for the case of the dual of `infd1`, the first condition is trivial since $P = 0$. Figure 3.2 compares the convergence of our approach, against standard COSMO, to a certificate of infeasibility. LOBPCG yields practically identical convergence as the exact projection for all of the quantities except $\left\|A^T\bar{y}^k\right\|$, where slower convergence is observed.

Note that SDPLIB also contains two instances of primal infeasible problems: `infp1` and `infp2`. However, in these problems, there is a single positive semidefinite variable of size 30 and, in ADMM, the matrices projected to the semidefinite cone have rank $15 = 30/2$ across all the iterations (except for the very first few). Thus, according to §3.5.1 LOBPCG yields identical results to the exact projection, hence a comparison would be of little value.

3.6.3 Distributional Ambiguity in Bayesian Optimization

We next consider the solution of thousands of semidefinite problems generated by the Bayesian Optimization Application of Chapter 2. Solving these problems entails the projection of millions of matrices to the semidefinite cone, which will serve as a benchmark to demonstrate the robustness of our approach. Recall that performing Bayesian Optimization using the Distributionally Ambiguous Framework of Chapter 2 entails the solution of Semidefinite Optimization problems of the following form:

$$\begin{aligned} \inf_{Y_0,\ldots,Y_\ell} \quad & \sum_{i=1}^{\ell} \langle Y_i, C_i^k \rangle \\ \text{subject to} \quad & Y_i \succeq 0, \qquad i = 0, \ldots, \ell \\ & \sum_{i=0}^{k} Y_i = \Omega^k(X), \end{aligned} \tag{3.28}$$

where C_i are defined as

$$C_i^k := \begin{bmatrix} 0 & e_i/2 \\ e_i^T/2 & -y_{\min}^k \end{bmatrix}, \qquad i = 1, \ldots, \ell, \tag{3.29}$$

using e_i, the standard basis vectors in $\mathbb{R}^\ell$, and $y_{\min}^k$, the minimum outcome of h observed so far. (3.28) is solved at every point where the nonlinear solver employed for the acquisition function $\alpha^k(\Omega^k(X))$ requests the objective or the gradient of $\alpha^k(\Omega^k(X))$.

Solving (3.28) with the approach suggested in this Chapter is particularly attractive for two reasons. First, according to Lemma 2.A.2, the optimal matrices $\{\bar{Y}_i\}$ of (3.28) are guaranteed to be of rank one, thus LOBPCG will be particularly efficient in computing the projections to the semidefinite cone. Second, as we show in Appendix 3.C, the linear systems embedded in ADMM are trivial to solve, thus the solution time is expected to be heavily dominated by the projections.

In Table 3.2 we compare the average solution time of problems (3.28) generated by running Bayesian Optimization in a standard test function. Our approach exhibits a significant, up to 10x, speedups and no considerable increase in the average number of ADMM's iterations. Table 3.3 provides a detailed breakdown of ADMM's solution time.

Table 3.2: Results of solving (3.28) for problems generated by the first iteration of Bayesian Optimization in the Eggholder function using the same setup as Chapter 2. Batch size (ℓ) denotes the number of columns in X, $\text{iter}^{\text{exact}}$ ($\text{iter}^{\text{exact}}$) denote average number of iterations reported for COSMO when exact projections (LOBPCG) are used, and t^{exact} (t) denotes the average solution time of original (LOBPCG variant of) COSMO. Hardware used: a single thread of Intel Gold 5120 with 192GB of memory.

Batch size	$\text{iter}^{\text{exact}}$	iter	t^{exact}	t	Speedup
20	118.6	114.5	4.31×10^{-1}	2.96×10^{-1}	1.45
30	120.4	130.0	1.28	5.40×10^{-1}	2.37
40	141.4	144.3	3.23	8.60×10^{-1}	3.76
50	155.3	151.7	6.47	1.31	4.94
100	152.8	167.8	4.50×10^{1}	6.16	7.31
200	134.7	145.7	3.50×10^{2}	3.36×10^{1}	10.43

Table 3.3: Detailed timings for the experiment of Table 3.2. $t_{\text{projection}}$ ($t_{\text{linear system}}$) denotes the average time spent on line 5 (line 3) of Algorihm 3.1.

Batch size	$t^{\text{exact}}_{\text{projection}}$	$t_{\text{projection}}$	$t^{\text{exact}}_{\text{linear system}}$	$t_{\text{linear system}}$
20	4.1×10^{-1}	2.8×10^{-1}	1.1×10^{-2}	2.8×10^{-3}
30	1.2	5.0×10^{-1}	3.4×10^{-2}	8.8×10^{-3}
40	3.0	7.5×10^{-1}	1.1×10^{-1}	2.4×10^{-2}
50	6.1	1.1	2.1×10^{-1}	5.5×10^{-2}
100	4.0×10^{1}	3.4	3.0	6.7×10^{-1}
200	3.1×10^{2}	1.4×10^{1}	2.5×10^{1}	4.7

3.7 Conclusions

We have shown that state-of-the art approximate eigensolvers can bring significant speedups to ADMM for the case of Semidefinite Programming. We have extended the results of [8] to show that infeasibility can be detected even in the presence of appropriately controlled projection errors, thus ensuring the same overall asymptotic behavior as an exact ADMM method. Future research directions include exploring the performance of other state-of-the-art eigensolvers from the Linear Algebra community [59].

3.A Expressing ADMM in operator form

In this section, we show how ADMM can be recast as an iteration of an averaged nonexpansive operator. Recall that ADMM amounts to the following iteration

$$\chi^{k+1} = \text{prox}_f(\psi^k - \omega^k) \tag{3.30}$$

$$\psi^{k+1} = \text{prox}_g(\bar{\chi}^{k+1} + \omega^k) \tag{3.31}$$

$$\omega^{k+1} = \omega^k + (\bar{\chi}^{k+1} - \psi^{k+1}) \tag{3.32}$$

where $\bar{\chi}^{k+1} = \alpha\chi^{k+1} + (1-\alpha)\psi^k$. Following [71, Appendix A], we will show that ADMM performs the following iteration on the variable $\phi^k := \bar{\chi}^k + \omega^{k-1}$:

$$(\forall k \in \mathbb{N}) \quad \phi^{k+1} = T\phi^k = \phi^k - \alpha(\text{prox}_f R_g \phi^k - \text{prox}_g \phi^k), \tag{3.33}$$

where T is defined according to (3.6) and $R_g = 2\text{prox}_g - Id$. To begin, note that due to (3.31) we have for every $k \in \mathbb{N}$

$$\psi^k = \text{prox}_g(\bar{\chi}^k + \omega^{k-1}) = \text{prox}_g \phi^k. \tag{3.34}$$

Furthermore, recalling the definition of ϕ^k and (3.32) we get

$$\phi^{k+1} - \phi^k = \bar{\chi}^{k+1} - \bar{\chi}^k + \omega^{k+1} - \omega^k = \bar{\chi}^{k+1} - \psi^k = \alpha(\chi^k - \psi^k), \tag{3.35}$$

and

$$\psi^k - \omega^k = 2\psi^k - \bar{\chi}^k - \omega^{k-1} = 2\psi^k - \phi^k, \tag{3.36}$$

which gives

$$\chi^{k+1} = \text{prox}_f(\psi^k - \omega^k) = \text{prox}_f(2\psi^k - \phi^k). \tag{3.37}$$

In total, we have:

$$\psi^k = \text{prox}_g \phi^k \tag{3.38}$$

$$\chi^{k+1} = \text{prox}_f(2\psi^k - \phi^k) \tag{3.39}$$

$$\phi^{k+1} = \phi^k + \alpha(\chi^{k+1} - \psi^k), \tag{3.40}$$

from which (3.33) follows easily. Finally, using the definitions of ϕ^k, $\bar{\chi}^{k+1}$ and (3.40), we can also get an expression for ω^k w.r.t to ϕ^k:

$$\begin{aligned}\omega^k &= \phi^{k+1} - \bar{\chi}^{k+1} = \phi^{k+1} - \alpha\chi^{k+1} + (1-\alpha)\psi^k \\ &= \phi^k - \psi^k = (Id - \operatorname{prox}_g)\phi^k.\end{aligned}$$

3.B A convergence result for approximate iterations of nonexpansive operators

In this section we provide a proof for Theorem 3.3.1. We achieve this by generalizing some of the results of [72] and [73] to account for sequences generated by approximate evaluation of a nonexpansive operators $T : \mathcal{D} \to \mathcal{D}$ for which $\operatorname{cl}\mathcal{R}(Id - T)$ has the *minimum property* defined below:

Definition(Minimum Property)**.** Let $K \subseteq \mathcal{H}$ be closed and let ℓ be the minimal element of $\operatorname{cl}\operatorname{conv} K$. The set K has the minimum property if $\ell \in K$.

Note that $\operatorname{cl}\mathcal{R}(Id - T)$ has the minimum property when T is defined as (3.6) because the domain of (3.6) is convex [72, Lemma 5]. Thus Theorem 3.3.1 follows from the following result:

Proposition 3.B.1. *Consider some $\mathcal{D} \subseteq \mathcal{H}$ that is closed, a nonexpansive $T : \mathcal{D} \to \mathcal{D}$ and assume that $\operatorname{cl}\mathcal{R}(Id - T)$ has the minimum property. For any sequence that satisfies*

$$(\forall k \in \mathbb{N}) \quad x^{k+1} \approx_{\epsilon^k} Tx^k,$$

for some $x^0 \in \mathcal{D}$ and a summable nonnegative sequence $(\epsilon^k)_{k\in\mathbb{N}}$, we have

$$\lim_{k\to\infty}(x^{k+1} - x^k) = \lim_{k\to\infty} x^k/k = -\ell,$$

where ℓ is the unique element of minimum norm in $\operatorname{cl}\mathcal{R}(Id - T)$.

Proof. We will first show that $\lim_{k\to\infty} x^k/k = -\ell$. The nonexpansiveness of T gives

$$\begin{aligned}\left\|x^n - T^n x^0\right\| &\leq \left\|Tx^{n-1} - TT^{n-1}x^0\right\| + \epsilon_n, \quad \forall n \in \mathbb{N} \\ \Rightarrow \frac{1}{n}\left\|x^n - T^n x^0\right\| &\leq \frac{1}{n}\sum_{i=1}^{n}\epsilon_i \Rightarrow \lim_{n\to\infty}\left\|\frac{x^n}{n} - \frac{T^n x^0}{n}\right\| = 0,\end{aligned}$$

where the summability of $(\epsilon_i)_{i\in\mathbb{N}}$ was used in the last implication. Thus, the claim follows from [72, Theorem 2].

It remains to show that $\lim_{k\to\infty} x^{k+1} - x^k$ also converges to $-\ell$. We will begin by showing that $\lim_{k\to\infty}\left\|x^{k+1} - x^k\right\|$ exists and is bounded. To this end, define $\delta^k := Tx^{k-1} - x^k$ and consider the sequence

$$s^k := \left\|x^{k+1} - x^k\right\| + \sum_{i=k}^{\infty}\left\|\delta^{i+1} - \delta^i\right\|.$$

Note that $\sum_{i=k}^{\infty}\|\delta^{i+1} - \delta^i\|$ converges to a finite value for every k, as it is the limit $n \to \infty$ of the nondecreasing sequence $(\sum_{i=k}^{n}\|\delta^{i+1} - \delta^i\|)$ that is bounded above because $(\|\delta^i\|)$ is summable. Since $\left\|x^k - x^{k-1}\right\| \leq \left\|x^{k+1} - x^k\right\| + \left\|\delta^{k+1} - \delta^k\right\|$ we conclude that (s^k) is nonincreasing. Since (s^k) is also bounded below by zero we conclude that $\lim_{k\to\infty} s^k$ exists and is bounded. Finally, because $\lim_{k\to\infty}\sum_{i=k}^{\infty}\|\delta^{i+1} - \delta^i\| = 0$, we conclude that $\lim_{k\to\infty}\left\|x^{k+1} - x^k\right\|$ also exists and is bounded.

We will now show that $\lim_{k\to\infty}\left\|x^{k+1} - x^k\right\| = \|\ell\|$. To show this, note that

$$x^{k+1} - x^k = Tx^k - x^k - \delta^{k+1}, \tag{3.41}$$

and, following [73, Theorem 2.1], we have

$$\begin{aligned}\lim_{k\to\infty}\left\|x^{k+1} - x^k\right\| &\leq \lim_{k\to\infty}\left\|Tx^k - x^k\right\| + \lim_{k\to\infty}\left\|\delta^{k+1}\right\| \\ &\leq \|\ell\| = \lim_{k\to\infty}\frac{1}{k}\left\|x^k - x^0\right\| \\ &\leq \lim_{k\to\infty}\frac{1}{k}\sum_{i=1}^{k}\left\|x^{i+1} - x^i\right\| \\ &= \lim_{k\to\infty}\left\|x^{k+1} - x^k\right\|,\end{aligned}$$

where the definition of the minimum property was used in the second line of the above equation, and the properties of the Cesàro summation were used in the last.

Hence, $\lim_{k\to\infty}\left\|x^{k+1} - x^k\right\| = \|\ell\| \overset{(3.41)}{\Longrightarrow} \lim_{k\to\infty}\left\|Tx^k - x^k\right\| = \|\ell\|$ resulting in $\lim_{k\to\infty} Tx^k - x^k = -\ell$ due to [72, Lemma 2]. The desired $\lim_{k\to\infty} x^{k+1} - x^k = -\ell$ follows then from (3.41). □

3.C Explicit formula for prox_f in problem (3.28)

In this section, we show that prox_f, i.e. the linear system of Algorithm 3.1 (line 3) for (3.28) admits a trivial solution. To show this, note that for (3.28) we have $P = 0$ and $A^T = \begin{bmatrix}\mathbf{1}_m \otimes I_{m^2} & I_{m^3}\end{bmatrix}$, so the linear systems embedded in Algorithm 3.1 are of the form[8]

$$\begin{bmatrix} \sigma I_{m^3} & \rho_1 \mathbf{1}_m \otimes I_{m^2} & \rho_2 I_{m^3} \\ \rho_1 \mathbf{1}_m^T \otimes I_{m^2} & -\rho_1 I_{m^2} & \\ \rho_2 I_{m^3} & & -\rho_2 I_{m^3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}, \tag{3.42}$$

where $m := \ell + 1$, for some $x := [x_1;\ x_2;\ x_3]$, $y := [y_1;\ y_2;\ y_3]$. From the first block row of (3.42) we get

$$x_3 = x_1 - y_3/\rho_2. \tag{3.43}$$

Thus, we can reduce x_3 from (3.42) to get:

$$\begin{bmatrix} (\sigma + \rho_2) I_{m^3} & \rho_1 A_1^T \\ \rho_1 A_1 & -\rho_1 I_{m^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 + y_3 \\ y_2 \end{bmatrix}, \tag{3.44}$$

where

$$A_1^T := \mathbf{1}_m \otimes I_{m^2}. \tag{3.45}$$

Using the first block row of (3.44), we have

$$x_1 = \left(-\rho_1 A_1^T x_2 + y_1 + y_3\right)/(\sigma + \rho_2). \tag{3.46}$$

Substituting the above in the second block row (3.44), we get

$$\left(\frac{\rho_1^2}{\sigma + \rho_2} A_1 A_1^T + \rho_1 I\right) x_2 = \frac{\rho_1}{\sigma + \rho_2} A_1 (y_1 + y_3) - y_2$$

or, since $A_1 A_1^T = (\mathbf{1}_m^T \otimes I_{m^2})(\mathbf{1}_m \otimes I_{m^2}) = m I_{m^2}$ [1, (1.3.1)-(1.3.2)], we get

$$x_2 = \left(\frac{\rho_1^2}{\sigma + \rho_2} m + \rho_1\right)^{-1} \left(\frac{\rho_1}{\sigma + \rho_2} A_1 (y_1 + y_3) - y_2\right) \tag{3.47}$$

Equation (3.47), (3.46) and (3.43) give the solution of (3.42) in terms of operations of vector scaling, vector additions, and multiplication with A_1 and its transpose. These operations scale linearly with the size of y and are highly parallelizable.

[8] Note that we allow for two different values of penalty parameters ρ, following [53, §5.2].

4

Accuracy of approximate projection to the semidefinite cone

4.1 Introduction

In this chapter we prove the key theoretical result that allowed for the approximate ADMM algorithm of the previous chapter to have convergence guarantees in semidefinite programming. In particular, we show that the accuracy of approximate projections onto the semidefinite cone is independent of spectral gaps. The results of this chapter are also more widely useful, as the projection of symmetric or Hermitian matrices to the positive semidefinite cone is a standard operation that arises frequently in scientific computing. A common, practical, example is restoring positive definiteness of partially unknown or corrupted correlation matrices [74] arising in e.g., economics [75], integrated circuit design [76] and wireless communications [77]. Further, more generic, examples include quasi-newton optimization methods [33, §4.2.2] and incomplete matrix factorizations of sparse matrices [78, §15.11].

The projection operator Π maps a Hermitian matrix to a nearest positive semidefinite matrix. As such, it belongs to the class of *matrix nearness problems*, a

survey of which can be found in [79]. When a unitarily invariant norm[1] is chosen as a distance metric, the projection operator can be expressed in "closed form". Indeed, we show in §4.2.2 that if $A \in \mathbb{H}^n$ with eigenvalue decomposition

$$A = \begin{bmatrix} V_+ & V_- \end{bmatrix} \begin{bmatrix} \Lambda_+ & \\ & \Lambda_- \end{bmatrix} \begin{bmatrix} V_+ & V_- \end{bmatrix}^T, \tag{4.1}$$

where $[V_+ \; V_-]$ is an $n \times n$ orthogonal matrix with V_+ having k columns and Λ_+, Λ_- diagonal matrices containing the positive and non-positive eigenvalues respectively, then we can set:

$$\Pi(A) := V_+ \Lambda_+ V_+^T. \tag{4.2}$$

When $\Pi(A)$ is computed in practice, either via the full eigenvalue decomposition or an iterative method such as the (block) Lanczos method, one obtains approximations $\widehat{\Lambda}$ ($\tilde{k} \times \tilde{k}$ diagonal), and $\widehat{V}$ ($n \times \tilde{k}$ orthonormal) to Λ_+, V_+ such that $\widehat{V}\widehat{\Lambda}\widehat{V}^T \approx V_+ \Lambda_+ V_+^T$. We usually (but not necessarily) take $\widehat{\Lambda} = \widehat{V}^T A \widehat{V}$, which holds by the standard Rayleigh-Ritz process [56, Ch. 11]. Note that the sizes k and $\tilde{k}$ are not assumed to agree since the number of positive eigenvalues computed may not be exact, particularly in the presence of eigenvalues close to zero.

With a backward stable algorithm [1, §3.4.10], we have $\|A\widehat{V} - \widehat{V}\widehat{\Lambda}\| = O(\epsilon_u)\|A\|$, where ϵ_u is the unit roundoff. The question we address is: what does this imply about the size of the projection error $\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\|$?

The first observation can arise if we assume that $k = \tilde{k}$ and define $\widehat{V} = V_+ + \Delta V$, $\widehat{\Lambda} = \Lambda_+ + \Delta\Lambda$, resulting in

$$\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\| \leq 2\|\Delta V \Lambda_+ V_+^T\| + \|V_+\Delta\Lambda V_+^T\| + O(\|\Delta\|^2) \tag{4.3}$$

where $\|\Delta\| := \max(\|\Delta V\|, \|\Delta\Lambda\|)$. This suggests that bounding the term $\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\|$ might be achieved by bounding the term $\|\Delta V\|$, which usually dominates the $\|\Delta\Lambda\|$ term since eigenvalues of Hermitian matrices are always well-conditioned, hence $\|\Delta\Lambda\| = O(\epsilon_u)\|A\|$ with a backward stable method. The term ΔV is the

[1] I.e. a matrix norm that is unchanged when the matrix is multiplied on the right or left by a unitary matrix.

error in the computed eigenvectors, for which the classical Davis-Kahan $\sin\theta$ theorem [80] shows that

$$\|\sin\angle(V_+, \widehat{V})\| \leq \frac{\|R\|}{\text{gap}} \tag{4.4}$$

in any unitarily invariant norm, where $\angle(V_+, \widehat{V})$ is a $k \times k$ diagonal matrix containing the principal angles between V_+ and $\widehat{V}$. Here $R := A\widehat{V} - \widehat{V}\widehat{\Lambda}$ is the residual and 'gap' is the distance between the computed positive eigenvalues $\widehat{\Lambda}$ and the exact nonpositive eigenvalues, i.e.

$$\text{gap} := \min_{i,j} \left|\widehat{\Lambda}_{(i,i)} - \Lambda_{-(j,j)}\right|. \tag{4.5}$$

The bound (4.4) is essentially sharp (although sometimes improvable [70]), and implies[2] $\|\Delta V\| \lesssim \frac{\|R\|}{\text{gap}}$. Together with (4.3), we obtain

$$\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\| \lesssim \frac{2\|R\|\|\Lambda_+V_+^T\|}{\text{gap}} \leq \frac{2\|R\|\|A\|}{\text{gap}}. \tag{4.6}$$

The problem here is that (4.3) suggests that the projection error would be *large* if gap is *small*, and we have no control over how small gap can be! Indeed, in many interesting cases A has eigenvalues of small magnitude (both positive and negative). This causes problems in multiple ways–the gap indeed gets small, and since approximations to the small eigenvalues may have the wrong signs, $\widehat{\Lambda}$ may not contain the correct number of positive eigenvalues. The situation is further exacerbated when $\widehat{\Lambda}$, $\widehat{V}$ are computed not to full working precision, but only to a looser tolerance $\epsilon \gg \epsilon_u$. Then, $\|R\| = O(\epsilon\|A\|)$ instead of $O(\epsilon_u\|A\|)$.

The purpose of this chapter is to show that, fortunately, the problems suggested by (4.6) are not a concern, i.e., clustered eigenvalues do *not* affect the projection accuracy. Specifically, our main result is

$$\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\|_F \leq \sqrt{2}\|R\|_F, \tag{4.7}$$

[2] $\|\Delta V\|$ also depends on the "small gap" $:= \min_{i\neq j}\left|\widehat{\Lambda}_{(i,i)} - \Lambda_{(j,j)}\right|$, where $\Lambda := \left[\begin{smallmatrix}\Lambda_+ & \\ & \Lambda_-\end{smallmatrix}\right]$, [80] but this is not a practical concern when $\|R\| \leq$ "small gap". See [70, Theorem 3.1 and Remark 3.1] for a rigorous discussion on the effect of spectral gaps on $\|\Delta V\|$.

which holds when $\widehat{\Lambda}$ has the same size as Λ_+ (otherwise the bound worsens, but only slightly). Unlike (4.6), the bound (4.7) is sharp up to a constant smaller than $\sqrt{2}$. Since $\|R\|_F = \|A\widehat{V} - \widehat{V}\widehat{\Lambda}\|_F$ is easy to compute, (4.7) provides a practical means to estimate the projection accuracy.

Noting that $\Pi(A) = V_+\Lambda_+V_+^T = A(V_+V_+^T)$, we also treat an alternative measure of the projection accuracy

$$\|A(V_+V_+^T - \widehat{V}\widehat{V}^T)\|, \tag{4.8}$$

and prove similar bounds for this quantity. The two quantities $\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\|$ and (4.8) are closely related, and we show in Section 4.3 that they must lie within $\|R\|$ of each other.

Here is an intuitive explanation for the gap-independence (which is easier to see with (4.8)): while $\|\Delta V\|$ does depend on 1/gap, large errors in ΔV lie only in directions of eigenvectors v_i of A with small corresponding eigenvalues λ_i. Essentially, ΔV has $O(\|R\|/|\lambda_i|)$ magnitude in the direction of v_i. Crucially, such errors are suppressed when multiplied by A as in (4.8), precisely by λ_i. Thus they cancel out to yield the quantity $\|R\|$ in our bounds for both $\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\|_F$ and $\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\|$ in (4.7) and (4.8), respectively. In what follows we make this intuition precise.

4.2 Main result for the projection error

In this section we derive the main result for the projection error $\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\|_F$. We begin with a presentation of the main idea used in this Section's proofs. Assume, for the purposes of this introductory presentation, that we have $\widehat{\Lambda} = \widehat{V}^TA\widehat{V}$, as is common in practice, and that $\widehat{\Lambda} \succeq 0$. Then, note that

$$\begin{bmatrix}\widehat{V} & \widehat{V}_\perp\end{bmatrix}^T A\begin{bmatrix}\widehat{V} & \widehat{V}_\perp\end{bmatrix} = \begin{bmatrix}\widehat{\Lambda} & \widehat{R}^T \\ \widehat{R} & D\end{bmatrix} \tag{4.9}$$

where $D := \widehat{V}_\perp^TA\widehat{V}_\perp$ and $\|\widehat{R}\| = \|R\|$ for any unitarily invariant norm (derived in detail in Theorem 4.2.1:(4.14)). Thus $\widehat{A} := \begin{bmatrix}\widehat{V} & \widehat{V}_\perp\end{bmatrix}\begin{bmatrix}\widehat{\Lambda} & 0 \\ 0 & D\end{bmatrix}\begin{bmatrix}\widehat{V} & \widehat{V}_\perp\end{bmatrix}^T$ approximates A with $\|A - \widehat{A}\| \leq \sqrt{2}\|R\|$. Now, if we assume that $\widehat{V}_\perp$ contains 'most'

of the negative eigenspace of A, in that $D \preceq 0$, then we have $\Pi(\widehat{A}) = \widehat{V}\widehat{\Lambda}\widehat{V}^T$. We can then write

$$\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\| = \|\Pi(\widehat{A}) - \Pi(A)\|. \tag{4.10}$$

The result (4.7) then follows immediately from the non-expansiveness of Π in the Frobenius norm [81, Proposition 4.16] [82, Theorem 5.1], i.e., $\|\Pi(\widehat{A}) - \Pi(A)\|_F \leq \|\widehat{A} - A\|_F = \sqrt{2}\|R\|_F$.

The following proofs generalize this result for the cases where $\widehat{\Lambda} \neq \widehat{V}^T A\widehat{V}$ and/or $D \not\preceq 0$.

Theorem 4.2.1. *Suppose that the matrix $A \in \mathbb{S}^n$, $\widehat{V}$ is an $n \times k$ orthonormal matrix with $k \leq n$ and $\widehat{\Lambda}$ is a $k \times k$ positive semidefinite matrix. Then writing $R = A\widehat{V} - \widehat{V}\widehat{\Lambda}$ and $D_+ = \Pi(\widehat{V}_\perp^T A\widehat{V}_\perp)$,*

$$\left\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - \Pi(A)\right\|_F^2 \leq \|R\|_F^2 + \left\|\widehat{V}_\perp^T A\widehat{V}\right\|_F^2 + \|D_+\|_F^2. \tag{4.11}$$

Proof. Define

$$B := \widehat{V}\widehat{\Lambda}\widehat{V}^T + \widehat{V}_\perp \Pi_{\mathbb{H}_-}(\widehat{V}_\perp^T A\widehat{V}_\perp)\widehat{V}_\perp^T.$$

Then

$$\begin{aligned}\left\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - \Pi(A)\right\|_F^2 &= \|\Pi(B) - \Pi(A)\|_F^2 \\ &\leq \|B - A\|_F^2,\end{aligned} \tag{4.12}$$

where the inequality in the second line follows from the nonexpansiveness of the projection operator in the Frobenius norm. Since this norm is invariant with respect to unitary transformation, we have

$$\begin{aligned}\|B - A\|_F^2 &= \left\|\begin{bmatrix}\widehat{V}^T \\ \widehat{V}_\perp^T\end{bmatrix}(B - A)\begin{bmatrix}\widehat{V} & \widehat{V}_\perp\end{bmatrix}\right\|_F^2 \\ &= \left\|\begin{bmatrix}\widehat{\Lambda} - \widehat{V}^T A\widehat{V} & \widehat{V}^T A\widehat{V}_\perp \\ \widehat{V}_\perp^T A\widehat{V} & -\Pi(\widehat{V}_\perp^T A\widehat{V}_\perp)\end{bmatrix}\right\|_F^2,\end{aligned} \tag{4.13}$$

where the term in the lower right hand corner is formed using the identity $(I - \Pi_{\mathbb{H}_-}) = \Pi$. Considering next the norm of the residual $(A\widehat{V} - \widehat{V}\widehat{\Lambda})$ and applying a unitary transformation again, we have

$$\|R\|_F^2 = \|A\widehat{V} - \widehat{V}\widehat{\Lambda}\|_F^2 = \left\|\begin{bmatrix}\widehat{V}^T \\ \widehat{V}_\perp^T\end{bmatrix}(A\widehat{V} - \widehat{V}\widehat{\Lambda})\right\|_F^2 = \left\|\begin{bmatrix}\widehat{V}^T A\widehat{V} - \widehat{\Lambda} \\ \widehat{V}_\perp^T A\widehat{V}\end{bmatrix}\right\|_F^2. \tag{4.14}$$

The result then follows from combination of (4.12), (4.13) and (4.14). □

Theorem 4.2.1 makes no assumption about the relationship of the matrix A to the matrix $\widehat{V}\widehat{\Lambda}\widehat{V}^T$. If we further assume that the latter matrix has been constructed from an approximation of $\Pi(A)$ based on the Rayleigh-Ritz procedure, then we can go a bit further:

Corollary 4.2.2. *Suppose that $(\widehat{V}, \widehat{\Lambda})$ in Theorem 4.2.1 satisfy the further relation $\widehat{\Lambda} = \widehat{V}^T A \widehat{V}$. Then*

$$\left\| \widehat{V}\widehat{\Lambda}\widehat{V}^T - \Pi(A) \right\|_F^2 \leq 2\|R\|_F^2 + \|D_+\|_F^2. \tag{4.15}$$

Proof. The top block in the rightmost expression in (4.14) becomes zero by assumption, leaving the relation $\|A\widehat{V} - \widehat{V}\widehat{\Lambda}\| = \left\|\widehat{V}_\perp^T A \widehat{V}\right\|$ to be applied in (4.11). □

The term $\|D_+\|_F$ can be bounded by approximate computation of the largest eigenvalues of $(\widehat{V}_\perp)^T A \widehat{V}_\perp$ by e.g. Lanczos (in which instead of applying the unknown $\widehat{V}_\perp$, we can apply $I - \widehat{V}_+\widehat{V}_+^T$). With a stable computation, we expect $\|D_+\|$ to be very small. It is identically zero if $\widehat{V}_\perp$ contains 'most' of the negative eigenspace of A, in that $\widehat{V}_\perp^T A \widehat{V}_\perp \preceq 0$:

Corollary 4.2.3. *Suppose that the assumptions of Theorem 4.2.1 and Corollary 4.2.2 hold, and in addition $\widehat{V}_\perp^T A \widehat{V}_\perp \preceq 0$. Then*

$$\left\| \widehat{V}\widehat{\Lambda}\widehat{V}^T - \Pi(A) \right\|_F^2 \leq 2\|R\|_F^2. \tag{4.16}$$

Proof. Obvious since the assumption about $\widehat{V}_\perp$ zeros the term being projected in (4.15). □

It is noteworthy how concise the proofs are—perhaps verging on appearing trivial. The key fact enabling this is the nonexpansiveness of the operator Π, along with the introduction of the auxiliary matrix B. We also note that the above proof provides little insight into why a small gap does not harm the bound. In Section 4.3 we present a first-principles derivation for bounding (4.8), which demonstrates clearly why the bounds are independent of the gap.

Before concluding this subsection we present an example that establishes sharpness of Corollary 4.2.3 up to a constant smaller than 2, i.e., we show that $\left\| \widehat{V}\widehat{\Lambda}\widehat{V}^T - \Pi(A) \right\|_F^2 > \|R\|_F^2$ can be possible:

Example 4.2.1. Consider

$$A = \begin{bmatrix} 1 & -1 \\ -1 & 0 \end{bmatrix}, \quad \widehat{V} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \text{ and } \quad \widehat{\Lambda} = \widehat{V}^T A \widehat{V} = [1], \tag{4.17}$$

in which $R = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $\widehat{V}\widehat{\Lambda}\widehat{V}^T = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $\Pi(A) = \begin{bmatrix} \sqrt{5}+3 & -\sqrt{5}-1 \\ -\sqrt{5}-1 & 2 \end{bmatrix}/(2\sqrt{5})$, and

$$\left\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - \Pi(A)\right\|_F^2 / \|R\|_F^2 \approx 1.2764 > 1.$$

4.2.1 When A is nearly positive definite

In practice, projection onto the semidefinite cone can be done in two ways:

i) compute the *positive* eigenpairs $\widehat{V}, \widehat{\Lambda}$ such that $\widehat{V}\widehat{\Lambda}\widehat{V}^T \approx V_+\Lambda_+V_+^T$ (which we implicitly assumed in the above arguments), or

ii) compute the *negative* eigenpairs $\widehat{V}_-, \widehat{\Lambda}_-$ such that $\widehat{V}\widehat{\Lambda}\widehat{V}^T \approx V_-\Lambda_-V_-^T$, and obtain the approximate projection as $A - \widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T$.

The former approach is conceptually more straightforward and is efficient when A has a small number of positive eigenvalues. By contrast, the latter approach is much more efficient when A is nearly positive definite, with the number of negative eigenvalues being small relative to the matrix size n.

We expect the second situation to be equally common if not more, and it is therefore important to derive analogous bounds applicable in case (ii). Fortunately, this is a trivial extension. We note that

$$\begin{aligned} &\left\|(A - \widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T) - \Pi(A)\right\|_F^2 = \left\|(A - \widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T) - (A - V_-\Lambda_-V_-^T)\right\|_F^2 \\ =&\left\|\widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T - V_-\Lambda_-V_-^T\right\|_F^2 = \left\|\widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T - (\Pi_{\mathbb{H}_-}(A))\right\|_F^2 \\ =&\left\|\widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T - (-\Pi(-A))\right\|_F^2, \end{aligned}$$

which is the accuracy of $\widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T$ as an approximate projection of the matrix $-A$ onto the semidefinite cone. We can therefore invoke the above results with $A \leftarrow -A$ to obtain the following:

Corollary 4.2.4. *Suppose that the matrix $A \in \mathbb{S}^n$, $\widehat{V}_-$ is a $n \times k$ orthonormal matrix with $k \leq n$ and $\widehat{\Lambda}$ is a $k \times k$ negative semidefinite matrix. Then writing $R = A\widehat{V}_- - \widehat{V}_-\widehat{\Lambda}_-$ and $D_- = \Pi_{\mathbb{H}_-}(\widehat{V}_{-,\perp}^T A \widehat{V}_{-,\perp})$,*

$$\left\|(A - \widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T) - \Pi(A)\right\|_F^2 \leq \|R\|_F^2 + \left\|\widehat{V}_{-,\perp}^T A \widehat{V}_-\right\|_F^2 + \|D_-\|_F^2. \tag{4.18}$$

If in addition we have $\widehat{\Lambda}_- = \widehat{V}_-^T A \widehat{V}_-$, then

$$\left\|(A - \widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T) - \Pi(A)\right\|_F^2 \leq 2\|R\|_F^2 + \|D_-\|_F^2, \tag{4.19}$$

and if furthermore $\widehat{V}_{-,\perp}^T A \widehat{V}_{-,\perp} \succeq 0$, then

$$\left\|(A - \widehat{V}_-\widehat{\Lambda}_-\widehat{V}_-^T) - \Pi(A)\right\|_F^2 \leq 2\|R\|_F^2. \tag{4.20}$$

4.2.2 Extension to other norms

As we have already mentioned in the Introduction, a closest positive definite matrix to $A \in \mathbb{H}^n$ can be obtained by (4.2) for *any* unitarily invariant norm as a distance metric. This is shown in the following Lemma:

Lemma 4.2.5. *Given $A \in \mathbb{H}^n$ with eigendecomposition defined in* (4.1)*, $V_+\Lambda_+V_+^T$ is a solution to $\min_{X \succeq 0}\|A - X\|$ for any unitarily invariant norm.*

Proof. Consider any $X \in \mathbb{H}_+^n$ and denote with $\alpha_1 \geq \ldots \alpha_k \geq 0 \geq \alpha_{k+1} \cdots \geq \alpha_n$ the eigenvalues of A and $\chi_1 \geq \ldots \chi_n \geq 0$ those of X. Then according to [83, Corollary 7.4.9.3] and following [83, (7.4.9.2)] we have:

$$\begin{aligned}
\|A - X\| &\geq \|\mathrm{diag}(\alpha_1 - \chi_1, \ldots, \alpha_n - \chi_n)\| \\
&\geq \|\mathrm{diag}(0, \ldots, 0, \alpha_{k+1} - \chi_{k+1}, \ldots, \alpha_n - \chi_n)\| \\
&\geq \|\mathrm{diag}(0, \ldots, 0, \alpha_{k+1}, \ldots, \alpha_n)\|.
\end{aligned}$$

Thus $\|A - X\| \geq \|\mathrm{diag}(0, \ldots, 0, \alpha_k, \ldots, \alpha_n)\|$, with equality obtained in the final inequality by selecting $X = V_+\Lambda_+V_+^T$. □

Lemma 4.2.5 has already been proven for the Frobenius and the spectral norm in the literature [79], but we are unaware of a proof that considers any unitarily invariant norm.

The projector to the semidefinite cone, i.e. $\Pi(A) := \operatorname{argmin}_{X\succeq 0}\|A-X\|$, is not unique in general (though it is for the Frobenius norm). For example, the projection error could be defined as $\min_{X\in\Pi(A)}\left\|\widetilde{V}\widetilde{\Lambda}\widetilde{V}^T - X\right\|_2$. Nevertheless, for the rest of this section we consider the upper bound $\left\|\widetilde{V}\widetilde{\Lambda}\widetilde{V}^T - V_+\Lambda_+V_+^T\right\|$ for reasons of simplicity and define $\Pi(A)$ according to (4.2).

Perhaps surprisingly, the proof of Theorem 4.2.1 does not carry over to other norms, including the spectral norm. Specifically, the nonexpansiveness (4.12) of Π does not extend to every unitarily invariant norm; simple computations reveal counterexamples for e.g. the spectral norm $\|A\|_2 = \sigma_1(A)$ and trace norm $\|A\|_* = \sum_i \sigma_i(A)$.

Let us further investigate the spectral norm in particular. The issue is that for $n \geq 2$ there exist $A, B \in \mathbb{H}^n$ such that $\|\Pi(A)-\Pi(B)\|_2 > \|A-B\|_2$. An example is

$$A = \begin{bmatrix} M & 1 \\ 1 & 1/M \end{bmatrix}, \quad B = \begin{bmatrix} M+1 & 0 \\ 0 & -1+1/M \end{bmatrix},$$

which as $M \to \infty$ gives $\|\Pi(A)-\Pi(B)\|_2/\|A-B\|_2 \to \frac{\sqrt{5}+1}{2\sqrt{2}} \approx 1.1441$.

The potential expansiveness of Π might be explained as follows. Let $[V_{B_+}\ V_{B_-}]$ be the orthogonal matrix of B's eigenvectors such that

$$B = \begin{bmatrix} V_{B_+} & V_{B_-} \end{bmatrix} \begin{bmatrix} \Lambda_{B_+} & \\ & \Lambda_{B_-} \end{bmatrix} \begin{bmatrix} V_{B_+} & V_{B_-} \end{bmatrix}^T,$$

where $\Lambda_{B_+} \succ 0, \Lambda_{B_-} \preceq 0$ are diagonal. Then $\Pi(B) = [V_{B_+}\ V_{B_-}]\left[\begin{smallmatrix} \Lambda_{B_+} & \\ & 0 \end{smallmatrix}\right][V_{B_+}\ V_{B_-}]^T$. Supposing $A \succeq 0$, write $A = [V_{B_+}\ V_{B_-}]\begin{bmatrix} \widetilde{A}_{11} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{bmatrix}[V_{B_+}\ V_{B_-}]^T$. Thus

$$\|\Pi(A)-\Pi(B)\|_2 = \left\| \begin{bmatrix} \widetilde{A}_{11}-\Lambda_{B_+} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{bmatrix} \right\|_2,$$

and potential expansiveness of Π means that this might be larger than

$$\|A-B\|_2 = \left\| \begin{bmatrix} \widetilde{A}_{11}-\Lambda_{B_+} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22}-\Lambda_{B_-} \end{bmatrix} \right\|_2,$$

which is counterintuitive as we clearly have $\|\widetilde{A}_{22}\|_2 \leq \|\widetilde{A}_{22} - \Lambda_{B_-}\|_2$, as $\widetilde{A}_{22}$ and $-\Lambda_{B_-}$ are both positive semidefinite. This fact is related to a classical result by Davis, Kahan and Weinberger [84] on norm-preserving dilation, which implies that it is possible for the strict inequality

$$\left\|\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}\right\|_2 < \left\|\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & 0 \end{bmatrix}\right\|_2 \tag{4.21}$$

to hold, even when the matrices are Hermitian. For example, consider the case where $\widetilde{A}_{22}$ is negligible relative to $-\Lambda_{B_-}$. Then the above comparison essentially reduces to that of $\left\|\begin{bmatrix} \widetilde{A}_{11}-\Lambda_{B_+} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & 0 \end{bmatrix}\right\|_2$ and $\left\|\begin{bmatrix} \widetilde{A}_{11}-\Lambda_{B_+} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & -\Lambda_{B_-} \end{bmatrix}\right\|_2$, which is in the form treated by Davis-Kahan-Weinberger.

Note that while it is possible for (4.21) to hold, we still have

$$\left\|\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & 0 \end{bmatrix}\right\|_2 < \sqrt{\|[X_{11}\ X_{12}]\|_2^2 + \|[X_{21}\ 0]\|_2^2} \leq \sqrt{2}\left\|\begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}\right\|_2.$$

Similarly, for any $A \succeq 0$ we have

$$\begin{aligned} \|A - \Pi(B)\|_2 &= \left\|\begin{bmatrix} \widetilde{A}_{11} - \Lambda_{B_+} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{bmatrix}\right\|_2 \\ &\leq \sqrt{\|[\widetilde{A}_{11} - \Lambda_{B_+}, \widetilde{A}_{12}]\|_2^2 + \|[\widetilde{A}_{21}, \widetilde{A}_{22}]\|_2^2} \\ &\leq \sqrt{\|[\widetilde{A}_{11} - \Lambda_{B_+}, \widetilde{A}_{12}]\|_2^2 + \|[\widetilde{A}_{21}, \widetilde{A}_{22} - \Lambda_{B_-}]\|_2^2} \\ &\leq \sqrt{2}\left\|\begin{bmatrix} \widetilde{A}_{11} - \Lambda_{B_+} & \widetilde{A}_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} - \Lambda_{B_-} \end{bmatrix}\right\|_2 = \sqrt{2}\|A - B\|_2. \end{aligned}$$

We conclude that $\|\Pi(A) - \Pi(B)\|_2 \leq \sqrt{2}\|\Pi(A) - B\|_2$ for any $A, B \in \mathbb{H}^n$, which shows quasi-nonexpansiveness of $\frac{1}{\sqrt{2}}\Pi$ in the spectral norm [81]. Unfortunately, this result is not sufficient to derive a bound like (4.16) using the techniques of Section 4.2.

Nevertheless, based on randomly generated examples, we conjecture that, when $\widehat{V}^T A \widehat{V} \succeq 0$, we have $\left\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - \Pi(A)\right\|_2^2 \leq \delta\|R\|_2^2$ for a "small" constant δ, perhaps $\delta = 2$. Note that the counterexample 4.2.1 gives

$$\left\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - \Pi(A)\right\|_2^2 / \|R\|_2^2 \approx 1.0935 > 1,$$

thus the conjectured constant δ has to be larger than 1.

4.3 Bounding $\|A(V_+V_+^T - \widehat{V}\widehat{V}^T)\|_F$

We now turn to the alternative measure (4.8) for the projection accuracy. First note that from $AV_+ = V_+\Lambda_+$ and $A\widehat{V} = \widehat{V}\widehat{\Lambda} + R$ we have

$$AV_+V_+^T = V_+\Lambda_+V_+^T, \qquad A\widehat{V}\widehat{V}^T = \widehat{V}\widehat{\Lambda}\widehat{V}^T + R\widehat{V}^T,$$

and hence

$$AV_+V_+^T - A\widehat{V}\widehat{V}^T = V_+\Lambda_+V_+^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T - R\widehat{V}^T.$$

Therefore we have

$$\|V\Lambda_+V^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\| - \|R\| \le \|A(V_+V_+^T - \widehat{V}\widehat{V}^T)\| \le \|V\Lambda_+V^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\| + \|R\|. \tag{4.22}$$

The two accuracy measures $\|V\Lambda_+V^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\|$ and $\|A(V_+V_+^T - \widehat{V}\widehat{V}^T)\|$ are therefore at most $\|R\|$ apart; this immediately gives the bound $\|A(V_+V_+^T - \widehat{V}\widehat{V}^T)\|_F \le \sqrt{2\|R\|_F^2 + \|D_+\|_F^2} + \|R\|_F$ as a corollary of Theorem 4.2.1.

Here we follow a different argument to bound $\|A(V_+V_+^T - \widehat{V}\widehat{V}^T)\|$ directly, producing a tighter result. Moreover, while the proof is longer than in Theorem 4.2.1, it clearly reveals how the gap-independence comes about.

In what follows we assume $\widehat{\Lambda}$ is obtained by the Rayleigh-Ritz process, i.e., $\widehat{\Lambda} = \widehat{V}^TA\widehat{V}$. Furthermore, we define $[\lambda_1 \ \dots \ \lambda_n]^T := [\mathrm{diag}(\Lambda_+)^T \ \mathrm{diag}(\Lambda_-)^T]^T$ and $[v_1 \ \dots v_n] := [V_+ \ V_-]$.

Theorem 4.3.1. *Under the notation and assumptions in Theorem 4.2.1,*

$$\|A(V_+V_+^T - \widehat{V}\widehat{V}^T)\|_F^2 \le 2\|R\|_F^2 + 2\|R\|_F\|D_+\|_F + \|D_+\|_F^2 \le 2(\|R\|_F + \|D_+\|_F)^2. \tag{4.23}$$

Proof. Following the arguments in [1, §2.5.3] we have for any unitarily invariant norm

$$\begin{aligned}\|A(V_+V_+^T - \widehat{V}\widehat{V}^T)\| &= \left\|[V_+ \ V_-]\begin{bmatrix}\Lambda_+ & \\ & \Lambda_-\end{bmatrix}\begin{bmatrix}V_+ & V_-\end{bmatrix}^T(V_+V_+^T - \widehat{V}\widehat{V}^T)\begin{bmatrix}\widehat{V} & \widehat{V}_\perp\end{bmatrix}\right\| \\ &= \left\|\begin{bmatrix}\Lambda_+ & 0 \\ 0 & \Lambda_-\end{bmatrix}\begin{bmatrix}0 & V_+^T\widehat{V}_\perp \\ (V_-)^T\widehat{V} & 0\end{bmatrix}\right\|. \end{aligned} \tag{4.24}$$

Let us examine the size of the ith row of the $(1,2)$ block of the rightmost matrix term, i.e., $\|v_i^T\widehat{V}_\perp\|_2$. This is precisely the sine of the angle between $\widehat{V}$ and v_i. Assume, without loss of generality, that $\widehat{V}_\perp$ was chosen such that $A\widehat{V}_\perp = \widehat{V}_\perp D + \widehat{R}$ where $D = \begin{bmatrix} D_+ & \\ & D_- \end{bmatrix}$ and $\widetilde{D} = \begin{bmatrix} 0 & \\ & D_- \end{bmatrix}$, is diagonal and $\|\widehat{R}\| = \|R\|$ in any unitarily invariant norm. Then,

$$v_i^T A\widehat{V}_\perp = \lambda_i v_i^T\widehat{V}_\perp = v_i^T\widehat{V}_\perp D + v_i^T\widehat{R} = v_i^T\widehat{V}_\perp\widetilde{D} + v_i^T\widehat{V}_\perp\begin{bmatrix} D_+ & \\ & 0 \end{bmatrix} + v_i^T\widehat{R}.$$

Hence we have

$$v_i^T\widehat{V}_\perp(\lambda_i I - \widetilde{D}) = v_i^T(\widehat{V}_\perp\begin{bmatrix} D_+ & \\ & 0 \end{bmatrix} + \widehat{R}),$$

thus $v_i^T\widehat{V}_\perp = v_i^T(\widehat{V}_\perp\begin{bmatrix} D_+ \\ 0 \end{bmatrix} + \widehat{R})(\lambda_i I - \widetilde{D})^{-1}$ and the inverse is guaranteed to exist since λ_i is necessarily positive. Therefore, using the fact $\widetilde{D} \preceq 0$ we obtain

$$\|v_i^T\widehat{V}_\perp\|_2 \leq \frac{\|v_i^T(\widehat{V}_\perp\begin{bmatrix} D_+ \\ 0 \end{bmatrix} + \widehat{R})\|_2}{\min_j\left|\lambda_i - \widetilde{D}_{(j,j)}\right|} \leq \frac{\|v_i^T(\widehat{V}_\perp\begin{bmatrix} D_+ \\ 0 \end{bmatrix} + \widehat{R})\|_2}{\lambda_i}. \tag{4.25}$$

Since this holds for $i = 1, \ldots, k$, it follows that the ith row of $\Lambda_+ V_+^T\widehat{V}_\perp$ is bounded by $\|v_i^T(\widehat{V}_\perp\begin{bmatrix} D_+ \\ 0 \end{bmatrix} + \widehat{R})\|_2$. Note how the $\Lambda_{+(i,i)} = \lambda_i$ and $1/\lambda_i$ terms neatly cancel out; this is why the result is gap-independent. Together with the bounds for $i = 1, \ldots, k$, we obtain

$$\|\Lambda_+ V_+^T\widehat{V}_\perp\|_F \leq \|V_+^T(\widehat{V}_\perp\begin{bmatrix} D_+ \\ 0 \end{bmatrix} + \widehat{R})\|_F \leq \|\widehat{V}_\perp\begin{bmatrix} D_+ \\ 0 \end{bmatrix} + \widehat{R}\|_F \leq \|D_+\|_F + \|\widehat{R}\|_F, \tag{4.26}$$

where we used the fact that V_+ has orthonormal columns, hence $\|V_+^T X\|_F \leq \|X\|_F$ for any matrix X.

Similarly, we examine the ith row of $V_-^T\widehat{V}$, which appears in the $(2,1)$ block of (4.24). From $A\widehat{V} = \widehat{V}\widehat{\Lambda} + R$ we have

$$v_{k+i}^T A\widehat{V} = \lambda_{k+i} v_{k+i}^T\widehat{V} = v_{k+i}^T(\widehat{V}\widehat{\Lambda} + R).$$

Hence we have

$$v_{k+i}^T\widehat{V}(\lambda_{k+i} I - \widehat{\Lambda}) = v_{k+i}^T R,$$

so $v_{k+i}^T \widehat{V} = v_{k+i}^T R(\lambda_{k+i} I - \widehat{\Lambda})^{-1}$, giving

$$\|v_{k+i}^T \widehat{V}\| \leq \frac{\|v_{k+i}^T R\|}{\min_j \left|\lambda_{k+i} - \widetilde{\Lambda}_{+_{(j,j)}}\right|} \leq \frac{\|v_{k+i}^T R\|}{\|\lambda_{k+i}|}.$$

Here we have used the facts that $\widehat{\Lambda} \succ 0$ and $\lambda_{k+i} \leq 0$; we assumed $\lambda_{k+i} < 0$, which can be done because the terms with $\lambda_{k+i} = 0$ do not contribute to the terms in (4.24). Since this holds for $i = k+1, \ldots, n$, the $(k+i)$th row of (4.24) is bounded in norm by $\|v_{k+i}^T R\|_2$. Thus

$$\|\Lambda_- V_-^T \widehat{V}\|_F \leq \|R\|_F. \tag{4.27}$$

Putting (4.24), (4.26) and (4.27) together, we obtain

$$\|A(V_+ V_+^T - \widehat{V}\widehat{V}^T)\|_F \leq \sqrt{2\|R\|_F^2 + 2\|R\|_F \|D_+\|_F + \|D_+\|_F^2},$$

as required. It is easy to see the final expression is upper bounded by $\sqrt{2}(\|R\|_F + \|D_+\|_F)$. □

4.4 Experiments

In this section we demonstrate the results of this chapter with a set of experiments. We begin with an experiment that demonstrates the independence of the projection accuracy on the spectral gap. We consider a set of parametric matrices $A \in \mathbb{S}^{500}$ with 20 eigenvalues geometrically distributed on $[10^{-10}, 1]$, one at ϵ, where ϵ is a parameter which is used to control the spectral gap, one at $-\epsilon$ and the rest uniformly distributed on $[-1, 0]$. We compute approximate projections by running `ARPACK` [57] (accessed via `MATLAB`'s `eigs` with its default parameters) on A requesting the 25 largest eigenpairs and returning $\widehat{V}\widehat{\Lambda}\widehat{V}^T$, where $(\widehat{V}, \widehat{\Lambda})$ are the positive Ritz pairs obtained by `ARPACK`. We let `ARPACK` iterate until convergence (set to its default value, 10^{-14}) and we do this for varying values of ϵ. In Figure 4.1 we plot the resulting projection errors as a function of the spectral gap. We observe that the projection accuracy is not impaired by small spectral gaps, and that the bound of Corollary 4.2.2 successfully estimates the high accuracy of the projection.

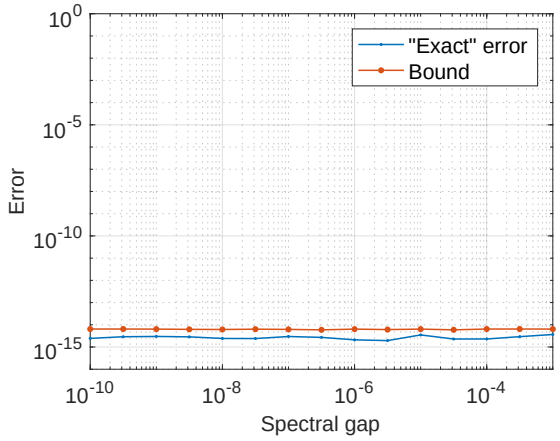


Figure 4.1: Left: "Exact" projection error $\left\|V_+\Lambda_+V_+^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\right\|_F$, where V_+, Λ_+ are obtained via a full eigenvalue decomposition and $\widehat{V}, \widehat{\Lambda}$ are computed with `ARPACK`, and the respective bound obtained via Corollary 4.2.2 for problems with varying spectral gap.

Next, we consider the accuracy of the projection for the iterates produced by `ARPACK`. We consider a matrix $A \in \mathbb{S}^{500}$ with eigenvalues distributed according to the previous experiment with $\epsilon = 10^{-10}$. We compute approximate projections by running `ARPACK` as before. Figure 4.2 (left) shows the projection accuracy as a function of `ARPACK`'s iterations. The upper bound of Corollary 4.2.2 is compared with the "exact" projection error $\left\|V_+\Lambda_+V_+^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\right\|_F$ where V_+, Λ_+ are obtained via a full eigenvalue decomposition. Although our bound eventually approximates the exact error, it exhibits oscillatory behaviour before convergence.

We will show that this oscillatory behaviour is caused by positive Ritz pairs with large residuals. The following lemma provides a criterion for excluding some positive Ritz pairs resulting in improved error bounds, which as we will see, exhibit significantly reduced oscillatory behaviour. The following lemma uses the notation $\left[a_i\right]_{i\in\mathcal{E}}$ for horizontal concatenation of the vectors $a_{\mathcal{E}_{[1]}}, a_{\mathcal{E}_{[2]}}, \ldots$ where $\mathcal{E}_{[i]}$ is the $i-$th smallest element of the set of indices $\mathcal{E}$.

Lemma 4.4.1. *Suppose that* $\widehat{\Lambda} = diag(\hat{\lambda}_1, \ldots, \hat{\lambda}_k) \succeq 0$ *and* $\widehat{V} = [\hat{v}_1 \ \ldots \ \hat{v}_k]$ *are a set of Ritz pairs for some* $A \in \mathbb{H}^n$. *Furthermore, consider any* $\widetilde{V} = [\hat{v}_i]_{i\notin\mathcal{E}}$ $\widetilde{\Lambda} = \mathrm{diag}([\hat{\lambda}_i]_{i\notin\mathcal{E}})$ *where* $\mathcal{E} \subseteq \{1, \ldots, k\}$ *with every* $i \in \mathcal{E}$ *satisfying*

$$(\sqrt{2}-1)\|r_i\|_2 > \max\Big(\hat{\lambda}_i, \lambda_{\max}(\widehat{V}_\perp^T A \widehat{V}_\perp)\Big), \tag{4.28}$$

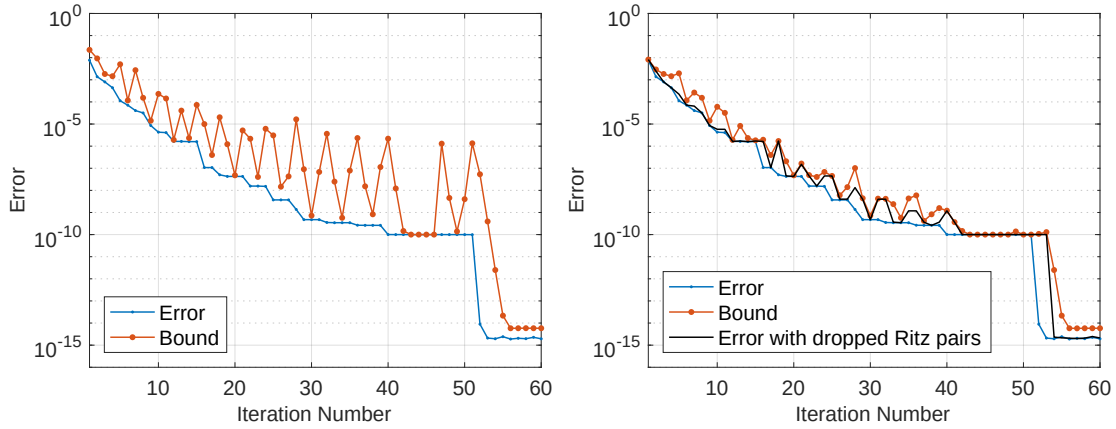


Figure 4.2: Left: Convergence of the "exact" projection error $\left\|V_+\Lambda_+V_+^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\right\|_F$, where V_+, Λ_+ are obtained via a full eigenvalue decomposition and $\widehat{V}, \widehat{\Lambda}$ are computed with `ARPACK`, and the respective bound obtained via Corollary 4.2.2. Right: Convergence of $\left\|V_+\Lambda_+V_+^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\right\|_F$, $\left\|V_+\Lambda_+V_+^T - \widetilde{V}\widetilde{\Lambda}\widetilde{V}^T\right\|_F$ (black solid) and the respective bound obtained by applying Corollary 4.2.2 for $\widetilde{V}\widetilde{\Lambda}\widetilde{V}^T$.

with $r_i := A\hat{v}_i - \hat{\lambda}_i\hat{v}_i$. *Then, the bound of Corollary 4.2.2 for the projection error*

$$\left\|\widetilde{V}\widetilde{\Lambda}\widetilde{V}^T - V_+\Lambda_+V_+^T\right\|_F$$

is smaller (i.e. better) than the respective bound of $\left\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\right\|_F$.

Proof. We will first prove the case where $\mathcal{E}$ contains a single index and then generalize for the case where $\mathcal{E}$ has multiples elements.

Without loss of generality, assume that $\mathcal{E} = \{1\}$. Define $[\hat{v}_1 \ \widehat{V}_2] := \widehat{V}$, $\begin{bmatrix}\hat{\lambda}_1 & 0\\ 0 & \widehat{\Lambda}_2\end{bmatrix} := \widehat{\Lambda}$, and $[r_1 \ R_2] := R$. Corollary 4.2.2 gives the following bound for the approximate projection $\widehat{V}\widehat{\Lambda}\widehat{V}^T = \hat{\lambda}_1\hat{v}_1\hat{v}_1^T + \widehat{V}_2\widehat{\Lambda}_2\widehat{V}_2^T$:

$$\left\|\widehat{V}\widehat{\Lambda}\widehat{V}^T - V_+\Lambda_+V_+^T\right\|_F^2 \leq \underbrace{2\|R_2\|_F^2 + 2\|r_1\|_2^2 + \|\Pi(D)\|_F^2}_{:=\text{bound}_1} \tag{4.29}$$

where $D := \widehat{V}_\perp^T A\widehat{V}_\perp$. If we do not include $\hat{\lambda}_1\hat{v}_1\hat{v}_1^T$ in our approximate projection, then we obtain the following bound:

$$\left\|\widehat{V}_2\widehat{\Lambda}_2\widehat{V}_2^T - V_+\Lambda_+V_+^T\right\|_F \leq \underbrace{2\|R_2\|_F^2 + \left\|\Pi(\tilde{D})\right\|_F^2}_{:=\text{bound}_2}$$

where $\tilde{D} := [\hat{v}_1 \ \widehat{V}_\perp]^T A[\hat{v}_1 \ \widehat{V}_\perp]$. Note that

$$\text{bound}_2 - \text{bound}_1 = -2\|r_1\|_2^2 + \left\|\Pi(\tilde{D})\right\|_F^2 - \|\Pi(D)\|_F^2 \tag{4.30}$$

and denote with $\mu_1 \leq \cdots \leq \mu_{k+1}$ the eigenvalues of D and $\tilde{\mu}_1 \leq \cdots \leq \tilde{\mu}_k$ those of $\tilde{D}$. Then, using [56, Theorem 10.1.1] on $\tilde{D} = \begin{bmatrix} \hat{\lambda}_1 & \hat{v}_1^T A \widehat{V}_\perp \\ \widehat{V}_\perp^T A \hat{v}_1 & \widehat{V}_\perp^T A \widehat{V}_\perp \end{bmatrix}$ we get $\tilde{\mu}_i \leq \mu_i$ where $i = 1, \ldots k$. Thus,

$$\left\|\Pi(\tilde{D})\right\|_F^2 - \|\Pi(D)\|_F^2 = \sum_{i=1}^{k+1} \max^2(\tilde{\mu}_i, 0) - \sum_{i=1}^{k} \max^2(\mu_i, 0) \tag{4.31}$$

$$\leq \max^2(\tilde{\mu}_{k+1}, 0). \tag{4.32}$$

Furthermore, using [56, Theorem 10.3.1] on $\tilde{D} = \begin{bmatrix} \hat{\lambda}_1 & 0 \\ 0 & D \end{bmatrix} + \begin{bmatrix} 0 & \hat{v}_1^T A \widehat{V}_\perp \\ \widehat{V}_\perp^T A \hat{v}_1 & 0 \end{bmatrix}$ we get

$$\tilde{\mu}_{l+1} \leq \max(\mu_l, \hat{\lambda}_1) + \left\| \begin{bmatrix} 0 & \hat{v}_1^T A \widehat{V}_\perp \\ \widehat{V}_\perp^T A \hat{v}_1 & 0 \end{bmatrix} \right\|_2 = \max(\mu_l, \hat{\lambda}_1) + \|r\|_2, \tag{4.33}$$

where the last equality holds because $\widehat{V}$ and $\widehat{\Lambda}$ were obtained by performing the Rayleigh-Ritz process on A (see (4.14)).

Combining (4.30), (4.32) and (4.33) gives

$$\text{bound}_2 \leq \text{bound}_1 \quad \text{when} \quad \max(\mu_l, \hat{\lambda}) \leq \left(\sqrt{2} - 1\right)\|r\|_2. \tag{4.34}$$

This completes the proof for the case where $\mathcal{E}$ is a singleton.

Consider now the case where $\mathcal{E} = \{i_1, \ldots, i_l\}$ with $l > 1$. According to the proceeding part of this proof, the Ritz pairs $([\hat{\lambda}_i]_{i \neq i_1}, [\hat{v}_i]_{i \neq i_1})$ will produce an approximate projection with a lower approximation bound (obtained via Corollary 4.2.2) than those of $\widehat{V}\widehat{\Lambda}\widehat{V}^T$. Likewise, $([\hat{\lambda}_i]_{i \neq i_1, i_2}, [\hat{v}_i]_{i \neq i_1, i_2})$ will produce an approximate projection with a further improved bound if

$$(\sqrt{2} - 1)\|r_{i_2}\|_2 > \max\left(\hat{\lambda}_{i_2}, \lambda_{\max}(W^T A W)\right). \tag{4.35}$$

where W is an orthonormal matrix spanning the nullspace of $[\hat{v}_i]_{i \neq i_1}^T$. However, (4.35) is implied by (4.28) because $\text{span}(W) \subset \text{span}(\widehat{V}_\perp)$ and thus $\lambda_{\max}(W^T A W) \leq \lambda_{\max}(\widehat{V}_\perp^T A \widehat{V}_\perp)$ [54, Corollary 4.1]. Since the same argument holds for the rest of the indices contained in $\mathcal{E}$, this concludes the proof. □

Figure 4.2 (right) aims to demonstrate the usefulness of Lemma 4.4.1 in reducing the oscillations of Figure 4.2 (left). It considers the same experiment as Figure 4.2

(left), but compares the convergence of the "exact" accuracy of $\left\|V_+\Lambda_+V_+^T - \widetilde{V}\widetilde{\Lambda}\widetilde{V}^T\right\|_F$ where $(\widetilde{\Lambda}, \widetilde{V})$ consists of the positive Ritz pairs contained in $(\widehat{\Lambda}, \widehat{V})$ for which $(\sqrt{2}-1)\|r_i\|_2 \leq \max\left(\hat{\lambda}_i, \lambda_{\max}(\widehat{V}_\perp^T A \widehat{V}_\perp)\right)$, with the respective bound obtained by applying Corollary 4.2.2. We observe a significant reduction in the oscillatory behaviour of our bound as compared to Figure 4.2 (left). We further include in Figure 4.2 (right) the convergence of $\left\|V_+\Lambda_+V_+^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\right\|_F$ for comparison. It is worth noting that, in this experiment, $\left\|V_+\Lambda_+V_+^T - \widehat{V}\widehat{\Lambda}\widehat{V}^T\right\|_F$ converges almost monotonically and it is never greater than either of $\left\|V_+\Lambda_+V_+^T - \widetilde{V}\widetilde{\Lambda}\widetilde{V}^T\right\|_F$ or our bound.

5

Optimal Approximation of Doubly Stochastic Matrices

5.1 Introduction

We now turn our attention in using ADMM for a particular Quadratic Programming problem. In contrast to SDPs, in Quadratic Programming ADMM spends most of its time in solving linear systems. Thus, recognizing structure in the KKT systems is a key driver of speed. We show how this can be done in the following optimization problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|X - C\|^2 \\ \text{subject to} & X \text{ nonnegative (elementwise)} \\ & X_{(i,j)} = 0 \; \forall \; (i,j) \text{ with } C_{(i,j)} = 0 \\ & X\mathbf{1} = \mathbf{1}, \;\; X^T\mathbf{1} = \mathbf{1}, \end{array} \tag{P}$$

which approximates the symmetric real, $n \times n$ matrix C under the Frobenius norm in the set of *doubly stochastic* matrices $X \in \mathbb{R}^{n \times n}$ (also known as the Birkhoff polytope [85]), i.e., the set of matrices with nonnegative elements whose columns and rows sum to one, with the added constraint that the solution must have the same sparsity pattern as C. Problem (P) is a *matrix nearness* problem, i.e., a problem of finding a matrix with certain properties that is close to some given matrix; see [79] for a survey on matrix nearness problems.

Adjusting a matrix so that it becomes doubly stochastic is relevant in many fields, e.g., for preconditioning linear systems [86], as a normalization tool used in spectral clustering [87], and image filtering [88], or as a tool to estimate a doubly stochastic matrix from incomplete or approximate data used e.g., in longitudinal studies in life sciences [89], or to analyze the 3D structure of the human genome [90]

A related and widely used approach is to search for a diagonal matrix D such that DCD is doubly stochastic. This is commonly referred to as the *matrix balancing* problem [86]. Such a scaling matrix D exists and is unique whenever C has total support. Perhaps surprisingly, when C has only nonnegative elements, then the matrix balancing problem can be considered as a matrix nearness problem. This is because DCD has been shown to minimize the relative entropy measure [91, Observation 3.19], i.e., it is the solution of the following convex problem

$$\begin{array}{ll} \text{minimize} & \sum_{i,j} X_{(i,j)} \log X_{(i,j)}/C_{(i,j)} \\ \text{subject to} & X \text{ doubly stochastic} \\ & X_{(i,j)} = 0 \;\forall\; (i,j) \text{ with } C_{(i,j)} = 0, \end{array} \tag{5.1}$$

where we define $0 \cdot log(0) = 0$, $0 \cdot log(0/0) = 0$, and $1 \cdot log(1/0) = \infty$. Note, however, that the relative entropy is not a proper distance metric since, it is not symmetric, does not satisfy the triangular inequality and can take the value ∞.

The *matrix balancing problem* can be solved by iterative methods with remarkable scalability. Standard and simple iterative methods, such as the Sinkhorn-Knopp algorithm, exist that exhibit linear per-iteration complexity w.r.t. the number of nonzeros in C and linear convergence rate [86], [91]. More recent algorithms can exhibit a super-linear convergence rate and increased robustness in many practical situations [92]. Furthermore, efficient algorithms exists for the related *mass transportation problem*, where we seek the minimization of a linear objective in the set of doubly stochastic matrices [93].

The aim of this chapter is to show that the direct minimization of a least squares objective in doubly stochastic approximation, which has a long history dating back to the influential paper of [94][1], can also be solved efficiently. This

[1] [94] consider the weighted least squares cost $\frac{1}{2}\sum_{i,j}[X_{(i,j)} - C_{(i,j)}]^2/C_{(i,j)}$ which we treat in §5.3.1.

gives practitioners a new solution to a very important problem of doubly stochastic matrix approximation, which might prove useful for cases where the relative entropy metric is not suitable to their problem.

The approach we present is also flexible in the sense that it can handle other interesting cases, for example, where C is non-symmetric or rectangular, $\|\cdot\|$ is a weighted Frobenius norm, and $X\mathbf{1}$ and $X^T\mathbf{1}$ are required to sum to an arbitrary, given vector. We discuss these generalizations in §5.3.1.

Related work The authors of [87] consider the problem (P) in the case when C is fully dense. They suggest an alternating projections algorithm that has linear per-iteration complexity. The approach of [87] resembles the results of §5.3.1 but, as we will see in the experimental section, it is not guaranteed to converge to an optimizer.

The chapter is organized as follows: In Section 5.2, we introduce a series of reformulations to Problem (P), resulting in a problem that is much easier to solve. In Section 5.3, we suggest a solution method that reveals a particular, and useful, structure in the problem. Finally, in Section 5.4, we present a series of numerical results that highlight the scalability of the approach.

5.2 Modeling (P) efficiently

In this section, we present a reformulation of the doubly stochastic approximation problem (P) suitable for solving very large-scale problems. One of the difficulties with the original formulation, (P), is that it has n^2 variables and $2n^2+2n$ constraints. Attempting to solve (P) with an off-the-shelf QP solver, such as Gurobi [95], can result in out-of-memory issues just by representing the problem's data even for matrices with a small amount of nonzeros (in the order of tens of thousands) that are extremely sparse.

In order to avoid this issue we will perform a series of reformulations that will eliminate variables from (P). The final problem will have significantly fewer variables and constraints while maintaining a remarkable degree of structure, which will be revealed and exploited in the next section.

We first take the obvious step of eliminating all variables in (P) that are constrained to be zero, as prescribed by the constraint

$$X_{(i,j)} = 0 \text{ for all } (i,j) \text{ such that } C_{(i,j)} = 0. \tag{5.2}$$

Indeed, consider vec$(\cdot)$, the operator that stacks a matrix into a vector in a column-wise fashion and H an $n_{\text{nz}} \times n^2$ matrix, where $n_{\text{nz}} := \text{card}(C)$, that selects all the nonzero elements of vec(C). Note that H depends on the sparsity pattern S of C, defined as the $0-1$ $n \times n$ matrix

$$S_{(i,j)} := \begin{cases} 0 & C_{(i,j)} = 0 \\ 1 & \text{otherwise,} \end{cases} \tag{5.3}$$

but we will not denote this dependence explicitly. We can now isolate the nonzero variables contained in X and C in $n_{\text{nz}}-$dimensional vectors defined as

$$x := H \operatorname{vec}(X), \quad c := H \operatorname{vec}(C). \tag{5.4}$$

Note that x is simply a re-writing of X in $\mathbb{R}^{n_{\text{nz}}}$, since for any $X \in \mathcal{S}$ where

$$\mathcal{S} := \left\{ X \in \mathbb{R}^{n \times n} \;\middle|\; X_{(i,j)} = 0 \text{ for all } C_{(i,j)} = 0 \right\}$$

we have

$$H^T H \operatorname{vec}(X) = \operatorname{vec}(X) \Leftrightarrow H^T x = \operatorname{vec}(X),$$

due to the fact that $H^T H = \text{diag}(\text{vec}(S))$. Thus every $x \in \mathbb{R}^{n_{\text{nz}}}$ defines a unique $X \in \mathcal{S}$ and vice versa.

We can now describe the constraints of (P), i.e., $X \geq 0$, $X\mathbf{1}_n = \mathbf{1}_n$ and $X^T\mathbf{1}_n = \mathbf{1}_n$, on the "$x-$space". Obviously $X \geq 0$ trivially maps to $x \geq 0$. Furthermore, recalling the standard Kronecker product identity

$$\operatorname{vec}(LMN) = (N^T \otimes L) \operatorname{vec}(M) \tag{5.5}$$

for matrices of compatible dimension, the constraints $X\mathbf{1}_n = \mathbf{1}_n$ and $X^T\mathbf{1}_n = \mathbf{1}_n$ of (P) can be rewritten in an equivalent vectorized form as

$$\begin{bmatrix} \mathbf{1}_n^T \otimes I_n \\ I_n \otimes \mathbf{1}_n^T \end{bmatrix} \operatorname{vec}(X) = \mathbf{1}_{2n} \tag{5.6}$$

or, by noting that $H^T x = \text{vec}(X)$, as

$$\begin{bmatrix} \mathbf{1}_n^T \otimes I_n \\ I_n \otimes \mathbf{1}_n^T \end{bmatrix} H^T x = \mathbf{1}_{2n}. \tag{5.7}$$

Thus (P) can be rewritten in the "$x-$space" as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|x - c\|^2 \\ \text{subject to} & x \geq 0 \\ & \begin{bmatrix} \mathbf{1}_n^T \otimes I_n \\ I_n \otimes \mathbf{1}_n^T \end{bmatrix} H^T x = \mathbf{1}_{2n}. \end{array} \tag{P_1}$$

A further reduction of the variables and constraints of (P_1) can be achieved by exploiting the symmetry of C. To this end, note that when C is symmetric the optimal doubly stochastic approximation will also be symmetric according to the following proposition:

Proposition 5.2.1. *If C is symmetric, then the optimal solution X^* of* (P) *is also symmetric.*

Proof. Assume the contrary, so that X^* is optimal but asymmetric. Then the matrix $(X^*)^T$ is a feasible solution for (P) since it remains element-wise negative when its row and column sums are exchanged, and has an identical objective value since C is assumed symmetric. Then define the symmetric matrix $\tilde{X}$ as the convex combination

$$\tilde{X} := \frac{1}{2}(X^* + X^{*T})$$

which is also a feasible point for (P). Since the objective function is strictly convex (at least on the subset of elements of X not constrained to be zero), the objective function evaluated at $\tilde{X}$ will be strictly lower than that for both X^*, contradicting the optimality of X^*. □

It follows that restricting the feasible set of (P) to symmetric matrices does not affect its solution. We will exploit this by eliminating all the variables embedded into X that are below its main diagonal. Define an upper triangular matrix X_u consisting of scaled elements of X such that $X = X_u + X_u^T$, and likewise for C, i.e.

$$X_u := U \odot X, \quad C_u := U \odot C \tag{5.8}$$

where

$$U := \begin{bmatrix} \frac{1}{2} & 1 & \cdots & 1 \\ & \ddots & & \vdots \\ & & \frac{1}{2} & 1 \\ 0 & & & \frac{1}{2} \end{bmatrix}. \tag{5.9}$$

As in the previous definitions, define H_u as the matrix that stacks all the nonzeros of C_u in an column-wise fashion, which is used to extract the nonzero elements of X_u and C_u, i.e., scaled nonzero elements of the upper triangular part of X, to the vectors

$$x_u := H_u \operatorname{vec}(X_u), \quad c_u := H_u \operatorname{vec}(X_u). \tag{5.10}$$

We can now write down our reduced optimization problem. Note that, although it might not be directly evident, the following problem possesses a remarkable degree of internal structure that is exploited in the suggested solution algorithm of the next section.

Theorem 5.2.2. *Consider the following problem*

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}\|p \odot (x_u - c_u)\|^2 \\ \textit{subject to} & x_u \geq 0 \\ & Ax_u = \mathbf{1}_n, \end{array} \tag{P$_2$}$$

where

$$p := H_u \operatorname{vec}\left(\begin{bmatrix} 2 & \sqrt{2} & \cdots & \sqrt{2} \\ & \ddots & & \vdots \\ & & 2 & \sqrt{2} \\ 0 & & & 2 \end{bmatrix}\right)$$

$$A_1 := \mathbf{1}_n^T \otimes I_n, \quad A_2 := I_n \otimes \mathbf{1}_n^T$$

and $A := (A_1 + A_2)H_u^T$. *Solving* (P) *for a symmetric* C *is equivalent to solving* (5.2.2) *in the sense that* (P) *is feasible iff* (P$_2$) *is, and the optimizer* X^* *of* (P) *can be constructed from the optimizer* x_u^* *of* (P$_2$) *using* (5.10) *and* (5.8).

Proof. We will first show that every feasible x_u of (P$_2$) defines a feasible X for (P), where $\operatorname{vec}(X_u) := H_u^T x_u$, with the same objective value. Similarly to S, define $S_u \in \mathbb{R}^{n \times n}$ as an "0/1" matrix that represents the sparsity of C and the upper triangular of C respectively, i.e.

$$S_{u_{(i,j)}} = \begin{cases} 0 & C_{(i,j)} = 0, \text{ or } i < j \\ 1 & \text{otherwise.} \end{cases}$$

The equality of the objective value can be shown as follows:

$$\begin{aligned}\|p \odot (x_u - c_u)\| &= \left\|(\mathbf{1}_{n\times n}\sqrt{2} + (2-\sqrt{2})I_n) \odot (X_u - C_u)\right\| \\ &= \left\|(\mathbf{1}_{n\times n}\sqrt{2} + (2-\sqrt{2})I_n) \odot U \odot (X - C)\right\| \\ &= \left\|(\mathbf{1}_{n\times n}\sqrt{2} + (1-\sqrt{2})I_n) \odot S_U \odot (X - C)\right\| \\ &= \|S \odot (X - C)\| = \|X - C\|.\end{aligned} \tag{5.11}$$

Furthermore, similarly to (5.5)-(5.7), we have

$$\begin{aligned}X\mathbf{1}_n &= (X_u + X_u^T)\mathbf{1}_n \\ &= (\mathbf{1}_n^T \otimes I_n + I_n \otimes \mathbf{1}_n^T)\,\mathrm{vec}(X_u) \\ &= (\mathbf{1}_n^T \otimes I_n + I_n \otimes \mathbf{1}_n^T)H_u^T x_u = Ax_u,\end{aligned} \tag{5.12}$$

resulting in $X\mathbf{1}_n = \mathbf{1}_n$ due to the feasibility of x_u for (P_2). Due to the symmetry of X we also get $X^T\mathbf{1}_n$. Finally, X is nonnegative by construction and has sparsity pattern S. Therefore X is feasible for (P).

Likewise, following (5.11)-(5.12) in reverse order, we can show that every symmetric feasible matrix X of (P) defines an $x_u := H_u\,\mathrm{vec}(X_u)$, where $X_u := U \odot X$, that is feasible for (P_2) and has identical objective value. Since only symmetric optimizers exist for (P) (Lemma 5.2.1) this concludes the proof. □

Unlike (P) which has n^2 and $2n$ constraints, (P_2) has approximately $n_{\mathrm{nz}}/2$ and n constraints. Furthermore, it possesses a specific internal structure that we will exploit in the solution algorithm presented in §5.3.

5.3 Solution Method

In this section, we describe how the reduced problem (P_2) can be solved with ADMM. Problem (P_2) is a quadratic program (QP) with strictly convex objective function. Solving QPs with ADMM has been widely studied in the literature [53], [26], [30]. We will follow the approach of [53] which can solve (P_2) by applying ADMM to the following splitting

$$\begin{array}{ll}\text{minimize} & f(\tilde{x}, \tilde{z}) + g(x, z) \\ \text{subject to} & (\tilde{x}, \tilde{z}) = (x, z)\end{array} \tag{5.13}$$

where f and g, are defined as

$$f(\tilde{x}, \tilde{z}) = \frac{1}{2} x^T P x - P c_u + \mathcal{I}_{A\tilde{x}=\tilde{z}}(\tilde{x}, \tilde{z})$$
$$g(x, z) = \mathcal{I}_{x \geq 0}(x) + \mathcal{I}_{z=\mathbf{1}_{2n}}(z)$$

and $P := \operatorname{diag}(p \odot p)$, $\bar{n}_{\text{nz}}$ are the number of nonzeros in the upper triangular of C and $\mathcal{I}_{A\tilde{x}=\tilde{z}}$, $\mathcal{I}_{x\geq 0}$, and $\mathcal{I}_{z=\mathbf{1}_{2n}}$ denote the indicator functions of the sets $\{(x, z) \in \mathbb{R}^{\bar{n}_{\text{nz}}} \times \mathbb{R}^{2n} \mid Ax = z\}$, $\{x \in \mathbb{R}^{\bar{n}_{\text{nz}}} \mid x \geq 0\}$, and $\{x \in \mathbb{R}^{2n} \mid x = \mathbf{1}_{2n}\}$ respectively.

Applying ADMM for the problem (5.13) results in Algorithm 5.1, where Π_+ denotes the projection of a vector to the nonnegative orthant and $\Pi_1(x) := \mathbf{1}$ for any vector x. Refer to Section 3.2 and [53, §3] for details.

Algorithm 5.1: Solving (P_2) with ADMM

1 **given** initial values x^0, z^0, y^0 and parameters $\rho > 0$, $\sigma > 0$, and $\alpha \in (0, 2)$;
2 **repeat**
3 $\quad (\tilde{x}^{k+1}, \tilde{z}^{k+1}) \leftarrow$ solution of the linear system
$$\begin{bmatrix} (P + \sigma I_{\bar{n}_{\text{nz}}}) & \rho A^T \\ \rho A & -\rho I_{2n} \end{bmatrix} \begin{bmatrix} \tilde{x}^{k+1} \\ \tilde{z}^{k+1} \end{bmatrix} = \begin{bmatrix} \sigma x^k - w^k + A^T(\rho z^k - y^k) + P c_u \\ 0 \end{bmatrix};$$
4 $\quad x^{k+1} \leftarrow \Pi_+(\alpha \tilde{x}^{k+1} + (1-\alpha) x^k)$;
5 $\quad z^{k+1} \leftarrow \Pi_{\mathbf{1}}(\alpha \tilde{z}^{k+1} + (1-\alpha) z^k + \rho^{-1} y^k)$;
6 $\quad w^{k+1} \leftarrow w^k + \sigma(\alpha \tilde{x}^{k+1} + (1-\alpha) x^k - x^{k+1})$;
7 $\quad y^{k+1} \leftarrow y^k + \rho(\alpha \tilde{z}^{k+1} + (1-\alpha) z^k - z^{k+1})$;
8 **until** *termination condition is satisfied*;

The most computationally intensive operation of Algorithm 5.1 is the solution of the $(\bar{n}_{\text{nz}} + n) \times (\bar{n}_{\text{nz}} + n)$ linear system in line 3. We will show that its solution can be obtained by solving instead a reduced $n \times n$ linear system.

Fact 5.3.1. *Consider the following linear system*

$$\begin{bmatrix} P + \sigma I_{\bar{n}_{nz}} & \rho A^T \\ \rho A & -\rho I_{2n} \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{5.14}$$

where $x, r \in \mathbb{R}^{\bar{n}_{nz}}, z \in \mathbb{R}^{\bar{n}_{nz}}$ and $A \in \mathbb{R}^{2n \times \bar{n}_{nz}}$. Its solution can be obtained as follows

1. *Obtain z by solving the following $n \times n$ positive definite linear system*

$$(\rho A (P + \sigma I_{\bar{n}_{nz}})^{-1} A^T + I_n) z = A(P + \sigma I_{\bar{n}_{nz}})^{-1} r \tag{5.15}$$

2. Obtain x as $(P + \sigma I_{\bar{n}_{nz}})^{-1}(r - \rho A^T z)$.

Proof. The first block row gives $(P + \sigma I_{\bar{n}_{\text{nz}}})x + \rho A^T z = r \Leftrightarrow x = (P + \sigma I_{\bar{n}_{\text{nz}}})^{-1}(r - \rho A^T z)$. Reducing the variable x from (5.14) results in (5.15). □

Thus solving (5.14) can be reduced to solving a linear system with left hand side

$$(\rho A(P + \sigma I_{\bar{n}_{\text{nz}}})^{-1} A^T + I_n). \tag{5.16}$$

Fortunately, the reduced matrix (5.16), which is equal to $A\,\mathrm{diag}(H_u\,\mathrm{vec}(U))A^T$ with

$$U := \begin{bmatrix} (4+\sigma)^{-1} & (2+\sigma)^{-1} & \cdots & (2+\sigma)^{-1} \\ & \ddots & & \vdots \\ & & (4+\sigma)^{-1} & (2+\sigma)^{-1} \\ 0 & & & (4+\sigma)^{-1} \end{bmatrix},$$

turns out to be positive definite with a sparsity pattern matching that of $C + I$:

Theorem 5.3.2. *The following relation holds*

$$\begin{aligned} &A\,\mathrm{diag}(H_u\,\mathrm{vec}(D))A^T \\ =&S \odot (D + D^T) + \mathrm{diag}(S \odot (D + D^T)\mathbf{1}) \end{aligned}$$

for any upper triangular $n \times n$ matrix D.

Proof. The proof is provided in §5.A. □

The solution of the reduced linear system (5.15) can be obtained given an initial Cholesky factorization of $\rho A(P + \sigma I_{\bar{n}_{\text{nz}}})^{-1} A^T + I_n$, or even with a factorization free algorithm e.g., the CG method which requires repeated matrix-vector multiplications with $\rho A(P + \sigma I_{\bar{n}_{\text{nz}}})^{-1} A^T + I_n$. The fact that the linear system to be solved has the same sparsity pattern of $S + I$ can be particularly beneficial in cases where a fill-in reducing permutation is already known for the matrix under approximation C (and thus for S), since the same permutation could be used before the factorization of $\rho A(P + \sigma I_{\bar{n}_{\text{nz}}})^{-1} A^T + I_n$ resulting in reduced fill-in and thus significant speedup.

Convergence and Feasibility

We terminate Algorithm 5.1 when the primal and dual residuals of (P_2)

$$r_{\text{prim}} := \|Au - \mathbf{1}\|_\infty,$$
$$r_{\text{dual}} := \left\|Px - c_u + A^T y + w\right\|_\infty$$

become smaller than some acceptable tolerance. Algorithm 5.1 is guaranteed to converge to the solution of (P_2) whenever (P_2), or equivalently (P), is feasible.

We next establish conditions that characterize the feasibility of (P):

Lemma 5.3.3. *Problem* (P) *is feasible if and only if there exists a set of indices* $\mathcal{E} = \{(i_1, j_1) \ldots, (i_n, i_n)\}$ *corresponding to exactly one nonzero element from each row and column of* C.

Proof. Regarding the "if" part, the $n \times n$ matrix

$$X_{(i,j)} = \begin{cases} 1 & \text{if} \quad i, j \in I \\ 0 & \text{otherwise} \end{cases} \tag{5.17}$$

is a feasible point for (P). Regarding the "only if" part, since (P) has a feasible point X, then according to [96, Theorem 1] there exists a set of indices $\mathcal{E} = \{(i_1, j_1) \ldots, (i_n, i_n)\}$ corresponding to exactly one nonzero element from each row and column of X. Due to the second constraint of (P), the same argument also holds for C. □

Note that Lemma 5.3.3 and [96, Theorem 1] imply that (P) is feasible whenever the matrix balancing problem is feasible, i.e. when the matrix C has total support [92].

5.3.1 Special cases and generalizations

The case where C is almost or fully dense: In the special case where C is fully dense we have $S = \mathbf{1}_n \mathbf{1}_n^T$, and Theorem 5.3.2 gives

$$\rho A(P + \sigma I_{\tilde{n}_{\text{nz}}})^{-1} A^T + I_n = \alpha I + \beta \mathbf{1}_{n \times n}$$

where

$$\alpha := \frac{\sigma\rho}{(2+\sigma/2)(2+\sigma)} + \frac{n\rho}{2+\sigma} + 1 \quad \text{and} \quad \beta := \frac{\rho}{2+\sigma}.$$

Using the Sherman-Morrison formula, we can calculate the inverse of (5.15) explicitly as

$$(\rho A(P + \sigma I_{\bar{n}_{\text{nz}}})^{-1}A^T + I_n)^{-1} = \frac{1}{\alpha}\left(I - \frac{\beta \mathbf{1}_n \mathbf{1}_n^T}{\alpha + \beta n}\right). \tag{5.18}$$

We can then solve (5.14) and perform ADMM on (P_2) without the need to perform an initial matrix factorization.

This approach can also be extended to cases where C has a relatively small number of zero elements. Indeed, by avoiding the elimination of the zero variables of (P) we get the following variant of (P_2):

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|p \odot x_u - c_u\|^2 \\ \text{subject to} & x_u \geq 0 \\ & x_{u_{(i)}} = 0 \; \forall \; i \text{ with } c_{u_{(i)}} = 0 \\ & Ax_u = \mathbf{1}_{2n} \end{array} \tag{P_3}$$

where p, A, x_u, and c_u are defined according to Section 5.2, but with H_u an $\frac{(n+1)n}{2} \times n^2$ linear map that extracts *all* the upper triangular elements of a vectorized $n \times n$ matrix. (P_3) can then be solved with Algorithm 5.1, with the following two changes. First, we replace Π_+ (line 2, Algorithm 5.1) with $\Pi_{\mathcal{S}}$, where $\mathcal{S} := \{x \in \mathbb{R}^{(n^2+n)/2} \mid x \geq 0 \text{ and } x_{(i)} = 0 \text{ for all } i \text{ such that } c_{(i)} = 0\}$. Secondly, the solution of the linear system of line 3 of Algorithm 5.1 is trivially solved using Fact 5.3.1 and (5.18).

Solving variants of (P) **with Algorithm 5.1:** Algorithm 5.1 can easily be adjusted to the case where $\|\cdot\|$ in the objective of (P) is a weighted Frobenius norm, i.e. $\|X\| = \|W \odot X\|_F$ where W is a given symmetric matrix. The only thing that has to change is the definition of p, and thus of $P := \text{diag}(p \odot p)$, to:

$$p := H_u \,\text{vec}\left(\begin{bmatrix} 2 & \sqrt{2} & \cdots & \sqrt{2} \\ & \ddots & & \vdots \\ & & 2 & \sqrt{2} \\ 0 & & & 2 \end{bmatrix} \odot W\right).$$

Theorem 3.2 could then be used to solve the linear system of algorithm 5.1 (line 3) efficiently. Similarly, we can allow for general constraints $X\mathbf{1} = r$ and $X^T\mathbf{1} = r$

in (P), where r is a given nonnegative vector, by simply changing $\Pi_{\mathbf{1}}$ in line 5 of Algorithm 5.1 to $\Pi_r(x) := r$. Finally, non-square or non-symmetric matrices C can be solved via use of (P) for the symmetric matrix $\begin{bmatrix} 0 & C \\ C^T & 0 \end{bmatrix}$.

5.4 Numerical Experiments

In this section we present numerical results of Algorithm 5.1 on a range of matrix normalization problems. We provide a `Julia` implementation of the Algorithm, along with code that generates all the results of this section at:

`https://github.com/oxfordcontrol/DoublyStochastic.jl`

5.4.1 Normalizing Hi-C Contact Matrices of the Human Genome in the 3D Space

We first present results on the application of our method to real-world contact matrices describing the 3D structure of the human genome, starting with a description of the nature of these matrices. The human genome has an end-to-end length on the order of meters when unfolded, but fits inside the cell nucleus with dimensions on the order of micrometers, implying that the genome is heavily folded in the 3D space. The 3D structure of the genome can be examined by breaking the genome into a number of pieces and measuring how many contacts exist between each piece in the 3D space [90]. This produces *Hi-C contact matrices*, where the term *Hi-C* describes the particular experimental procedure used.

The process is, however, subject to errors and experimental constraints. To alleviate these issues, the contact matrix is normalized[2] so that all its rows and columns sum to the same value. The matrix balancing approach is often the method of choice for this task [90, Supplemental Material II.b], but other methods have also been suggested in the literature [97].

[2]The presence of experimental errors might affect the sparsity pattern of the contact matrix which might put in question the second constraint of (P). Nevertheless, we still believe that the presence of this sparsity constraint is reasonable, as standard methods [90, Supplemental Material II.b] also produce matrices that satisfy it.

We will show that our approach can also be used for this task even for contact matrices containing hundreds of millions of nonzero entries as in [90]. In particular, we consider normalization of the contact matrix of the 7^{th} chromosome of the GM12878 cell[3], thus replicating [90, Figure 1.C]. Since the genome consists of sequences of the bases *adenine, guanine, cytosine* and *thymine*, it is typical to measure the length of each genome piece by the average number of bases it contains. The total range of the contact matrices is 0 to 160 mega-bases (Mb). We consider two discretization lengths, 1 kilo-base (Kb), and 5Kb, which result in contact matrices of 151 and 82 million nonzeros respectively. Figure 5.1 provides detailed views of the contact matrices, spanning the range [137.2–137.8Mb] for the contact map at 5Kb resolution and the [137.55–137.75Mb] for the one at 1Kb resolution. These regions were chosen to highlight interesting regions of the contact matrix as they appear in [90, Figure 1.C, rightmost column, two bottom subfigures].

Our approach produces considerably different normalized contact matrices than the matrix balancing approach. In particular, our approach results in contact matrices that have increased sparsity and higher contrast. This is unlike the matrix balancing approach which results in a normalized matrix that has exactly the same nonzeros as the original matrix. Although further investigation is necessary for the evaluation of the suitability of the approach in Hi-C data, the results indicate that our method can be used to normalize very large Hi-C datasets leading to promising visual results.

5.4.2 Spectral clustering problems

Next, we present results of running our algorithm on correlation matrices arising from spectral clustering [87]. In spectral clustering one is given a set of points $\{x_{(i)} \in \mathbb{R}^d\}$ to be arranged into l clusters. To this end, an *affinity matrix* C is generated with each entry $C_{(i,j)}$ representing a measure of the pairwise similarity

[3]The data corresponding to the thresholding criterion MAPQ ≥ 30 were used [90, Supplemental Material IIa.4]. Obtained from the GM12878 "combined" intrachromosomal tarball at `ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63525`

between points i and j. This matrix is then normalized and used by later stages

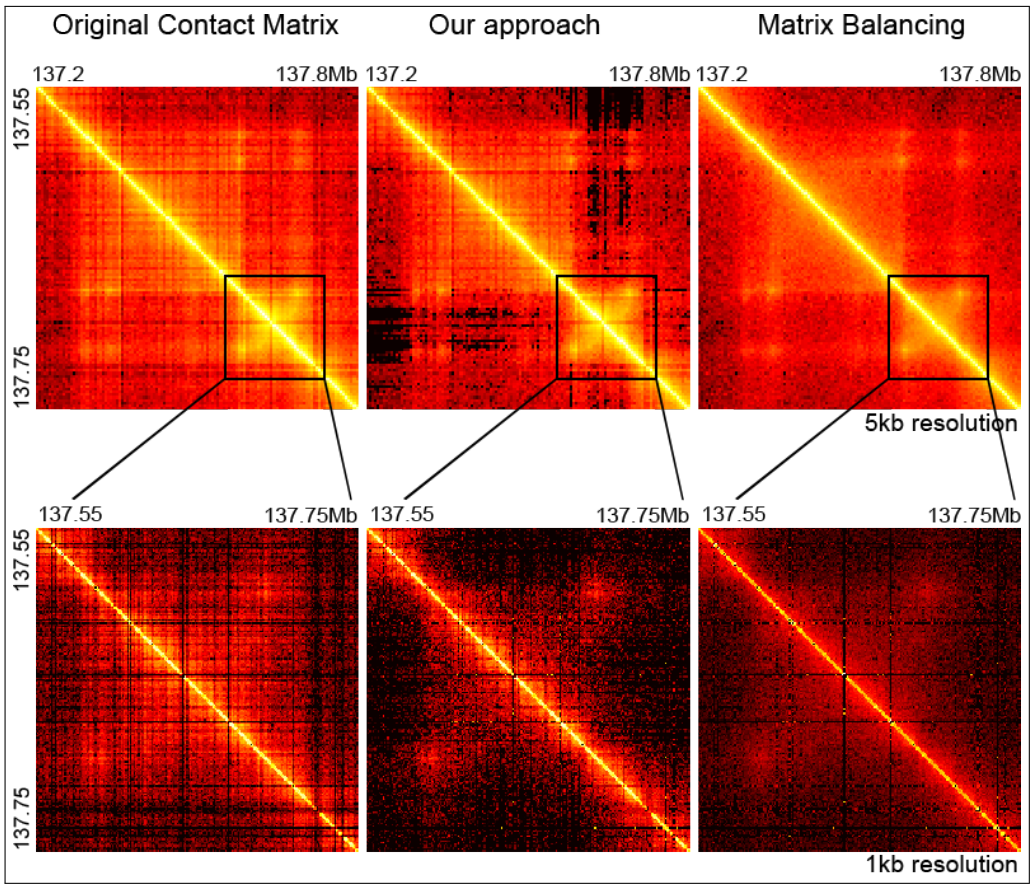


Figure 5.1: Details of Hi-C Contact Matrices for the 7$^{\text{th}}$ chromosome of the GM12878 cell, corresponding to [90, Figure 1.C, rightmost column, two bottom subfigures]. Top row shows the area $[137.2 - 137.8\text{Mb}]^2$ for the contact matrix of 5Kb resolution. Bottom row shows the area $[137.55 - 137.75\text{Mb}]^2$ for the contact matrix of 1Kb resolution. Areas representing zero contacts are depicted in black, while areas with a high number of contacts are shown in yellow. The total area of the contact matrices is $[0 - 160\text{Mb}]^2$, thus the areas depicted are zoomed 71 and 640 thousand times in the top and bottom figures respectively. The leftmost column shows the original contact matrices. [90] normalize the $n \times n$ contact matrix C via the matrix balancing method of [92] so that its columns and rows sum to $\sum_{i,j} C_{(i,j)}/n$. The resulting normalized matrices are depicted in the rightmost column. The middle column depicts the results of Algorithm 5.1 for normalizing the matrices so that its columns and rows sum to $\sum_{i,j} C_{(i,j)}/n$. The Conjugate Gradient method is used to solve the linear system (5.15) since using a Cholesky factorization resulted in memory issues. A tolerance of 10^{-3} is used for the termination of our Algorithm. The normalization of the 5Kb and 1Kb contact matrix take 701 and 3136 seconds respectively on a single-threaded implementation on Intel Gold 5120, 192GB memory.

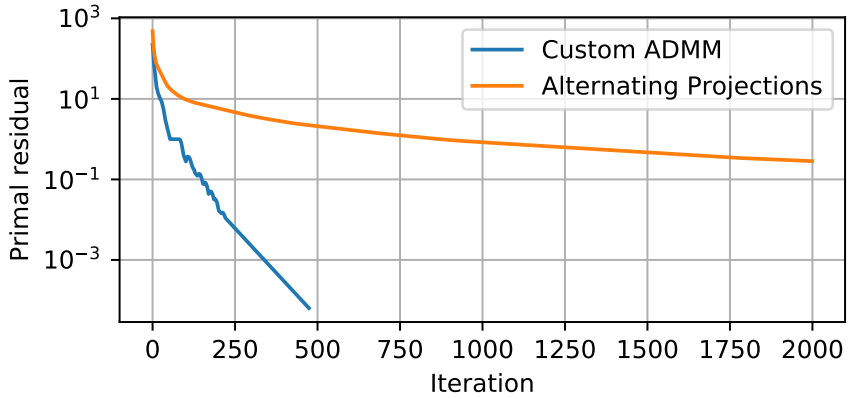


Figure 5.2: Comparison of the primal convergence (i.e., feasibility) of Algorithm 5.1 vs. the approach of [87] (5.20) on the Spambase dataset with (5.21) as affinity criterion with $\sigma = 100$.

of the clustering procedure. [87] suggested that the normalization

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|X - C\|^2 \\ \text{subject to} & X \text{ nonnegative} \\ & X\mathbf{1}_n = \mathbf{1}_n,\ X^T\mathbf{1}_n = \mathbf{1}_n, \end{array} \tag{5.19}$$

leads to superior clustering performance (in terms of clustering error rate) in various different test cases. Note that solving (5.19) is equivalent to solving (P) for $C + \epsilon_u \mathbf{1}_{n\times n}$. Note that the formulation (5.19) of [87] does not exploit sparsity in C. Nevertheless, [87] suggested that the following iterative scheme can be used to solve (5.19)

$$\begin{aligned} \tilde{X}^k &= X^k + n^{-2}(\mathbf{1}_n^T X^k \mathbf{1}_n + n)\mathbf{1}_{n\times n} \\ &\quad - n^{-1}(X^k \mathbf{1}_{n\times n} + \mathbf{1}_{n\times n} X^k) && \text{(5.20a)} \\ X^{k+1} &= \Pi_+\left(\tilde{X}^k\right). && \text{(5.20b)} \end{aligned}$$

The first step in (5.20) minimizes the objective of (P) subject to the equality constraints, while the second projects the iterate to the nonnegative orthant. However, this approach is not guaranteed to solve (P) to optimality. For example, in the simple case of $C = \frac{1}{10}\begin{bmatrix} 1 & 9 & 9 \\ 9 & 1 & 0 \\ 9 & 0 & 9 \end{bmatrix}$, (5.20) converges to a suboptimal point $\bar{X}$ with $\left\|\bar{X} - X^*\right\|_\infty = 0.0\overline{7}$ where X^* is the optimizer. Nevertheless, it appears that, in general, (5.20) converges to a feasible point. However, even convergence to a feasible point can be much slower than our approach, as demonstrated in Figure 5.2.

Table 5.1: Normalizing correlation matrices with Algorithm 5.1 for spectral clustering on Spambase. A tolerance of 10^{-4} is used for the termination of our Algorithm. The Timings are expressed in seconds. and compared against Gurobi with its default options (on (P_1)) and against solving $C+\epsilon_u \mathbf{1}_{n\times n}$ with the approach of §5.3.1 where and is $C(\sigma)$ the original affinity matrix. Hardware used: Intel i7-5557U CPU @ 3.10GHz, 8GB Memory.

σ	1.0	5.0	10.0	20.0
n_{nz}	3.9×10^4	1.8×10^6	4.0×10^6	7.3×10^6
t_{admm}	1.1×10^{-1}	5.7	1.5×10^1	2.6×10^1
t_{gurobi}	3.9×10^{-1}	4.6×10^1	1.1×10^2	2.0×10^2
$t_{\text{admm}}^{\text{dense}}$	7.7×10^2	6.9×10^2	6.4×10^2	7.2×10^2

Besides guaranteed convergence to the optimizer of (P), our approach can also handle sparsity in the affinity matrices. To demonstrate how exploiting sparsity can lead to significant speedups we consider the Spambase dataset[4] (considered in [87]) with an RBF kernel as an affinity criterion

$$C_{(i,j)}(\sigma) = e^{-\left\|x_{(i)}-x_{(j)}\right\|^2/\sigma^2}, \tag{5.21}$$

where σ is a parameter that is typically tuned to achieve the best clustering performance. Note that due to the exponential form of $C(\sigma)$, some values will be very small. Therefore, we truncate to zero all entries with value less than 10^{-7}. The runtimes of applying our algorithm to this dataset are listed in Table 5.1 and compared to timings achieved with Gurobi with its default options (note that we use (P_1) for Gurobi as we consider (P_1) to be a "standard" reformulation of (P)). We observe that exploiting sparsity can lead to significant speedup as compared to treating the affinity matrix as fully dense, even if we follow the approach of §5.3.1. At the same time, the optimizers of (P) for the affinity matrices $C(\sigma)$ considered in Table 5.1 appear to coincide with the ones for the fully dense $C(\sigma) + \epsilon_{\text{p}} \mathbf{1}_{n\times n}$.

5.4.3 Matrices from the SuiteSparse Collection

Finally, we consider all Undirected Weighted Graph Matrices, with less than 50 million nonzeros, contained in the SuiteSparse collection[5]. 69 matrices meet these

[4] `archive.ics.uci.edu/ml/datasets/spambase`
[5] Available at `sparse.tamu.edu`

criteria. We use Algorithm 5.1 to normalize every matrix C so that all of its columns and rows sum to $\max_{i,j}(C_{(i,j)})$. In order to ensure feasibility we add $\epsilon_p I$ to every matrix. All of the problems, except two, have nonnegative entries. For these two exceptions (`nopoly.mat` and `Stranke94.mat`), we change the negative entries to their absolute value, as we are unaware of practical cases where C has to be normalized to a doubly stochastic matrix and is expected to have negative entries.

Detailed comparison of our results with Gurobi presented in the Supplementary Material. Figure 5.3 shows the timings achieved by our method, and the speedups relative to Gurobi (on (P_1)), as a function of each problem's nonzeros.

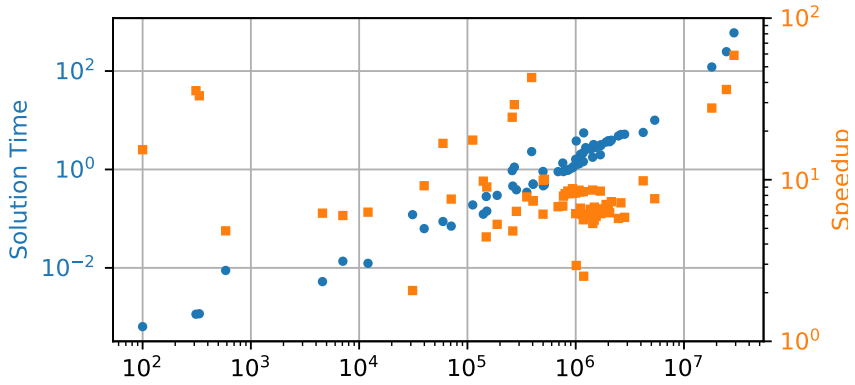


Figure 5.3: Results of Algorithm 5.1 for matrices from the SuiteSparse collection. Dots represent the timing of our Algorithm (with 10^{-4} tolerance), while squares represent the speedup achieved over Gurobi with its default options (on (P_1)). Hardware used: a single thread running on an Intel Gold 5120 with 192GB of memory.

5.A Proof of Theorem 5.3.2

In this section we will prove Theorem 5.3.2, i.e., we will show that

$$A\,\mathrm{diag}(H_u\,\mathrm{vec}(D))A^T = S \odot (D + D^T) + \mathrm{diag}(S \odot (D + D^T)\mathbf{1})$$

where S (S_u) is an integer $(0-1)$ matrix representing the sparsity pattern of C (C_u) and D is an upper triangular $n \times n$ matrix.

Using the fact that $H_u H_u^T = I$, the definition of A and defining $d := H_u\,\mathrm{vec}\,D$ we

get

$$\begin{aligned} A\operatorname{diag}(d)A^T &= A\operatorname{diag}(\sqrt{d})H_uH_u^T\operatorname{diag}(\sqrt{d})A^T \\ &= \Big(A\operatorname{diag}(\sqrt{d})\operatorname{diag}(H_u)\Big)\Big(A\operatorname{diag}(\sqrt{d})\operatorname{diag}(H_u)\Big)^T \end{aligned}$$

where $\sqrt{d}$ denotes the element-wise square root of d. Recalling that $A = A_1 + A_2$ we have

$$A\operatorname{diag}(\sqrt{d})\operatorname{diag}(H_u) = \underbrace{A_1\operatorname{diag}(\sqrt{d})\operatorname{diag}(H_u)}_{:=B_1} + \underbrace{A_2\operatorname{diag}(\sqrt{d})\operatorname{diag}(H_u)}_{:=B_2}$$

thus

$$A\operatorname{diag}(d)A^T = \begin{bmatrix} I_n & I_n \end{bmatrix}\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}^T\begin{bmatrix} I_n \\ I_n \end{bmatrix}$$

We now focus on the first part of the above symmetric product, To this end, define $\sqrt{s}_i$ the i^{th} column of $S_u \odot \sqrt{D}$ where $\sqrt{D}$ denotes the element-wise square root of D. Using the fact that $H_u^T\operatorname{diag}(x)H_u = \operatorname{diag}(\operatorname{vec}(S_u)\odot(H_u^Tx))$ for any vector x of appropriate dimensions we get

$$\begin{aligned} H_u^T\operatorname{diag}(\sqrt{d})H_u &= \operatorname{diag}(\operatorname{vec}(S_u)\odot(H_u^T\sqrt{d})) = \operatorname{diag}(\operatorname{vec}(S_u)\odot(H_u^TH\operatorname{vec}(\sqrt{D}))) \\ &= \operatorname{diag}(\operatorname{vec}(S_u\odot S_u\odot\sqrt{D})) = \operatorname{diag}(\operatorname{vec}(S_u\odot\sqrt{D})). \end{aligned} \tag{5.22}$$

Use the definitions of A_1, A_2 and (5.22) to get

$$\begin{aligned} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} \mathbf{1}^T\otimes I \\ I\otimes\mathbf{1}^T \end{bmatrix}H_u^T\operatorname{diag}(d)H_u &= \begin{bmatrix} I & \cdots & I \\ \mathbf{1}^T & & \\ & \ddots & \\ & & \mathbf{1}^T \end{bmatrix}\begin{bmatrix} \operatorname{diag}(\sqrt{s}_1) & & \\ & \ddots & \\ & & \operatorname{diag}(\sqrt{s}_n) \end{bmatrix} \\ &= \begin{bmatrix} \operatorname{diag}(\sqrt{s}_1) & \cdots & \operatorname{diag}(\sqrt{s}_n) \\ \sqrt{s}_1^T & & \\ & \ddots & \\ & & \sqrt{s}_n^T \end{bmatrix} \end{aligned}$$

We can then form the symmetric product of this matrix with itself to obtain

$$\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}^T = \begin{bmatrix} \operatorname{diag}(\sqrt{s}_1) & \cdots & \operatorname{diag}(\sqrt{s}_n) \\ \sqrt{s}_1^T & & \\ & \ddots & \\ & & \sqrt{s}_n^T \end{bmatrix}\begin{bmatrix} \operatorname{diag}(\sqrt{s}_1) & \sqrt{s}_1 & & \\ \vdots & & \ddots & \\ \operatorname{diag}(\sqrt{s}_n) & & & \sqrt{s}_n \end{bmatrix}$$

or, noting that $\sqrt{s_i}^T\sqrt{s_i} = \mathbf{1}^T s_i$, $\text{diag}^2(\sqrt{s_i}) = \text{diag}(s_i)$, $\text{diag}(\sqrt{s_i})\sqrt{s_i} = s_i$ where s_i denotes the $i-$th column of $S_u \odot \text{mat}(d)$,

$$\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}^T = \begin{bmatrix} \text{diag}(\sum_i s_i) & s_1 & \dots & s_n \\ s_1^T & \mathbf{1}^T s_1 & & \\ \vdots & & \ddots & \\ s_n^T & & & \mathbf{1}^T s_n \end{bmatrix} = \begin{bmatrix} \text{diag}((S_u \odot D)\mathbf{1}) & (S_u \odot D) \\ (S_u \odot D)^T & \text{diag}((S_u \odot D)^T\mathbf{1}) \end{bmatrix}.$$

Thus

$$\begin{aligned} A\,\text{diag}(d)A^T &= \begin{bmatrix} I & I \end{bmatrix}\begin{bmatrix} B_1 \\ B_2 \end{bmatrix}\begin{bmatrix} B_1^T B_2^T \end{bmatrix}\begin{bmatrix} I \\ I \end{bmatrix} \\ &= \text{diag}((S_u \odot D)\mathbf{1}) + (S_u \odot D) + (S_u \odot D)^T + \text{diag}((S_u \odot D)^T\mathbf{1}) \end{aligned}$$

or, recalling that D is upper triangular,

$$A\,\text{diag}(H_u\,\text{vec}(D))A^T = S \odot (D + D^T) + \text{diag}(S \odot (D + D^T).$$

5.B Detailed Results for SuiteSparse Matrices

In this Section we provide detailed results for §5.4.3

Table 5.2: Results of Algorithm 1 for Undirected Weighted Graph Matrices from the SuiteSparse collection. A tolerance of 10^{-4} is used for the termination of our Algorithm. The timings are in seconds and they are compared against the solution times of Gurobi (on P1). Hardware used: a single thread running on an Intel Gold 5120 with 192GB of memory.

Problem Name	n	n_{nz}	t_{admm}	t_{gurobi}
GD97_b	47	311	1.14×10^{-3}	4.07×10^{-2}
Journals	124	12 068	1.24×10^{-2}	7.80×10^{-2}
MISKnowledgeMap	2427	59 449	8.75×10^{-2}	1.47
Sandi_authors	86	334	1.17×10^{-3}	3.87×10^{-2}
Stranke94	10	100	6.40×10^{-4}	9.81×10^{-3}
USAir97	332	4584	5.26×10^{-3}	3.27×10^{-2}
ak2010	45 292	262 390	4.60×10^{-1}	2.22
al2010	252 266	1 482 748	2.87	1.61×10^{1}
ar2010	186 211	1 090 521	1.39	1.17×10^{1}
astro-ph	16 706	259 208	9.47×10^{-1}	2.31×10^{1}
az2010	241 666	1 437 760	2.71	1.45×10^{1}
ca2010	710 145	4 199 511	5.65	5.57×10^{1}

Table 5.2: Continued.

Problem Name	n	n_{nz}	t_{admm}	t_{gurobi}
co2010	201 062	1 175 636	2.19	1.24×10^{1}
cond-mat-2003	31 163	271 221	1.12	3.26×10^{1}
cond-mat-2005	40 421	391 803	2.31	9.90×10^{1}
cond-mat	16 726	111 914	1.90×10^{-1}	3.35
ct2010	67 578	403 930	5.07×10^{-1}	3.76
de2010	24 115	140 171	1.23×10^{-1}	1.21
fl2010	484 481	2 830 775	5.20	3.04×10^{1}
ga2010	291 086	1 709 142	3.18	2.10×10^{1}
geom	7343	31 139	1.21×10^{-1}	2.49×10^{-1}
hep-th	8361	39 863	6.28×10^{-2}	5.76×10^{-1}
hi2010	25 016	149 142	2.82×10^{-1}	1.25
human_gene1	22 283	24 669 643	2.47×10^{2}	8.94×10^{3}
human_gene2	14 340	18 068 388	1.21×10^{2}	3.36×10^{3}
ia2010	216 007	1 237 177	2.78	1.69×10^{1}
id2010	149 842	878 106	1.01	8.30
il2010	451 554	2 616 018	5.14	3.70×10^{1}
in2010	267 071	1 548 787	3.00	1.92×10^{1}
ks2010	238 600	1 360 398	2.58	1.69×10^{1}
ky2010	161 672	949 450	1.14	9.44
la2010	204 447	1 185 081	5.54	1.40×10^{1}
lesmis	77	585	8.87×10^{-3}	4.28×10^{-2}
ma2010	157 508	934 118	1.08	9.49
md2010	145 247	845 625	9.59×10^{-1}	8.17
me2010	69 518	404 994	4.99×10^{-1}	3.69
mi2010	329 885	1 907 975	3.60	2.53×10^{1}
mn2010	259 777	1 486 879	2.89	1.95×10^{1}
mo2010	343 565	2 000 133	3.84	2.57×10^{1}
mouse_gene	45 101	28 967 291	5.97×10^{2}	3.52×10^{4}
ms2010	171 778	1 011 758	3.78	1.12×10^{1}
mt2010	132 288	770 956	9.05×10^{-1}	7.16
nc2010	288 987	1 705 607	3.05	1.88×10^{1}
nd2010	133 769	759 715	1.35	9.30
ne2010	193 352	1 107 206	2.03	1.35×10^{1}
netscience	1589	7073	1.36×10^{-2}	8.19×10^{-2}
nh2010	48 837	283 387	3.87×10^{-1}	2.46
nj2010	169 588	999 500	1.23	1.02×10^{1}
nm2010	168 609	999 579	1.61	9.92
nopoly	10 774	70 842	7.05×10^{-2}	5.35×10^{-1}
nv2010	84 538	501 536	4.66×10^{-1}	4.61
ny2010	350 169	2 059 713	3.66	2.29×10^{1}
oh2010	365 344	2 133 584	3.93	2.88×10^{1}
ok2010	269 118	1 543 266	2.83	1.68×10^{1}

Table 5.2: Continued.

Problem Name	n	n_{nz}	t_{admm}	t_{gurobi}
or2010	196 621	1 176 133	1.44	1.22×10^{1}
pa2010	421 545	2 480 007	4.80	2.76×10^{1}
rgg_n_2_15_s0	32 768	353 248	3.44×10^{-1}	2.69
ri2010	25 181	150 931	1.42×10^{-1}	1.29
sc2010	181 908	1 075 068	1.30	1.12×10^{1}
sd2010	88 360	499 082	9.10×10^{-1}	5.57
tn2010	240 116	1 434 082	1.77	1.53×10^{1}
tx2010	914 231	5 370 503	1.00×10^{1}	7.65×10^{1}
ut2010	115 406	687 472	9.04×10^{-1}	6.15
va2010	285 762	1 687 890	1.99	1.69×10^{1}
vt2010	32 580	188 178	2.98×10^{-1}	1.58
wa2010	195 574	1 143 006	2.05	1.25×10^{1}
wi2010	253 096	1 462 500	3.21	1.98×10^{1}
wv2010	135 218	798 140	9.75×10^{-1}	7.99
wy2010	86 204	513 790	4.89×10^{-1}	4.93

6

An active-set algorithm for norm constrained quadratic problems

6.1 Introduction

In the final chapter of this thesis we deviate from convexity and consider a particular class of nonconvex programming that includes a (upper and/or lower bounding) norm constraint. Optimization with spherical constraints $\|x\|_2 = 1$ or norm constraints $r_{\min} \leq \|x\|_2 \leq r_{\max}$ is used in a number of scientific fields. Many optimization problems, such as eigenvalue problems [54], dimensionality reduction [98] and compressed sensing [99], are naturally posed on or in the unit sphere, while norm constraints are often useful for regularization e.g. in general nonlinear [100] or robust optimization [101]. In this chapter, we consider the solution of norm constrained problems in the form

$$\begin{array}{ll} \text{minimize} & f(x) := \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & r_{\min} \leq \|x\|_2 \leq r_{\max} \\ & Ax \leq b, \end{array} \tag{P}$$

where $x \in \mathbb{R}^n$ is the decision variable, $P \in \mathbb{S}^n$, i.e., an $n \times n$ symmetric matrix, $A = [a_1^T \dots a_m^T] \in \mathbb{R}^{m \times n}$, $q \in \mathbb{R}^n$, $b = [b_1 \dots b_m]^T \in \mathbb{R}^m$ and $r_{\min}, r_{\max}$ nonnegative scalars.

It is difficult, in general, to solve problems of the form (P) due to the nonconvexity of the lower bounding norm constraint and the potential indefiniteness of the matrix

P, which renders many, but certainly not all [101], of these problems intractable. Even finding a feasible point for (P) or testing if a first-order critical point of (P) is a local minimizer can be intractable (see Proposition 6.3.2 and [102]). As a result, we restrict our attention to the search for first-order critical points and we assume that a feasible point is given or can be computed.

A specific tractable variation of (P) is the famous Trust-Region Subproblem[1]:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Px + q^T x \\ \text{subject to} & \|x\|_2 = r \\ & Ax = b. \end{array} \tag{TRS}$$

Perhaps surprisingly the TRS can be solved to global optimality despite its nonconvexity. Global solution of the TRS has been widely studied in the literature. Specific approaches to its solution include exact Semidefinite Reformulations, Gradient Steps, Truncated Conjugate Gradient Steps, Truncated Lanczos Steps and Newton root-finding, with associated software packages for its solution. The reader can find details in [100] and [2, Chapter 4].

We suggest an active-set algorithm for the minimization of (P) that, as we proceed to describe, takes advantage of the existing efficient global solution methods for the TRS. Using such an approach, (P) is optimized by solving a series of equality constrained subproblems. Each subproblem is a modification of (P) where a subset of its inequalities (called the *working-set*) is replaced by equalities while the rest are ignored. If a subproblem does not include a norm constraint then it is called an *Equality-constrained Quadratic Problem* (EQP); otherwise it is a *Trust-Region Subproblem* (TRS). As described earlier the global solution for the TRS, and also for the EQP [103], has been widely studied in the literature. Thus we can efficiently address the global solution of each subproblem. However, a complication exists for the TRS because, unlike EQP, it can posess *local-nonglobal* minimizers i.e., local minimizers that are not globally optimal, which complicate the analysis. Indeed, if the optimal working-set includes a norm constraint and the global solutions of the

[1]Note that the TRS is typically defined in the ball $\|x\| \leq r$ rather than the sphere $\|x\| = r$. We consider only the boundary solution as this fits better the needs of this chapter.

respective TRS subproblem are infeasible for (P), then the solution of (P) must be obtained from a local-nonglobal optimizer of the TRS subproblem.

Thus, an algorithm for computing local-nonglobal minimizers of the TRS is required for our active-set algorithm. Unfortunately, algorithms for detecting the presence of/computing local-nonglobal solutions of (TRS) are significantly less mature than their global counterparts. A notable exception is the work of Martinez [104], which proved that there exists at most one local-nonglobal minimizer and gave conditions for its existence. Moreover, [104] presented a root-finding algorithm for the computation of the Lagrange Multiplier associated with the local-nonglobal minimizer that entailed the computation of the two smallest eigenvalues of the Hessian and the solution of a series of indefinite linear systems.

More recent work is based on a result from [105] which shows that each KKT point of the TRS, in the absence of linear equality constraints $Ax = b$ (i.e., with a norm constraint on x only), can be extracted from an eigenpair of

$$M := \begin{bmatrix} -P & qq^T/r^2 \\ I & -P \end{bmatrix}.$$

This result was used in [106] to calculate the local-nonglobal minimizer under the assumption that it exists. We extend these results by proving that both checking if the local-nonglobal minimizer exists and computing it, in the case where it exists, can be performed at the same time by simply calculating the two rightmost eigenpairs of M. Crucially, and similarly to the results of [105] for the global minimizer, this allows the use of efficient factorization-free methods such as the Arnoldi method for the detection and computation of the local-nonglobal minimizer. Furthermore, we show that this approach can efficiently handle equality constraints $Ax = b$ in the Trust-Region Subproblem, without the need of variable reduction, by means of a projected Arnoldi method.

The chapter proceeds as follows. In Section 6.2.1 we present an introduction to the TRS and present novel results for the detection/computation of its local-nonglobal minimizer and for the incorporation of linear equality constraints. In Section 6.3.1 we introduce an active-set algorithm for solving (P) starting from the

special case where $r_{\min} = r_{\max}$, and then generalizing to any $r_{\min}, r_{\max}$. Finally, in Section 6.4, we conclude with a series of numerical results.

6.2 The Trust-Region Subproblem

This section concerns the computation of global as well as local-nonglobal minimizers for (TRS). Although this section only considers solutions on the boundary $\|x\| = r$, the results can be trivially extended to the interior $\|x\| \leq r$. Indeed, if an interior solution exists then the TRS is essentially convex (P is positive semidefinite in the nullspace of A [107, §1]), thus no local-nonglobal solution exists and the interior solution is obtained by solving an equality constrained quadratic problem. The presence of a (necessarily global) interior solution can be detected by checking the sign of the Lagrange Multiplier of the global boundary solution(s) [104, Lemma 2.2].

For clarity of exposition, we will first assume there are no linear equality constraints in (TRS), resulting in the following simplified problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & \|x\|_2 = r, \end{array} \tag{T}$$

where $x \in \mathbb{R}^n$ is the decision variable, and $P \in \mathbb{S}^n$, $q \in \mathbb{R}^n, r \in \mathbb{R}_+$ are the problem data. For the rest of the section we will assume that $n > 1$, excluding the trivial one dimensional case where the feasible set consists of two points. We will then show in §6.2.1 how to extend the results of this section to allow for the inclusion of linear equality constraints. Finally, in §6.2.A we prove the main result of this section which is presented in a separate subsection to facilitate the presentation.

In the sequel we will make frequent use of the eigendecomposition of $P := W\Lambda W^T$ defined by an orthonormal matrix $W := [w_1 \dots w_n]$ and $\Lambda := \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ where $\lambda_1 \leq \cdots \leq \lambda_n$.

Every KKT point of (T) satisfies

$$Px + q + \mu x = 0 \Rightarrow x = -(P + \mu I)^{-1} q, \tag{6.1}$$

where μ is a Lagrange multiplier associated with the norm constraint[2]. Equation (6.1) is well defined when $\mu \neq -\lambda_i,\ i = 1, \ldots, n$. For the purposes of clarity of the subsequent introduction, which follows [2, §4], we will assume that this holds. However, the conclusions of this section are independent of this assumption.

Using the feasibility of x

$$\|x\|_2^2 = r^2 \Leftrightarrow \left\|(P + \mu I)^{-1} q\right\|_2^2 - r^2 = 0 \tag{6.2}$$

and noting that

$$\begin{aligned}\left\|(P + \mu I)^{-1} q\right\|_2^2 &= \left\|(W(\Lambda + \mu I)W^T)^{-1} q\right\|_2^2 = \left\|W(\Lambda + \mu I)^{-1} W^T q\right\|_2^2 \\ &= \left\|(\Lambda + \mu I)^{-1} W^T q\right\|_2^2 = \sum_{i=1}^{n} \frac{(w_i^T q)^2}{(\lambda_i + \mu)^2}\end{aligned}$$

we can express the rightmost condition in (6.2) as

$$s(\mu) := \sum_{i=1}^{n} \frac{(w_i^T q)^2}{(\lambda_i + \mu)^2} - r^2 = 0. \tag{6.3}$$

Determining the KKT points of (T) is therefore equivalent to finding the roots of s, which is often referred to as the *secular equation* [2]. We depict s for a particular choice of P, q, r in Figure 6.1. As might be expected from the tight connection between polynomial root-finding problems and eigenproblems, solving $s(\mu) = 0$ is equivalent to the following eigenproblem [105, Equation (22)]:

$$\underbrace{\begin{bmatrix} -P & qq^T/r^2 \\ I & -P \end{bmatrix}}_{:=M} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \mu \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}. \tag{6.4}$$

This elegant result was first noted more than 50 years ago [108, Equation (2.21)], only to be later disregarded as inefficient by some of the same authors [109], and then rediscovered by [105] who highlighted its great applicability owing to the remarkable efficacy of modern eigensolvers [57].

The relation between the spectrum of M and the Lagrange multipliers of (T) is formally stated below:

[2]Technically, μ is the Lagrange multiplier of the equivalent constraint $\frac{1}{2}\|x\|_2^2 = \frac{1}{2}r^2$. We define (T) with $\|x\|_2 = r$ for simplicity and to match the notation of [105].

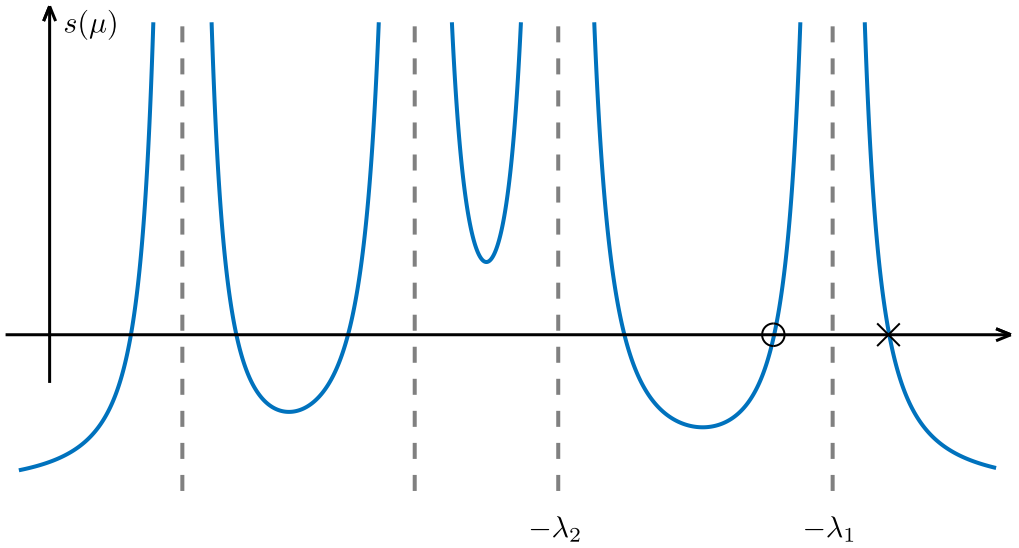


Figure 6.1: A typical example of the secular equation $s(\mu)$. The dashed vertical lines mark the locations of the eigenvalues of $-P$, i.e., $-\lambda_1, \ldots, -\lambda_n$. The Lagrange multiplier of the global solution of (T) is the rightmost root of $s(\mu)$ (marked by a cross). The TRS associated with this Figure also exhibits a local-nonglobal minimizer (i.e., in addition to the global minimizer marked with a cross) with a Lagrange multiplier that corresponds to the second-rightmost root of $s(\mu)$ (marked by a circle).

Lemma 6.2.1. *The spectrum of M includes the Lagrange multiplier of every KKT point of* (T).

Proof. This is a consequence of Proposition 6.2.9 of subsection 6.2.A and [108, Theorem 4.1]. □

Note that the eigenvalues of M are complex in general since M is asymmetric. Furthermore, Lemma 6.2.1 and all the subsequent results allow for potential "degenerate" KKT points with $\mu = -\lambda_i$.

We will focus on the KKT points that are minimizers of (T) rather than maximizers or saddle points. The characterization of global minimizers is widely known in the literature. The following theorem shows that the Lagrange multiplier μ^g of the global minimizer corresponds to the rightmost eigenvalue of M in $\mathbb{C}$, which is always real:

Theorem 6.2.2. *A KKT point of* (T) *with Lagrange multiplier μ^g is a global minimizer if and only if μ^g is the rightmost eigenvalue of M. Furthermore, we necessarily have that $\mu^g \in [-\lambda_1, \infty)$.*

Proof. See [105, Theorem 3.2]. □

In general, (T) is guaranteed to possess a unique global optimizer, except perhaps in a special case that has deservedly been given the following special name [2, §4.3]:

Definition(Hard case)**.** Problem (T) belongs to the *hard case* when $\mu^g = -\lambda_1$.

We will see that in the "hard case" (T) is still guaranteed to have a global minimizer, which might not necessarily be unique. Furthermore, the computation of the minimizer(s) can be more challenging than in the "standard" case.

Moreover, due to the nonconvexity of (T) there can exist a local minimizer that is not global. We derive an analogous result for this minimizer that is called "local-nonglobal" [104]:

Theorem 6.2.3. (T) *has a second-order sufficient local-nonglobal minimizer if and only if it does not belong to the hard case and the second-rightmost eigenvalue of* M *is real, simple and not in the spectrum of* $-P$*. Furthermore, if such a minimizer exists, then its Lagrange multiplier* μ^ℓ *is equal to this second-rightmost eigenvalue.*

Proof. See §6.2.A. □

The Lagrange multipliers for the global and local-nonglobal minimizers of (T) can therefore be identified by calculating the two rightmost eigenvalues[3] of M, which can be done efficiently e.g. with the Arnoldi method. Each of the respective multipliers corresponds to a minimizer that can be obtained by the respective eigenvectors of M as [105, Theorem 3.3]:

$$x^* = -\text{sign}(q^T z_2^*) r \frac{z_1^*}{\|z_1^*\|_2}, \tag{6.5}$$

unless $z_1^* = 0$, which we proceed to show happens iff (T) is in the hard case.

[3]Except perhaps when the local-nonglobal minimizer is not second-order sufficient.

Dealing with the hard-case: Extracting a solution x^* via (6.5) is not possible when $z_1^* = 0$, since (6.5) is not well-defined in that case. According to Proposition 6.2.9 of §6.2.A we have $z_1^* = 0$ only if μ^* is an eigenvalue of $-P$. Note that, due to [104], this can never happen for the local-nonglobal minimizer; thus we can always extract the local-nonglobal minimizer with (6.5). However, it is possible for global minimizers and is precisely the hard case, i.e., the case where $\mu^* = -\lambda_1$. In the hard case, the necessary and sufficient conditions for global optimality are, due to the KKT conditions of (T) and Theorem (6.2.2), the following:

$$(P - \lambda_1 I)x + q = 0 \tag{6.6}$$

$$\|x\|_2 = r. \tag{6.7}$$

Note that $P - \lambda_1 I$ is singular, and the above system of equations represents the intersection of an affine subspace with a sphere. This intersection must be non-empty, since (T) necessarily has a global minimizer as it arises from the minimization of a smooth function over a compact subset of $\mathbb{R}^n$. One can then solve (6.6)–(6.7), by computing the minimum length solution $y_{\min}$ of $(P - \lambda_1 I)x + q = 0$ and return $x^* = y_{\min} + \alpha v$ where α is a scalar such that $\|x^*\|_2 = \|y_{\min} + \alpha v\|_2 = r$ and v is any null-vector of $P - \lambda_1 I$. Interestingly, in the hard case the second part of the rightmost eigenvector of (6.4), z_2^*, is a null vector of $P - \lambda_1 I$ (Proposition 6.2.10)[4] and thus the only additional computation for extracting a hard case solution is finding the minimum-length solution of a symmetric linear system. This can be achieved e.g. with MINRES-QLP [110], or with the CG method [105, Theorem 4.3].

6.2.1 Equality-constrained Trust-Region Subproblems

We will now extend the results of this section thus far to also account for the presence of linear equality constraints, $Ax = b$, in the TRS. We will show that, given an operator that projects an $n-$dimensional vector to the nullspace of A, we can calculate global as well as local-nonlocal minimizers of (TRS) by applying a projected Arnoldi method to M.

[4]Except in a set of TRS instances of measure zero. In these cases we can compute an null vector of $P - \lambda_1$ with standard methods e.g. with a preconditioned conjugate gradient method.

For simplicity, and without loss of generality[5], we will assume that $b = 0$, thus solving the following problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & \|x\|_2 = r \\ & Ax = 0 \end{array} \tag{6.8}$$

where $x \in \mathbb{R}^n$ is the decision variable, $P \in \mathbb{S}^n$, i.e., an $n \times n$ symmetric matrix, A is a $m \times n$ matrix of full row rank and q is an $n-$dimensional vector. Similarly to the preceding results for (T), we assume that $n - m > 1$.

In principle, (6.8) can be solved with the "eigenproblem approach" described in this section if we reduce (6.8) into a smaller TRS subproblem via a nullspace elimination procedure [2, §15.3], obtaining

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}y^T \tilde{P} y + \tilde{q}^T y \\ \text{subject to} & \|y\|_2 = r \end{array} \tag{6.9}$$

where we define Z to be a $n \times (n-m)$ orthonormal matrix with $\mathcal{R}(Z) = \mathcal{N}(A)$ and

$$\tilde{P} := Z^T P Z, \quad \tilde{q} := Z^T q.$$

According to the preceding results of this section, global and local-nonglobal minimizer(s) y^* of (6.9) can be computed by calculating the two rightmost eigenvectors of

$$\tilde{M} := \begin{bmatrix} -\tilde{P} & \tilde{q}\tilde{q}^T/r^2 \\ I & -\tilde{P} \end{bmatrix}.$$

The respective solution(s) x^* of (P) can then be recovered as

$$x^* = Z y^*. \tag{6.10}$$

Disadvantages of this approach include that a nullspace basis is required and that the structure or sparsity of the problem might be destroyed. Thus, such an approach is impractical for sparse problems.

[5]Problems with linear constraints of the form $Ax = b$ can be transformed to (6.8) via a change of variables $\tilde{x} = x + x_0$ where $x_0 \perp \mathcal{N}(A)$ such that $Ax_0 = b$, resulting in a quadratic cost and constraints: $A\tilde{x} = 0$, $\|\tilde{x}\|_2 = \sqrt{r^2 - \|x_0\|_2^2}$.

We will present an alternative method that avoids these issues, mirroring standard approaches for solving equality constrained quadratic programs [103]. To this end, note that

$$\tilde{M} = \begin{bmatrix} -\tilde{P} & \tilde{q}\tilde{q}^T/r^2 \\ I & -\tilde{P} \end{bmatrix} = \begin{bmatrix} Z^T & 0 \\ 0 & Z^T \end{bmatrix} \begin{bmatrix} -P & qq^T/r^2 \\ I & -P \end{bmatrix} \begin{bmatrix} Z & 0 \\ 0 & Z \end{bmatrix}.$$

Since Z is an orthonormal matrix, every eigenvalue-eigenvector pair $\{\mu, [\tilde{z}_1;\ \tilde{z}_2]\}$ of $\tilde{M}$ corresponds to an eigenvalue-eigenvector pair $\{\mu, [z_1;\ z_2]\}$ of the matrix

$$\begin{bmatrix} ZZ^T & 0 \\ 0 & ZZ^T \end{bmatrix} \underbrace{\begin{bmatrix} -P & qq^T/r^2 \\ I & -P \end{bmatrix}}_{=M} \begin{bmatrix} ZZ^T & 0 \\ 0 & ZZ^T \end{bmatrix} \tag{6.11}$$

with $Z[\tilde{z}_1;\ \tilde{z}_2] = [z_1;\ z_2]$. Note that (6.11) also has an extra eigenvalue at zero with multiplicity $2(n-m)$.

Recalling that Z is an orthonormal matrix spanning $\mathcal{N}(A)$, we note that multiplication with ZZ^T is equivalent to projection onto the nullspace of A. Thus, in practice, eigenvalues of (6.11) can be calculated using the Arnoldi method on M where we project the starting vector and every requested matrix-vector product with M onto the nullspace of $\begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix}$.

Note that the two rightmost eigenvalues of $\tilde{M}$ correspond to the two rightmost eigenvalues of (6.11) if at least two eigenvalues of $\tilde{M}$ have positive real parts. We will show that we can always transform (6.8) so that this holds. Indeed if we shift P to $P - \alpha I$ with an appropriate $\alpha \in \mathbb{R}$ so that $\tilde{P}$ become negative definite[6], then $\tilde{M}$ has at least two eigenvalues with positive real part:

Lemma 6.2.4. *If $\tilde{P} \in \mathbb{S}^{n-m}$ is negative definite then $\tilde{M}$ has at least two eigenvalues with positive real part, unless we trivially have $n - m = 1$.*

Proof. To avoid redefining the notation used in the introduction of section we will show the equivalent result for (P, M), i.e., that M has at least two eigenvalues with positive real part when P is negative definite and $n > 1$.

[6]Such an α can be obtained by applying e.g., the Lanczos method, or the Gershgorin circle Theorem on P.

Since $M \in \mathbb{R}^{2n\times 2n}$ it has $2n > 4$ eigenvalues. Thus, it suffices to show that P has at most one eigenvalue with negative real part. To show this, first note that s has at most one (simple) real root with negative real part since

$$s'(a) = -2\sum_{j=1}^{n} \frac{(w_j^T q)^2}{(\lambda_j + \alpha)^3} \tag{6.12}$$

is negative for any $\alpha < 0$ since $\lambda_1, \ldots, \lambda_n < 0$ by assumption. To conclude the proof, it suffices to show that all the roots of s with negative real part are real. Indeed, note that for any $a, b \in \mathbb{R}$ we have

$$\mathrm{Im}(s(a+bi)) = -\sum_{j=1}^{n} \frac{2(\lambda_j + a)b}{((\lambda_j + a)^2 + b^2)^2}(w_j^T q)^2,$$

where i is the imaginary unit. A detailed derivation of the above equation is provided in Lemma 6.2.6. For any $a < 0$ we have $-a > 0 > \lambda_1, \ldots, \lambda_n \Rightarrow 2(\lambda_j + a) < 0$, $j = 1, \ldots, n$ by assumption. Thus $\mathrm{Im}(s(a+bi)) \neq 0 \Rightarrow s(a+bi) \neq 0$ for any $a < 0$ unless $b = 0$. □

Such a shift would not affect the (local/global) optimizers of (P) as

$$\frac{1}{2}x^T(P - \alpha I)x + q^T x = \frac{1}{2}x^T P x + q^T x - \frac{\alpha}{2}||x||_2^2$$

or, since $\|x\|_2 = r$,

$$\frac{1}{2}x^T(P - \alpha I)x + q^T x = \frac{1}{2}x^T P x + q^T x - \frac{\alpha}{2}r^2,$$

where $\alpha \in \mathbb{R}$. Thus, we can always transform (6.8) so that the two rightmost eigenvalues of $\tilde{M}$ are equal to the two rightmost eigenvalues of (6.11).

Having computed the two rightmost eigenvalues/vectors with a projected Arnoldi method applied to M, global/local solutions of (P) can be obtained from the respective eigenvectors. Indeed, recall that according to (6.5) and (6.10), an appropriate (rightmost/second-rightmost) eigenvector $[\tilde{z}_1^*;\ \tilde{z}_2^*]$ of (6.11) gives a solution x^* to (P) as follows

$$\begin{aligned} x^* = Zy^* &= -Z\mathrm{sign}(\tilde{q}^T\tilde{z}_2^*)r\frac{\tilde{z}_1^*}{\|\tilde{z}_1^*\|_2} = -\mathrm{sign}(q^T(Z\tilde{z}_2^*))r\frac{Z\tilde{z}_1^*}{\|Z\tilde{z}_1^*\|_2} \\ &= -\mathrm{sign}(q^T z_2^*)r\frac{z_1^*}{\|z_1^*\|_2} \end{aligned}$$

where $[z_1^*;\ z_2^*]$ is an eigenvector of M. Thus, x^* can be extracted solely by the eigenvector of M unless $z_1^* = 0$, which happens iff (6.8) belongs to the hard case.

Dealing with the hard case: In the hard case one would have to calculate the minimum length solution of

$$(\tilde{P} + \mu^* I)y + \tilde{q} = 0 \tag{6.13}$$

and then construct a global solution x^* as $x^* = Z(y + \alpha\tilde{z}_2^*) = Zy + \alpha z_2^*$, where α is a scalar such that $\|x^*\|_2 = r$. However, the minimum length solution of (6.13) can be obtained by the minimum length solution of

$$\begin{aligned}(P + \mu^* I)x + q &= 0\\ Ax &= 0\end{aligned}$$

which can be calculated via projected MINRES-QLP or a projected CG algorithm [105, Theorem 4.3] [103].

Algorithm 6.1 summarizes the proposed approach for computing global as well as local-nonglobal minimizers for (6.8).

6.2.A Proof of Theorem 6.2.3

In this subsection we provide a proof of our main theoretical result, Theorem 6.2.3, which shows that (T) has a second-order sufficient local-nonglobal minimizer if and only if it does not belong to the hard case and the second-rightmost eigenvalue of the matrix

$$M := \begin{bmatrix} -P & qq^T/r^2 \\ I & -P \end{bmatrix},$$

with counted algebraic multiplicities, is real, simple and not in the spectrum of $-P$. Furthermore, we show that if such a minimizer exists, its Lagrange multiplier μ^ℓ is equal to this second-rightmost eigenvalue.

The proof is based on the following Lemma, which follows from [104]:

Lemma 6.2.5. *A KKT point x^ℓ of* (T) *with Lagrange multiplier μ^ℓ is a second-order sufficient local-nonglobal minimizer iff $q \not\perp w_1$, $\mu^\ell \in (-\lambda_1, -\lambda_2)$ and $s'(\mu^\ell) > 0$.*

Proof. See [104, Theorem 3.1]. □

Algorithm 6.1: Calculation of global and local-nonglobal minimizers of the equality constrained trust-region subproblem (6.8)

Given the problem data $P \in \mathbb{S}^n$, $q \in \mathbb{R}^n$, $r \in \mathbb{R}_+$, $A \in \mathbb{R}^{m\times n}$ where A is full row rank with $m < n-1$ and $\Pi : \mathbb{C}^n \to \mathbb{C}^n$, a projector into the nullspace of A;

Shift P by a multiple of identity so that it becomes negative definite;

Run an Arnoldi method on

$$\Pi\left(\begin{bmatrix} -P & qq^T/r^2 \\ I & -P \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) = \mu \begin{bmatrix} z_1 \\ z_2 \end{bmatrix},$$

where $\Pi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) := \begin{bmatrix} \Pi(x_1) \\ \Pi(x_2) \end{bmatrix}$ for any $x_1, x_2 \in \mathbb{C}^n$, starting from an initial point $\Pi(z) \in \mathbb{C}^{2n}$, to calculate a rightmost eigenvalue/vector pair (μ^g, z^g);

if $z_1^g \neq 0$[7] **then**

  $x^g \leftarrow -\text{sign}(q^T z_2^g) r \frac{z_1^g}{\|z_1^g\|_2}$;

  Resume the projected Arnoldi method to compute a second-rightmost eigenvalue/vector (μ^ℓ, z^ℓ);

  if μ^ℓ *is real, simple and* $z_1^\ell \neq 0$ **then**

    $x^\ell \leftarrow -\text{sign}(q^T z_2^\ell) r \frac{z_1^\ell}{\|z_1^\ell\|_2}$;

    Return x^g and x^ℓ as the unique global and local-nonglobal minimizers;

  else

    Return x^g as the unique minimizer and state that no second-order sufficient local-nonglobal minimizer exists;

  end

else

  Compute the minimum length solution $x_{\min}$ of

$$\begin{aligned} (P + \mu^g I)x + q &= 0 \\ Ax &= 0 \end{aligned}$$

  via projected MINRES-QLP or projected CG [105, Theorem 4.3];

  Solve a quadratic equation to find α_1, α_2 such that $\left\|x_{\min} + \alpha_{1/2} z_2^g\right\|_2 = r$;

  Return $x_{\min} + \alpha_1 z_2^g$, $x_{\min} + \alpha_2 z_2^g$ as global solutions and state that no local-nonglobal minimizer exists;

end

[7] According to Proposition 6.2.10, checking if z_1^g (z_1^l) $\neq 0$ is equivalent to checking if μ^g (μ^l) is an eigevalue of $-P$, except in a set of TRS instances of measure zero.

Before we use Lemma 6.2.5 we will need the following two ancillary results:

Lemma 6.2.6. *For any complex number $a + bi \in \mathbb{C}$ where a, b are real numbers such that a is in the domain of s and $b \neq 0$, we have:*

$$\mathrm{Re}(s(a + bi)) < s(a). \tag{6.14}$$

Proof. Recall the definition of s:

$$s(a + bi) = \sum_{j=1}^{n} \frac{(w_j^T q)^2}{(\lambda_j + a + bj)^2} - r^2$$

and define, for convenience, $\kappa_j = w_j^T q$, $j = 1, \ldots, n$. Thus

$$\begin{aligned} s(a + bi) &= -r^2 + \sum_{j=1}^{n} \frac{\kappa_j^2}{(\lambda_j + a + bi)^2} \\ &= -r^2 + \sum_{j=1}^{n} \kappa_j^2 \frac{(\lambda_j + a)^2 - b^2}{((\lambda_j + a)^2 + b^2)^2} - \kappa_j^2 \frac{2(\lambda_j + a)b}{((\lambda_j + a)^2 + b^2)^2} i, \end{aligned}$$

hence

$$\begin{aligned} \mathrm{Re}(s(a + bi)) &= -r^2 + \sum_{j=1}^{n} \kappa_j^2 \frac{(\lambda_j + a)^2 - b^2}{((\lambda_j + a)^2 + b^2)^2} \\ &< -r^2 + \sum_{j=1}^{n} \kappa_j^2 \frac{(\lambda_j + a)^2}{((\lambda_j + a)^2)^2}, \end{aligned}$$

or, equivalently, $\mathrm{Re}(s(a + bi)) < s(a)$. □

Lemma 6.2.7. *If $s(x) < 0$ for some real x in the domain of $s : \mathbb{C} \to \mathbb{C}$ then s has an equal number of poles and zeros with real parts in $[x, \infty)$.*

Proof. We will prove this result with the argument principle. We propose encircling the half-space

$$S(x) := \{c \in \mathbb{C} \mid c = a + bi, \text{ where } a \geq x, b \in \mathbb{R}\}$$

with the contour of Figure 6.2 and show that it maps to a contour that lies strictly in the left half plane. The suggested closed contour C intersects the real axis at x and at ∞. It consists of two segments: $C_1 = \{\omega + \beta i \mid \beta \in \mathbb{R}\}$, i.e., a segment that is parallel to the imaginary axis, and C_2, a half-circle with infinite radius.

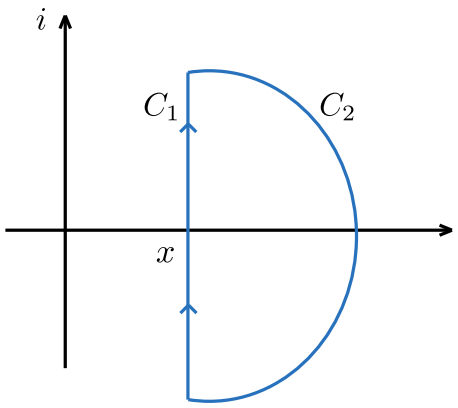


Figure 6.2: Suggested closed contour for encircling $S(x)$.

According to the argument principle the number of poles in S is equal to the number of zeros in S if there are no poles or zeros of s in C and $s(C)$ does not enclose the origin. By examining (6.3) we conclude that s has only real, finite poles. Thus, the contour C does not intersect any pole of s. It remains to show that there exist no zeros of s in C and $s(C)$ does not enclose the origin. We show both by proving that every point in $s(C) = s(C_1) \cup s(C_2)$ has negative real part. For $s(C_2)$, it is easy to see that, since $s(C_2) = \{-r^2\}$ as $\lim_{|w|\to\infty} s(w) = -r^2$. Regarding $s(C_1)$ note that $s(x) < 0$, and thus Lemma 6.2.6 gives $\mathrm{Re}(s(x + \beta i)) \leq s(x) < 0 \; \forall \beta \in \mathbb{R}$, i.e., any point in $s(C_1)$ has negative real part. □

We will now connect the second-rightmost root of the secular equation s with the results of Lemma 6.2.5. Note that, due to Lemma 6.2.6, if s has a real root μ then it cannot have a complex root with real part μ and vice versa. Thus, the concept of "the second-rightmost root of s" is well defined as long as we assume (or are prepared to check) that this root is real.

Lemma 6.2.8.

(i) Suppose that $q \not\perp w_1$ and $\mu \in (-\lambda_2, -\lambda_1)$ is an simple root of s with $s'(\mu) > 0$. Then, μ is the second-rightmost root of s in $\mathbb{C}$.

(ii) Suppose that $q \not\perp w_1, w_2$, $\lambda_1 \neq \lambda_2$, $\mu \in \mathbb{R}$ is the second-rightmost root of s in $\mathbb{C}$ and μ is simple. Then, $\mu \geq -\lambda_2$.

Proof. Note that in both cases $q \not\perp w_1$, thus (T) does not belong to the hard case [105]. Hence s has a real root in $(-\lambda_1, \infty)$ (Theorem 6.2.2) which we denote by μ^g. Moreover, μ^g is simple because $s'(\mu^g) = -2\sum_{j=1}^{n}(w_j^T q)^2/(\lambda_j + \mu^g)^3$ is negative since $\mu^g > -\lambda_1 > \cdots > -\lambda_n$.

In the subsequent analysis, we will frequently refer to the poles of s which are

$$\Big\{-\lambda_i \;\Big|\; i = 1, \ldots, n \text{ such that } q^T w_i \neq 0\Big\},$$

and have double multiplicity, and the following parametric set

$$S(x) := \{c \in \mathbb{C} \mid c = a + bi, \text{ where } a \geq x, b \in \mathbb{R}\}$$

defined for any $x \in \mathbb{R}$.

We will begin with claim (i). Since $\mu < -\lambda_1 < \mu^g$ we conclude that μ is not the rightmost root of s. In order to show that it is the *second* rightmost, it suffices to show that there are only two roots of s in $S(\mu)$.

Since $s(\mu) = 0$ and $s'(\mu) > 0$, we have $s(\mu - \epsilon) < 0$ for a sufficiently small $\epsilon > 0$. According to Lemma 6.2.7 s has equal number of zeros and poles in $S(\mu - \epsilon)$. By assumption $q \not\perp w_1$ and $\mu > -\lambda_2$, thus $-\lambda_1$ is the only (double) pole of s in $S(\mu - \epsilon)$. Hence, the number of zeros in $S(\mu - \epsilon) \subset S(\mu)$ must be two. This concludes the proof of claim (i).

We will now prove claim (ii). Assume the contrary, i.e., that $\mu < -\lambda_2$. Consider first the case where $s'(\mu) > 0$. Since $s(\mu) = 0$ we can choose a sufficiently small $\epsilon > 0$ such that $s(\mu - \epsilon) < 0$. Thus s has equal number of zeros and poles in $S(\mu - \epsilon)$ (Lemma 6.2.7). By assumption $\mu < -\lambda_2$ and $q \not\perp w_1, w_2$, and thus the double poles $-\lambda_1$ and $-\lambda_2$ are in $S(\mu - \epsilon)$. Thus s has (at least) four zeros in $S(\mu - \epsilon)$, and since ϵ can be arbitrarily small, the same holds for $S(\mu)$.

We conclude that there should be extra roots for s in $S(\mu)$, besides the simple roots μ^g and μ. Moreover, these extra roots cannot have real part equal to μ as according to Lemma 6.2.6 $s(\mu + bi) < s(\mu) = 0$ for any real $b \neq 0$. Thus their real parts must be in (μ, ∞). This means that μ is not the second-rightmost root of s, which is a contradiction.

Finally, if $s'(\mu) < 0$, then, using the same arguments as before, we conclude that there exist four roots in $S(\mu + \epsilon)$ for any sufficiently small $\epsilon > 0$. Thus, there exist extra roots (besides the simple root μ^g) with real part in (μ, ∞), which, again, contradicts the assumption that μ is the second-rightmost root of s. □

Lemma 6.2.8, in combination with Lemma 6.2.5, provides an "almost" iff relationship between the second-rightmost root of s and the local-nonglobal minimizer of (T). However, the quantity of primary interest is the second-rightmost eigenvalue of M instead of the second-rightmost root of s. The following result characterizes the spectrum of M and its tight connection with the roots of s:

Proposition 6.2.9. *The spectrum of M consists of the roots of s with matched algebraic multiplicity, and all of the eigenvalues $-\lambda_i$ of $-P$ that are non-simple or have w_i (the $i-$th eigenvector of P) perpendicular to q.*

Proof. We will prove the result by deriving a closed form expression for the characteristic polynomial of M. Since $-I$ and $P + \mu I$ commute, we have [111, Theorem 3]:

$$\det(\mu I - M) = \det\left(\begin{bmatrix} P + \mu I & -qq^T/r^2 \\ -I & P + \mu I \end{bmatrix}\right) = \det((P + \mu I)^2 - qq^T/r^2)$$

or, using the matrix determinant lemma,

$$\det(\mu I - M) = \left(q^T(P + \mu I)^{-2}q - r^2\right)\det((P + \mu I)^2)$$

for all $\mu \neq -\lambda_1, \cdots - \lambda_n$. Thus, recalling (6.2)-(6.3) we have

$$\det(\mu I - M) = s(\mu)\prod_{i=1}^{n}(\lambda_i + \mu)^2 = \left(\sum_{i=1}^{n}\frac{(w_i^T q)^2}{(\lambda_i + \mu)^2} - r^2\right)\prod_{i=1}^{n}(\lambda_i + \mu)^2 \tag{6.15}$$

$$= \sum_{i=1}^{n}(w_i^T q)^2\prod_{j\neq i}(\lambda_j + \mu)^2 - r^2\prod_{i=1}^{n}(\lambda_i + \mu), \tag{6.16}$$

for all $\mu \neq -\lambda_1, \cdots - \lambda_n$. Thus, (6.16) and the characteristic polynomial of M coincide in all $\mathbb{C}$ except perhaps on $2n$ points. It follows from continuity arguments that (6.16) is in fact the characteristic polynomial of M.

By examining (6.15)(2nd leftmost) and (6.16) we conclude that the eigenvalues of M include all the roots of s with matched algebraic multiplicity. The secular

equation s has $2n$ roots whenever the eigenvalues of P are distinct and $w_i^T q \neq 0$ for all $i = 1, \ldots, n$. Otherwise, i.e. for every i with λ_i non-simple or $w_i^T q = 0$, $M \in \mathbb{R}^{2n \times 2n}$ has extra eigenvalues at these $-\lambda_i$, which constitute the only differences between the spectrum of M and the roots of s. □

Proposition 6.2.10. *For every eigenpair $(\mu, [0;\ z_2])$ of M, (μ, z_2) is an eigenpair of $-P$. Conversely, every eigenpair of $(-\lambda_i, [z_1;\ z_2])$ has $z_1 = 0$ and $z_2 \in \mathcal{N}(P - \lambda_i I)$, except in a set of TRS instances of measure zero.*

Proof. The first part follows trivially from the first block row of (6.4). We now prove the "reverse" claim. Define $k := \dim(\ker(P - \lambda_i))$ and note that

$$\dim(\ker(M + \lambda_i I)) = \dim\left(\ker\begin{bmatrix} -P + \lambda_i I & qq^T/r^2 \\ I & -P + \lambda_i I \end{bmatrix}\right) \tag{6.17a}$$

$$= \dim\left(\ker\begin{bmatrix} 0 & qq^T/r^2 - (P - \lambda_i I)^2 \\ I & -P + \lambda_i I \end{bmatrix}\right) \tag{6.17b}$$

$$= \dim\left(\ker\left(\underbrace{(P - \lambda_i I)^2 - qq^T/r^2}_{:=Y}\right)\right), \tag{6.17c}$$

where (6.17b) was obtained by adding a multiple of $(P - \lambda_i I)$ of the second block row to the first block row.

Consider first the case where $q \perp \ker(P - \lambda_i I)$. Then, we can construct a set of k orthonormalnull vectors for $M + \lambda_i I$ as $[0;\ w_j], \ldots, [0;\ w_{j+k}]$ where $w_j, \ldots, w_{j+k}$ arenull vectors of $P - \lambda_i I$. To prove the desired claim, we will show that $\dim(\ker(M + \lambda_i I)) = k$, except in a set of TRS instances of measure zero. Indeed, according to [56, Corollary 10.3.1], $Y = (P - \lambda_i I)^2 - qq^T/r^2$ has

- k zero eigenvalues with eigenvectors in $\ker(P - \lambda_i I)$;
- $n - k - 1$ positive eigenvalues; and
- one eigenvalue in $\left[\zeta - \|q\|_2^2/r^2,\ \zeta\right]$ where ζ is the smallest positive eigenvalue of Y

This last eigenvalue is nonzero, except in a set of TRS instances of measure zero, and we have $\dim\ker(Y) = k = \dim(\ker(M + \lambda_i I))$ due to (6.17).

In the case where $q \not\perp \ker(P-\lambda_i I)$ we can construct a set of $k-1$ orthonormalnull vectors for M from $k-1$null vectors of $P-\lambda_i I$ that are perpendicular to q. Hence, to prove the desired claim, we have to show that $\dim(\ker(M+\lambda_i I)) = k-1$ except in a set of TRS instances of measure zero. Indeed, according to [56, Corollary 10.3.1], Y has

- $k-1$ zero eigenvalues with eigenvectors in $\{x \in \mathbb{R}^n \mid x \in \ker(P-\lambda_i I) \text{ and } x \perp q\}$;
- one negative eigenvalue with the corresponding eigenvector in the one-dimensional subspace $\{x \in \mathbb{R}^n \mid x \in \ker(P-\lambda_i I) \text{ and } x \not\perp q\}$;
- $n-k-1$ positive eigenvalues; and
- one eigenvalue in $[0, \zeta]$ where ζ is defined as before.

Thus the claim follows as before, since this last eigenvalue is nonzero except in a set of TRS instances of measure zero. □

We are now ready to prove the main result of this section. Note, again, that, due to Lemma 6.2.6 and Proposition 6.2.9, if M has a real eigenvalue μ then it cannot have a complex eigenvalue with real part μ and vice versa. Thus, the concept of "the second-rightmost eigenvalue of M" is well defined as long as we assume (or are prepared to check) that this eigenvalue is real.

"only-if" In this case we know that μ^ℓ is the Lagrange multiplier of a second-order sufficient local-nonglobal minimizer and we have to show that

(i) μ^ℓ is not in the spectrum of $-P$.

(ii) (T) is not in the hard case.

(iii) μ^ℓ is a simple eigenvalue of M.

(iv) μ^ℓ is the second-rightmost eigenvalue of M.

Points (i), (ii) and (iii) follow directly from Lemma 6.2.5 and Proposition 6.2.9 (note that $q \not\perp w_1$ implies that (T) is not in the hard case [105]). In order to prove (iv) we will first show that the spectrum (counted with algebraic multiplicities) of M with real part larger than $-\lambda_2$ coincides with the roots of s in the same region. This follows from Proposition 6.2.9, because λ_1 (due to $\mu^\ell \in (-\lambda_2, -\lambda_1)$) and $q^T w_1 \neq 0$. Point (iv) then follows from Lemma 6.2.8(i) which shows that μ^ℓ is the second-rightmost root of s, and thus the second-rightmost eigenvalue of M.

"if" In this case, we know that (T) is not in the hard case and that μ^ℓ is the second-rightmost eigenvalue of M which is real and simple, and we want to show that μ^ℓ is the Lagrange multiplier of the second-order sufficient local-nonglobal minimizer of (T).

Note that, by assumption, μ^ℓ is in the spectrum of M but not in that of $-P$. Thus μ^ℓ is a root of s due to Proposition 6.2.9. As a result, it suffices to show that μ^ℓ is in $(-\lambda_2, -\lambda_1)$ and $s'(\mu^\ell) > 0$ (Lemma 6.2.5).

We will first show that $\mu^\ell \in (-\lambda_2, -\lambda_1)$. Since we are not in the hard case, M has a real eigenvalue in $(-\lambda_1, \infty)$ (Theorem 6.2.2) which is simple and unique in $(-\lambda_1, \infty)$ due to Proposition 6.2.9 and the fact that $s'(\mu) = -2\sum_{j=1}^{n}(w_j^T q)^2/(\lambda_j+\mu)^3$ is negative in that region. Thus, $-\lambda_1$ is not in the spectrum of M as otherwise it would be its second-rightmost eigenvalue (which by assumption is real) and it would violate the assumption that μ^ℓ is not in the spectrum of $-P$. Thus, $\mu^\ell \in (-\infty, -\lambda_1)$. Furthermore, $\mu^\ell \neq -\lambda_2$ by assumption.

It remains to show that $\mu^\ell \geq -\lambda_2$. If $q^T w_2 = 0$ then, $(-\lambda_2, [0; w_2])$ is an eigenpair of M, thus we must have $\mu^\ell > -\lambda_2$ so as to avoid μ^ℓ (the second-rightmost eigenvalue of M) being in the spectrum of $-P$. Consider now the case where $q^T w_2 \neq 0$. Note that $\lambda_1 \neq \lambda_2$ and $q \not\perp w_1$ as otherwise $-\lambda_1$ is in the spectrum of M (6.2.9), which, according to the paragraph above is a contradiction. Thus, $\lambda_1 \neq \lambda_2$ and $q \not\perp w_1, w_2$ resulting in $\mu^\ell \geq -\lambda_2$ due to Lemma 6.2.8(ii).

Finally we show that $s'(\mu^\ell) > 0$ thereby concluding the proof. Note that $s'(\mu^\ell) \neq 0$ since by assumption μ^ℓ is a simple root of s. Assume the contrary, i.e., $s'(\mu^\ell) < 0$. Since s is convex in $(-\lambda_2, -\lambda_1)$ [104, (3.12)] and $\lim_{\mu \to -\lambda_1} s(\mu) = \infty$

we conclude that there exists another pole of s in $(\mu^\ell, -\lambda_1)$. This contradicts the assumption that μ^ℓ is the second-rightmost root of s.

6.3 An active-set algorithm for (P)

We are now in a position to present the main contribution of this chapter, an active-set algorithm for solving (P). We will first present an algorithm for the special case that $r_{\min} = r_{\max} := r$ and then describe how to generalize for any $r_{\min}, r_{\max}$.

6.3.1 Solving (P) when $r_{\min} = r_{\max} := r$

In this section we introduce a primal active-set approach for the optimization of (P) when $r_{\min} = r_{\max} := r$. It will be useful in the subsequent analysis to recall the Karush-Kuhn-Tucker (KKT) conditions of (P), which are

$$\nabla f(x) + A^T\kappa + \mu x = 0 \tag{6.18}$$

$$\kappa \geq 0 \tag{6.19}$$

$$\kappa_i(a_i^T x - b_i) = 0, \quad i = 1, \ldots, m \tag{6.20}$$

$$Ax \leq b, \qquad x^T x = r^2. \tag{6.21}$$

As is typical for a primal active-set approach, our algorithm starts from a given feasible point of (P) and generates iterates that remain feasible for (P) and have non-increasing objective values. At every iteration we treat a subset of the inequality constraints $Ax \leq b$ as equalities. We refer to this subset as the *working set* $\mathcal{W}_k$ and define

$$\bar{A}_k = [a_i]_{i\in\mathcal{W}_k}, \quad \bar{b}_k = [b_i]_{i\in\mathcal{W}_k},$$

where $[\cdot]$ denotes vertical (row-wise) concatenation. To simplify the analysis, we will assume that one of the simplest and most common constraint qualification holds for every iterate of the algorithm:

Assumption 1. *Linear Independence Constraint Qualification [LICQ]*
The LICQ holds for every iterate of the suggested Algorithm, i.e., $\begin{bmatrix}\bar{A}_k \\ \bar{x}_k^T\end{bmatrix}$ *is full row rank.*

Note that the LICQ is not always guaranteed to hold. See the last paragraph of §6.3.1 for more details.

We will first give a brief, schematic description of our algorithm. At every iteration k of our active-set based minimization procedure, we use Algorithm 6.1 to compute minimizers of the subproblem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & \|x\|_2 = r \\ & \bar{A}_k x = \bar{b}_k. \end{array} \tag{SP}$$

We then attempt to move towards those minimizers in the hope that we either (i) "hit" a new constraint or (ii) set x_{k+1} to a minimizer of (SP). Unfortunately due to the nonconvexity of the problem, we might achieve neither (i) nor (ii) by simply moving towards the minimizer of (SP). In that case, which we will show can happen only when (SP) has a unique minimizer, we perform a projected gradient descent followed (if necessary) by a move towards the global minimizer of (SP) that is guaranteed to achieve either (i) or (ii)[8]. Finally, at the end of each iteration, we check for termination and update the working set.

Our overall active-set procedure is shown in Algorithm 6.2. We now proceed in the following subsections to describe in detail each of the Algorithm's steps. For the rest of this section, we will say that a point is "feasible" when it is in the feasible region of (P). Also, given a point x we will call the projection of $\nabla f(x)$ to the nullspace of constraint normals of (SP), $\begin{bmatrix} \bar{A}_k \\ x^T \end{bmatrix}$, as "projected gradient of $f(\cdot)$ at x".

[8]Technically, in this case x_{k+1} will be a second order necessary stationary point of (SP).

Algorithm 6.2: Active Set method for solving (P) when $r_{\min} = r_{\max} := r$

Assume a feasible starting point x_0
Set W_0 to be a subset of the active constraints in x_0
for $k = 1, 2, \ldots$ **do**
  Solve (SP) with Algorithm 6.1;
  if *Algorithm 6.1 returned two distinct minimizers* **then**
    $x_{k+1} \leftarrow$ solution of the "2D subproblem"[9]defined by x_k and the two returned minimizers.
  else
    $x_{k+1} \leftarrow$ solution of the "2D subproblem" defined by x_k, the returned (global) minimizer and the projected gradient of f at x_k.
  end
  if x_{k+1} *is not a minimizer of* (SP) *and no new constraint was "hit"* **then**
    Run projected gradient descent starting from x_{k+1} until a new constraint is "hit" or until convergence;
    if *PGD converged to a feasible* x^s *with indefinite projected Hessian* **then**
      Compute a suitable limiting direction d along which x_{k+1} is a local maximum;
      $x_{k+1} \leftarrow$ solution of the $2D$ subproblem on the plane defined by x_s, a global minimizer of (SP) and d;
  if *a new constraint was "hit"* **then**
    Obtain W_{k+1} by adding one of the blocking constraints to W_k;
  else
    $W_{k+1} \leftarrow$ check_multipliers(x_{k+1}, W_k);
  end
end
Function `check_multipliers`(x, W)
  Compute the Lagrange multipliers (κ, μ) at x that satisfy (6.18)-(6.21) with $A = [a_i]_{i \in \mathcal{W}}$;
  if $\kappa \geq 0$ **then**
    Terminate Algorithm with solution $x^* = x$
  else
    $j \leftarrow \arg\min_{j \in W} \kappa_j$;
    $W \leftarrow W \setminus \{j\}$;
  end

[9] The "2D subproblems" refer to the moves along circular arcs towards the minimizers of (SP)

Move towards the minimizers of (SP)

Notice that Algorithm 6.1 always returns a global minimizer x^g of (SP) and possibly a second minimizer x^*.

Let us consider first the case where two distinct minimizers are returned which is treated in lines 5-6 of Algorithm 6.2. In this case it is always possible to either set x_{k+1} to a minimizer of (SP) or "hit" a new constraint as follows. Consider the circle defined by the intersection of the sphere $\|x\|_2 = r$ and the plane defined by x_k and the two returned minimizers. When (SP) is constrained in this circle, then it can be reduced to a two-dimensional TRS. As such, we will show that it has at most 2 (local/global) maximizers and 2 (local/global) minimizers, except perhaps when x_k is a global minimizer of (SP) in which case we can trivially set $x_{k+1} = x_k$.

Proposition 6.3.1. *A two dimensional TRS has at most 2 minimizers and 2 maximizers except when its objective is constant across its feasible region.*

Proof. It suffices to show the result only for the minimizers, as a negation of the TRS's objective would show the same result for its maximizers. Suppose that the TRS does not belong to the hard case. Then it has a unique global minimizer and at most one local-nonglobal minimizer [104]. If it belongs to the hard case, then only global minimizers can exist. These are intersections of the affine subspace (6.6) and the sphere (6.7). Note that this intersection is either a distinct point, two points or a circle. In the latter case, every feasible point of (T) is a global minimizer. This completes the proof. □

We can identify the two minimizers of this two-dimensional TRS as x^g and x^*. It follows that at least in one of the circular arcs connecting x_k with x^g and x^* the objective value is always less than $f(x_k)$. Thus, by moving into that circular arc one will either end up with x^g or x^* or "hit" a new constraint.

On the other hand, if a single global minimizer x^g was returned, which corresponds to lines 5-6 of Algorithm 6.2, then we consider the circle defined by the intersection of the sphere $\|x\|_2 = r$ and the plane defined by x_k, x^* and the projected gradient of $f(\cdot)$ at x_k. Obviously, if x^* is feasible for (P) then we choose $x_{k+1} = x^*$, but otherwise we are not guaranteed to "hit" a new constraint. This is because the associated two-dimensional TRS might possess a second minimizer,

which is not necessarily a stationary point of (SP), but might be the best feasible solution on this circle.

Perform projected gradient descent (if necessary)

This subsection concerns the case where the procedure described in the previous subsection could neither set x_{k+1} to a minimizer of SP nor "hit" a new constraint. In this case (lines 10-14 of Algorithm 6.2), we proceed by performing projected gradient descent (PGD) on (SP) starting from the current iterate of the active-set algorithm. This is guaranteed to converge to a KKT point x^s with $f(x^s) \leq f(x_k)$ [112, Theorem 4.5]. Suppose that x^s is not feasible for (P). Since the feasible region is a closed set, PGD exits the feasible region in finitely many steps. By stopping the projected gradient descent just before it exits the feasible region, we can find a point x_{k+1} with a newly "hit" constraint.

Consider now the case where x^s is feasible for (P). If the minimum eigenvalue $\lambda_{\min}$ of the projected Hessian of $f(\cdot)$ at x^s is nonnegative (lines 12-14 of Algorithm 6.2), then we set $x_{k+1} = x_s$ and we proceed to the steps outlined in the next subsection. Otherwise $\lambda_{\min} < 0$, and there exists a feasible sequence $\{z_k\}$ converging to x^s with an appropriate limiting direction d

$$d := \lim_{k\to\infty} \frac{z_k - x^s}{\|z_k - x^s\|_2}, \tag{6.22}$$

such that

$$f(z^k) \geq f(x^s) + \frac{1}{2}\lambda_{\min}\|z_k - x^s\|^2 + o\Big(\|z_k - x^s\|^2\Big). \tag{6.23}$$

In practice, we can choose d equal to some projected eigenvector associated with $\lambda_{\min} < 0$. Consider now the circle defined by the intersection of the sphere $\|x\|_2 = r$ and the plane defined by x^s, x^g and the limiting direction d. As before, this can be reduced to a two-dimensional TRS that possesses at most 2 minimizers and 2 maximizers (Proposition 6.3.1). We can identify two of them: x^g which is a global minimum and x^s which, according to (6.23), is a local maximum. It follows that in at least one of the two circular arcs connecting x^s with x^g the objective

value is always less than $f(x^s)$. Thus, by moving into that circular arc, and since x^s is feasible for (P) and x^g infeasible, we can identify a suitable feasible point x_{k+1} such that $f(x_{k+1}) \leq f(x^s) \leq f(x^g)$ for which a new constraint, not in the current working set, becomes active.

Update the working set and check for termination

If a new constraint was identified in the steps above, then we update the working set and proceed to the next iteration (lines 15-16 of Algorithm 6.2). Otherwise, x_{k+1} was set to a second-order necessary stationary point of (SP) and we proceed as follows (lines 17-18 of Algorithm 6.2). Stationarity of x_{k+1} gives

$$\nabla f(x_{k+1}) + \mu x_{k+1} + \sum_{i \in \mathcal{W}_k} a_i \kappa_i = 0$$

for some Lagrange multipliers μ and $\kappa_i \; \forall i \in \mathcal{W}_k$. Thus, (x_{k+1}, μ, κ) satisfy the first KKT condition (6.18) if we define $\kappa_i = 0 \; \forall \notin \mathcal{W}_k$. It follows from feasibility of x_{k+1} and the definition of κ that the third and fourth KKT conditions (6.20)-(6.21) also hold. If we also have that $\kappa_i \geq 0 \; \forall i \in \mathcal{W}_k$, then (x_{k+1}, μ, κ) is a KKT pair for (P) and we terminate the algorithm (lines 23-24 of Algorithm 6.2). If, on the other hand, $\kappa_i < 0$ for some $i \in \mathcal{W}_k$ (lines 25-27 of Algorithm 6.2), then (6.19) is not satisfied and x_{k+1} is not a local minimizer for (P). In fact, the objective $f(\cdot)$ may be decreased by dropping one of the constraints corresponding to a negative Lagrange multiplier [2, Section 12.3].

Finite termination

We will show that Algorithm 6.2 terminates in finitely many outer iterations, assuming that $x_{k+1} \neq x_k$ for every $k = 1, 2, \ldots$. Indeed at every iteration the algorithm either

- Moves to a stationary point of the current TRS subproblem; or
- Activates a new constraint.

Since there can be at most m constraints in the working set it follows that x_k visits a stationary point of the $k-$th TRS subproblem periodically (at least every m iterations). Furthermore, note that every TRS subproblem, resulting from a particular working set, has at most $2n$ sets of stationary points of the same objective value [108, Theorem 4.1 and subsequent comments]. Since there are a finite number of different working set configurations, it follows that there exist finitely many such sets. On the other hand, every constraint that is deleted from the working set has an associated negative Lagrange multiplier, thus every iterate x_k is not a stationary point of the next iteration. Furthermore, note that if x_k is not a stationary point for the $k+1$ subproblem then our algorithm generates x_{k+1} with $f(x_{k+1}) < f(x_k)$ unless $x_k \neq x_{k+1}$ which is excluded by assumption. This means that once the algorithm visits one of these sets of stationary points with equal objective value, it can never visit it again. Hence, the algorithm terminates after finitely many iterations.

It does not seem that it is possible to bound the number of iterations by a polynomial in m, n. Indeed, even finding an initial feasible point is NP-complete, as we show in §6.3.2.

Further Remarks

Similarly to active-set algorithms for quadratic programs, one can always update the working set such that $\bar{A}_k$ is full row rank. However, LICQ might still fail to be satisfied when the gradient of the spherical constraint lies on $\mathcal{R}(\bar{A}_k^T)$. In these cases we terminate the algorithm without guarantees about local optimality.

6.3.2 Solving (P) for any $r_{\min}, r_{\max}$

We are now ready to present an active-set algorithm that solves (P) for any $r_{\min}$ and $r_{\max}$. At every iteration of the suggested active-set algorithm, the spherical inequality constraint will either be in the working set or not. If it is, then we iterate as described in Algorithm 6.2; otherwise we proceed with a generic (nonconvex) quadratic programming active-set algorithm [2, §16.8]. We switch between the two algorithms when an iterate of the QP algorithm hits the spherical boundary or

when the Lagrange Multiplier μ of the norm constraint in Algorithm 6.2 is negative and less than any other Lagrange Multiplier κ_i.

In the special the case where $r_{\min} = 0$, the norm constraint reduces to $\|x\|_2 \leq r_{\max}$. In this case, when a switch from Algorithm 6.2 to the QP Algorithm happens, the projected Hessian of the Lagrangian of $f(\cdot)$ at x_k is positive semidefinite, i.e.

$$Q^T(P + \mu I)Q \succeq 0 \tag{6.24}$$

where $Q := [Q_1 \ q_2]$ is defined by the "thin" QR decomposition of $[\bar{A}_k^T \ x_k]$ and q_2 is an appropriate vector. Recall that for a switch to happen, we need $\mu < 0$; hence $Q^T P Q$ is positive definite. Thus, $Q_1^T P Q_1$, i.e., the projected Hessian of the next iteration that is to be handled by the generic QP algorithm, has at most one nonpositive eigenvalue [54, Corollary 4.1]. This means that even the popular, but less flexible, class of "inertia controlling" QP algorithms can be used as part of the suggested active-set algorithm for solving (P) when $r_{\min} = 0$.

Finding an initial point

Algorithm 6.2 is a primal active-set algorithm and, as such, it requires an initial feasible point. Unfortunately, finding such a point is, in general, NP-complete as we formally establish at the end of this subsection. Nevertheless, we can find a feasible point, or detect infeasibility, for (P) with standard tools, albeit in exponential worst-case complexity, as we proceed to show. Indeed, define the following problems:

$$\begin{array}{ll} \text{maximize} & x^T x \\ \text{subject to} & Ax \leq b \\ & \|x\|_\infty \leq r_{\min} \end{array} \qquad\qquad \begin{array}{ll} \text{minimize} & x^T x \\ \text{subject to} & Ax \leq b \end{array}$$

i.e., a convex minimization problem and a maximization of a convex function subject to linear constraints. Denote with $x^*_{\min}$ and $x^*_{\max}$ the solutions to the convex minimization and maximization problems respectively. Then, (P) is feasible iff $\|x^*_{\min}\|_2 \leq r_{\max}$ and $\|x^*_{\max}\|_2 \geq r_{\min}$, in which case a feasible point for (P) can be found by interpolating between $x^*_{\min}$ and $x^*_{\max}$.

The convex minimization problem above can be solved in polynomial time with interior point methods. On the other hand, the convex maximization problem can have exponential worst-case complexity, but it can be solved to local or even global optimality with standard commercial solvers e.g. with `IBM CPLEX`.

Finally, we formally show that determining if (P) is feasible is NP-complete:

Proposition 6.3.2. *Determining if there is a solution to*

$$\begin{array}{ll} \textit{find} & x \\ \textit{subject to} & Ax \leq b \\ & r_{\min} \leq \|x\|_2 \leq r_{\max} \end{array} \tag{F}$$

is NP-complete.

Proof. Determining if (F) has a solution is NP since we can easily check whether a candidate point x is feasible for (F). Furthermore, it can be decomposed into the following two independent problems: (i) "is there a point in the polytope $Ax \leq b$ such that $\|x\|_2 \leq r_{\max}$?" and (ii) "is there a point in the polytope $Ax \leq b$ such that $\|x\|_2 \geq r_{\min}$?" Problem (i) can be answered in polynomial time with e.g. interior point methods. Thus determining if (F) has a solution is reducible to problem (ii) which is known to be NP-complete [113, Problem 3]. □

6.4 Applications and Experiments

In this section we present numerical results of the suggested active-set algorithms on a range of numerical optimization problems. A `Julia` implementation of the algorithm, along with code for the generation of the results of this section is available at:

`https://github.com/oxfordcontrol/QPnorm.jl`

As described in the previous sections, the algorithm is based on a TRS solver, and a general (nonconvex) Quadratic Programming solver. We implemented these dependencies as separate packages that we also release publicly as listed below.

- `TRS.jl`: A `Julia` package for the computation of global and local-nonglobal minimizers of

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} & \|x\| = r \quad \text{or} \quad \|x\| \leq r \\ & Ax = b, \end{array} \tag{P}$$

  essentially implementing in `Julia` the theoretical results of Section 6.2.1 and [105]. Available at:

  `https://github.com/oxfordcontrol/TRS.jl`

- `GeneralQP.jl`: A `Julia` implementation of [114], i.e., an "inertia controlling" active-set solver for nonconvex, dense quadratic problems, with efficient and numerically stable routines for updating QR decompositions of the working set and LDLt factorizations of the projected Hessian. Available at:

  `https://github.com/oxfordcontrol/GeneralQP.jl`

  This solver is useful as a part of the suggested algorithm for solving (P) according to the remarks of §6.3.2.

For simplicity our implementations of the active-set algorithms are based on dense linear algebra that uses a QR factorization to compute/update a nullspace basis for every working set. Thus the presented results are limited to dense problems, except for §6.4.3 where the special structure of the constraints of (6.29) results in trivial, sparse QR factorizations of the nullspace bases.

Before presenting the results we will discuss some practical considerations regarding the suggested active-set algorithm. The eigenproblems (6.4) associated with the trust-region subproblems of each working set are solved with `ARPACK` [57]. In the occasional cases where `ARPACK` fails, a direct eigensolver is used. Finally, before solving the subproblem (SP) of every iteration, we perform a few projected gradient steps with the hope to quickly activate a new constraint.

We now proceed with the results, starting with the simplest case of dense random problems.

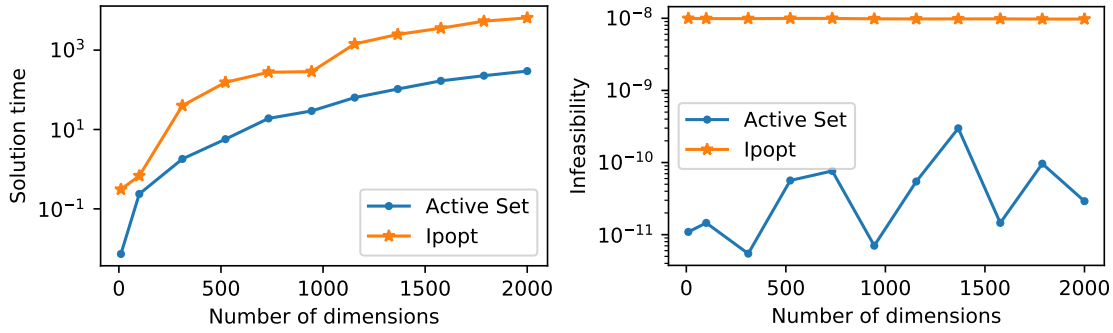


Figure 6.3: Timing results (left) and maximum feasibility violation (right) of the solution of random problems with varying size n generated with the following parameters $P = \text{Symmetric}(\text{randn}(n, n))$, $q = \text{randn}(n)$, $A = \text{randn}(1.5n, n)$, $b = \text{randn}(n)$ and $r = 100$. Hardware used: Intel E5-2640v3, 64GB memory. The maximum feasibility violation is defined as $\max((Ax^* - b)_1, \ldots, (Ax^* - b)_m, |\|x^*\|_2^2 - r^2|, 0)$ for a solution x^*.

6.4.1 Random Dense Constant Norm QPs

We compared the performance of our algorithm against the state-of-the-art nonlinear solver `Ipopt` [115] with its default parameters. We use the `Julia` interface of `Ipopt`, `Ipopt.jl`, which exhibits negligible interface overhead times due to the excellent interfacing capabilities of `Julia` with `C++`.

We consider a set of these randomly generated problems with varying number of variables n. A feasible point for each problem instance is calculated for our Algorithm as described in Section 6.3.2. The time required to compute the initial feasible points is included in the subsequent results.

Figure 6.3 (left) shows the execution times. We observe that our algorithm is significantly faster than `Ipopt` by a factor of up to 50. Both of the solvers achieve practically identical objectives (relative difference less than 10^{-9}) in all of the problem instances, except in two cases where there is a considerable difference due to the fact that the solvers ended up converging in different local minimizers. Finally, Figure 6.3 (right) shows the infeasibility of the returned solution, where we observe that our solver returns solutions of significantly smaller (i.e., better) infeasibility.

6.4.2 Computing search directions for Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is a powerful and popular algorithm that aims to solve the problem

$$\begin{array}{ll} \text{minimize} & g(x) \\ \text{subject to} & h(x) \leq 0, \end{array} \tag{6.25}$$

where $g : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$ are general (nonlinear) functions. At every iteration, SQP minimizes the nonlinear objective over some *search directions*. These search directions can, for example, be obtained by the optimization of a quadratic approximation of the original objective subject to some linear approximation of the original constraints. It is common to introduce a norm constraint $\|\delta x\|_2 \leq r$ that ensures that the solution will be inside a confidence or *trust* region of this approximation thus resulting in a problem of the form (P). This is particularly useful at points where the Hessian of the nonlinear objective is singular or indefinite as the search directions computed by the respective QPs might be unrealistically large or unbounded.

Typically, SQP algorithms solve problems simpler than (P) by e.g. not including the inequality constraints (P) in explicit form for the calculation of the search directions [2, §18.2]. We will show, however, that our solver is capable of computing search directions from (P) directly, without the need for these simplifications.

We demonstrate this on all the problems of the `CUTEst` collection [116] that have linear constraints and number of variables less than 2000. For each of these problems we consider the quadratic approximation

$$\begin{array}{ll} \text{minimize} & f(\delta x) := \frac{1}{2}\delta x^T \nabla^2 g(x_0)\delta x + \delta x^T \nabla g(x_0) + c \\ \text{subject to} & \|\delta x\|_2^2 \leq 1 \\ & \nabla h(x_0)\delta x + h(x_0) \leq 0 \end{array} \tag{6.26}$$

where x_0 is the closest feasible point (in the 2-norm) to the initial point suggested by `CUTEst`, which we compute with `GUROBI` [95]. We discard problems where `GUROBI` fails to calculate an initial feasible point. Furthermore, we do not consider problems for which (6.26) is convex, since these problems can be solved to global optimality in

polynomial time with standard solvers such as `MOSEK` [25]. Finally, some problems in the `CUTEst` collection are parametric; if the default parameters lead to a problem with more than 2000 variables then we choose a parameter that, if possible, leads to a number of variables close to, but not exceeding, 2000. Tables 6.4-6.6 list all the 63 problems considered, along with any non-default parameters.

We compare the performance of our algorithm against `Ipopt` with its default options. For some problems, `Ipopt` terminates without indication of failure but returns a low quality solution. We consider a problem as "solved" using the same criteria as [117]. That is, we require that the overall error of the KKT conditions is less than 10^{-4}, defined as

$$\text{error} := \max(\epsilon_{\text{primal inf}}, \epsilon_{\text{dual inf}}, \epsilon_{\text{stationarity}}, \epsilon_{\text{compl}}) \tag{6.27}$$

with

$$\begin{aligned}
\epsilon_{\text{primal inf}} &:= \max\Big(0, \delta x^T \nabla h(x_0) + h(x_0), \|\delta x\|_2 - 1\Big) \\
\epsilon_{\text{dual inf}} &:= -\min(0, \kappa_1, \ldots, \kappa_m, \mu) \\
\epsilon_{\text{stationarity}} &:= \Big\|\nabla^2 g(x_0)\delta x + \nabla g(x_0) + \kappa^T \nabla h(x_0) + 2\mu x\Big\|_\infty \\
\epsilon_{\text{compl}} &:= \max(\{\min(\kappa_i, |Ax - b|_i)\}, \min(\mu, |\|x\|_2^2 - 1|)),
\end{aligned}$$

and κ, μ are the Lagrange multipliers corresponding to the linear inequality and norm constraints of (6.26). Figure 6.4 shows the performance profile for the problems considered. The performance profile suggests that our algorithm significantly outperforms `Ipopt` on this set of problems especially in reliability, in the sense that we consistently obtain high quality solutions. Note that unlike the dense implementation of our solver, `Ipopt` can exploit the sparsity of the problems efficiently. Further speedups can be brought to our algorithm with a sparse implementation which might allow its use in large scale sparse problems. Moreover, as an active-set solver, our algorithm can efficiently exploit warm starting, which might be highly desirable in SQP where repeated solution of problems of the form (6.26) is required as part of the SQP procedure for minimizing (6.25). This is in contrast to `Ipopt` whose Interior Point nature makes warm starting very difficult to exploit.

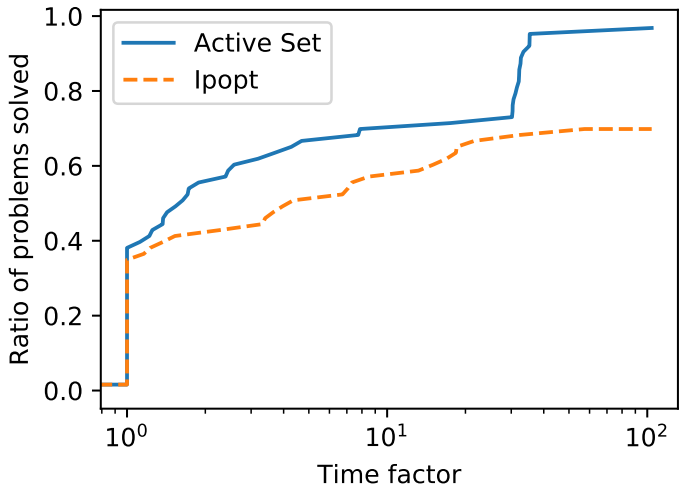


Figure 6.4: Performance graph of the suggested algorithm against `Ipopt` on calculating SQP search directions on all 63 problems of the `CUTEst` library listed in Tables 6.4-6.6. Matrices are passed to `Ipopt` in a sparse format. Hardware used: Intel E5-2640v3, 64GB memory.

6.4.3 Sparse Principal Component Analysis

Principal Component Analysis (PCA) is a standard, widely used dimensionality reduction algorithm. It has applications in numerous fields, including statistics, machine learning, bioinformatics, genetics, meteorology and others. Given a $k \times n$ data matrix D that consists of k points of n variables, PCA suggests a few linear combinations of these variables, which we call *principal vectors*, that explain as much variance of the data as possible. Standard PCA amounts to the following problem

$$\begin{array}{ll} \text{maximize} & x^T \Sigma x \\ \text{subject to} & \|x\|_2 \leq 1 \end{array}$$

where Σ is the covariance matrix of the data. We will assume for the rest of this section that the data matrix D is "centered", i.e., that it has column-wise zero mean. We can then consider Σ to be the empirical covariance matrix $\Sigma := \frac{D^T D}{k-1} \in \mathbb{S}^n_+$. The above problem is essentially an eigenvalue problem on Σ or a singular value problem on D and, as such, can be solved with standard linear algebra tools.

In general, each principal vector is a linear combination of all the variables, i.e., typically all components of the principal vectors are non-zero. This can pose an issue of interpretability of the reduced dimensions, as it is often desirable to express the principal vector as a combination of a few variables, especially when the variables are

associated with a user interpretable meaning. To alleviate this problem, sparsity has to be enforced to the original PCA problem, resulting in a new optimization problem that aims to identify a small set of variables the linear combination of which will hopefully still be able to explain a significant proportion of the variance of the data.

Unfortunately, enforcing sparsity in PCA results in a combinatorial optimization problem. Various remarkably efficient and scalable heuristics have been suggested to avoid this intractability [118], [119]. We will focus on one of the most popular heuristics that is based on a lasso type constraint originally introduced by [120], resulting in the following optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) := -x^T \Sigma x \\ \text{subject to} & \|x\|_2 \leq 1 \\ & \|x\|_1 \leq \gamma, \end{array} \tag{6.28}$$

where $x \in \mathbb{R}^n$ is the decision variable and $\gamma \geq 1$ is a parameter that controls the sparsity of the solution.

We will now show how Problem (6.28) can be solved with our algorithm. Note that (6.28) is not in the standard norm-constrained form (P) that we address. However, we show in the Appendix that it is equivalent to

$$\begin{array}{ll} \text{minimize} & g(w) := -(w_1 - w_2)^T \Sigma (w_1 - w_2) \\ \text{subject to} & \|w\|_2^2 \leq 1 \\ & \mathbf{1}^T w \leq \gamma \\ & w \geq 0, \end{array} \tag{6.29}$$

where $[w_1^T \ w_2^T] := w^T \in \mathbb{R}^{2n}$, in the sense that (6.28) and (6.29) have the same optimal value and every minimizer of (6.29) $\bar{w}$ defines a minimizer $\bar{x} = \bar{w}_1 - \bar{w}_2$ for (6.28). Also, the optimal $\bar{w}$ has the same cardinality as the respective optimal $\bar{x} = \bar{w}_1 - \bar{w}_2$.

The reader might think that solving (6.29) with an active-set method would require a solve time that is polynomial in the number of variables, thus limiting its scalability as compared to simpler gradient based methods that are scalable but might have inferior accuracy and, potentially, weaker convergence guarantees. This is not true however, since our algorithm takes advantage of the sparsity of the iterates. Indeed, excluding the lasso constraint, the working set $\mathcal{W}_k$ of

Algorithm 6.2 applied in Problem (6.29) can be interpreted as the set of variables w_i that are fixed to zero. Thus every subproblem can be trivially reduced to k dimensions, where k is the number of variables w_i that are not included in the current working set. This property is particularly attractive when the user is interested in a very sparse solution and has been recognized in the literature of ℓ_1-penalized active-set algorithms [121, §6].

We will demonstrate the scalability and flexibility of our algorithm by applying it in a large "bag of words" dataset from articles of the New York Times newspaper obtained by the University of California, Irvine (UCI) Machine Learning Repository[10]. In this dataset, every row of the associated data matrix D corresponds to a word used in any of the documents and every column to a document. The entries of the matrix are the number of occurrences of a word in a document. The dataset contains $m \approx 300,000$ articles that consist of $n \approx 102,660$ words (or keywords) with ≈ 70 million non-zeros in the dataset matrix D. We seek to calculate a few sparse principal vectors each of which will hopefully correspond to a user-interpretable category, like politics, economics, the arts etc.

The usefulness of Sparse PCA on this dataset has been already demonstrated in the literature. Both [122] and [123] have generated a set of five sparse principal vectors, each of which is a linear combination of five words. The results of [122] are listed in Table 6.1. The five resulting principal vectors have distinct, user interpretable associated meanings that correspond to sports, economics, politics, education, and US foreign policy. The resulting words can be used to create user interpretable categories of the documents [123]. However, this interpretability is in general lost when we increase the cardinality of the sparse principal vectors, i.e., when we require more representative words for each category. For example, the result of applying `TPower` [119] (a state-of-the art algorithm for Sparse PCA) in the NYtimes dataset to generate a sparse principal vector that consists of 30 words, listed in Table 6.2, does not have a distinct associated meaning. For reference, `TPower` runs in 4.91 seconds on a standard Laptop with an Intel i7-5557U CPU and

[10]Available at `http://archive.ics.uci.edu/ml/datasets/Bag+of+Words`

Table 6.1: Results of [122] on applying sparse PCA on the NYtimes dataset. Each column lists words that correspond to nonzero entries in the respective principal vectors. The resulting principal vectors have clear associated meanings that correspond to sports, economics, elections, education, and the US.

Vector #1	Vector #2	Vector #3	Vector #4	Vector #5
game	companies	campaign	children	attack
play	company	president	program	government
player	million	zzz_al_gore	school	official
season	percent	zzz_bush	student	zzz_u_s
team	stock	zzz_george_bush	teacher	zzz_united_states

8GB of memory and has a resulting variance of 58.79. Another popular algorithm, ℓ_1-`GPower`, runs in 355.86 seconds and results in a variance of 56.66. The results of our active-set algorithm are better than ℓ_1-`GPower`[11] and comparable (slightly inferior) to `TPower`: our algorithm runs in 5.22 seconds and has a resulting variance equal to 58.23. However, our algorithm is more general, in the sense that it can solve any problem of the form (P) which, as we proceed to show, allows for variants of (6.28) that can generate principal vectors with more meaningful interpretation.

Indeed, improving interpretability of the results of Sparse PCA can be achieved by the incorporation of additional constraints in the underlying optimization problem. Unfortunately, most of the algorithms that are specific for sparse PCA do not allow for the presence of extra constraints. An exception is perhaps `GPower` of [118], which is a generic framework for minimizing a concave function f over a compact subset $\mathcal{F}$. The iterates of `GPower` are generated as

$$x_{k+1} = \begin{array}{ll} \text{argmin} & x^T \partial f(x_k)/\partial x_k \\ \text{subject to} & x \in \mathcal{F} \end{array} \tag{6.30}$$

where $\partial f(x_k)/\partial x_k$ is a subgradient of f. [118] showed that for the case of Sparse PCA of (6.28) the above optimization problem reduces to two matrix multiplications with D and D^T. However, in the presence of general linear inequality constraints, solving (6.30) can be at least as hard as solving a linear program (LP), which is obviously considerably more computationally demanding than two matrix multiplications.

[11] Note that ℓ_1 penalized algorithms (such as our algorithm and ℓ_1-`GPower`) do not allow for the specification of the desired sparsity directly. Instead, their sparsity is controlled by a penalty parameter. We find a penalty parameter that results in 30 nonzeros via means of a binary search, while exploiting warm starting of the solvers.

Table 6.2: Results of applying non-negative sparse PCA on the NYtimes dataset with `TPower`. Each column lists words that correspond to nonzero entries in the generated principal vector. Note that increasing the cardinality of the sparse principal vector has severely affected the interpretability of the principal vector as compared to the results of Table 6.1 that have clear associated meaning.

Words	Weights	Words (continued)	Weights (continued)
teamed	−0.45	hitch	−0.08
gameday	−0.37	playland	−0.08
seasonable	−0.35	leaguer	−0.08
playful	−0.28	nightcap	−0.07
playa	−0.24	percentage	0.27
gamesmanship	−0.21	companywide	0.23
coachable	−0.14	companion	0.15
pointe	−0.13	stockade	0.13
runaround	−0.11	marketability	0.12
win95	−0.11	millionaire	0.11
yardage	−0.11	governmental	0.09
guzzle	−0.09	billionaire	0.09
player	−0.09	businesses	0.08
wonder	−0.09	zzz_bush_administration	0.07
ballad	−0.09	fundacion	0.07

In contrast, due to the generality of our approach, imposing any linear constraint is straightforward for our algorithm. To demonstrate this, we impose a non-negativity constraint on the elements of the principal vector to avoid cases where the principal vectors consist of elements that have opposing meanings which happens, for example, in the results of `TPower` listed in Table 6.2. Table 6.3 lists five sparse principal vectors obtained with this approach which take 24.37 seconds to compute on a standard Laptop with an Intel i7-5557U CPU and 8GB of memory. Note that, unlike the results of Table 6.2, the principal vectors of Table 6.3 have a clear associated meaning that can be labeled as sports, economics, politics, US' foreign policy and education. These words could for example be used for organizing the documents in a user interpretable way [123].

Table 6.3: Results on applying non-negative sparse PCA on the NYtimes dataset with the suggested active-set algorithm. Each column lists words that correspond to nonzero entries in the respective principal vectors. The words are sorted in decreasing magnitude of the respective weights. Given a set of computed principal vectors $\{x_i\}$, the next principal vector is computed on the deflated data matrix $D - \sum(Dx_i)x_i^T$. The algorithm is initialized with the 30 most positive, or most negative, entries (depending on which one leads to higher explained variance) of the first singular vector of the respective matrix. Furthermore we perform a standard post-processing polishing step at each resulting principal vector, as described in [118, §4.2]. Note that, unlike Table 6.2 our algorithm results in principal vectors that have clear, user interpretable associated meanings that corresponds to sports, economics, politics, education, and US foreign policy. These words could be used for document classification [123].

Vector #1	Vector #2	Vector #3	Vector #4	Vector #5
team	percent	zzz_al_gore	official	school
game	company	zzz_bush	government	student
season	million	zzz_george_bush	attack	children
player	companies	campaign	zzz_u_s	program
play	stock	president	zzz_united_states	teacher
games	market	election	zzz_bush	parent
point	billion	political	military	high
coach	business	zzz_white_house	zzz_american	percent
run	fund	republican	palestinian	public
win	analyst	voter	zzz_taliban	education
yard	money	vote	zzz_afghanistan	kid
guy	zzz_enron	presidential	terrorist	district
won	firm	democratic	war	test
played	investor	tax	leader	family
hit	sales	zzz_republican	zzz_israel	college
ball	industry	administration	group	home
playing	plan	zzz_clinton	bin	child
league	cost	ballot	laden	class
left	investment	votes	country	group
home	quarter	plan	administration	job
fan	deal	democrat	security	boy
shot	financial	zzz_washington	forces	money
field	customer	zzz_congress	zzz_israeli	help
playoff	economy	poll	zzz_pakistan	mother
night	chief	support	american	system
goal	price	candidate	zzz_washington	friend
final	executive	zzz_florida	nation	girl
start	growth	candidates	terrorism	private
quarterback	earning	governor	weapon	voucher
football	share	vice	foreign	grade

6.5 Conclusion

We introduced an active-set algorithm for the solution of quadratic functions subject to linear constraints and a single norm constraint. The suggested algorithm is based on repeated solutions of the TRS, for which we derived novel theoretical results regarding its local-nonglobal minimizer. The usefulness of the suggested algorithm was demonstrated in a range of real world experiments.

6.A Auxiliary Results

In this Section we show that the suggested active-set algorithm of §6.3.2 can perform ℓ_1 sparse PCA, that is, to solve the problem

$$\begin{array}{ll} \text{minimize} & f(x) := -x^T \Sigma x \\ \text{subject to} & \|x\|_2 \leq 1 \\ & \|x\|_1 \leq \gamma. \end{array}$$

Theorem 6.A.1. *Problems* (6.28) *and* (6.29) *have the same optimal value and every (local) minimizer* $\bar{w}$ *of* (6.29) *defines a (local) minimizer* $\bar{x} = \bar{w}_1 - \bar{w}_2$ *for* (6.28).

Proof. First note that both Problems (6.28) and (6.29) have a minimizer as their objective is smooth and their feasible set compact. For any feasible x of (6.28), the choice

$$w_1 = x_+ \quad \text{and} \quad w_2 = -x_-,$$

where x_+ and x_- are defined element-wise as follows

$$x_+ := \max(x, 0) \quad \text{and } x_- := \min(x, 0),$$

is feasible for (6.29) since $\|w_1\|_2^2 + \|w_2\|_2^2 = \|x\|_2^2 = 1$ and $\mathbf{1}^T w_1 + \mathbf{1}^T w_2 = \|x\|_1 \leq \gamma$, while $f(x) = g([w_1;\ w_2])$ since

$$-(w_1 - w_2)^T \Sigma (w_1 - w_2) = -x^T \Sigma x.$$

Thus the optimal value of (6.29) is less than that of (6.28).

Moreover, for any minimizer $(\bar{w}_1, \bar{w}_2)$ of (6.29), choosing $\bar{x} = \bar{w}_1 - \bar{w}_2$ gives $f(\bar{x}) = g([\bar{w}_1;\ \bar{w}_2])$ with $\bar{x}$ feasible for (6.28) as we proceed to show. Indeed, note that $\bar{w}_1^T \bar{w}_2 \geq 0$ and $\|\bar{w}_1\|_2^2 + \|\bar{w}_2\|_2^2 = 1$ thus

$$\|\bar{x}\|_2^2 = \|\bar{w}_1\|_2^2 + \|\bar{w}_2\|_2^2 - 2\bar{w}_1^T \bar{w}_2 \leq \|\bar{w}_1\|_2^2 + \|\bar{w}_2\|_2^2 = 1 \tag{6.31}$$

and

$$\|\bar{x}\|_1 = \|\bar{w}_1 - \bar{w}_2\|_1 \leq \|\bar{w}_1\|_1 + \|\bar{w}_2\|_1 \leq \gamma. \tag{6.32}$$

Assume that $\bar{x}$ is not a minimizer for (6.28). Then, there exists an $\tilde{x}$ with $\|\tilde{x} - \bar{x}\|_\infty$ arbitrarily small such that $f(\bar{x}) > f(\tilde{x})$. Thus $(\tilde{w}_1, \tilde{w}_2) := (\tilde{x}_+, -\tilde{x}_-)$ is feasible for (6.29) with $g([\tilde{w}_1;\ \tilde{w}_2]) = f(\tilde{x}) < f(\bar{w}) = g([\bar{w}_1;\ \bar{w}_2]))$ and

$$\left\| \begin{bmatrix} \tilde{w}_1 - \bar{w}_1 \\ \tilde{w}_2 - \bar{w}_2 \end{bmatrix} \right\|_\infty = \left\| \begin{bmatrix} \tilde{x}_+ - \bar{x}_+ \\ -(\tilde{x}_- - \bar{x}_-) \end{bmatrix} \right\|_\infty \leq \|\tilde{x} - \bar{x}\|_\infty \tag{6.33}$$

i.e., $(\tilde{w}_1, \tilde{w}_2)$ is arbitrarily close to $(\bar{w}_1, \bar{w}_2)$. This contradicts the assumption that $(\bar{w}_1, \bar{w}_2)$ is a local minimizer for (6.29).

Finally, since the optimal value of (6.29) is less than the one of (6.28) and the global optimum $(\bar{w}_1, \bar{w}_2)$ of (6.29) defines a feasible $\bar{x}$ for (6.28) with $g([\bar{w}_1;\ \bar{w}_2]) = f(\bar{x})$ we conclude that the optimal values of (6.28) and (6.29) coincide. □

Furthermore, the following complementarity condition holds which implies that the sparsity of a solution $\bar{w}$ of (6.29) is equal to the sparsity of the solution $\bar{x} = \bar{w}_1 - \bar{w}_2$ for Problem (6.28).

Lemma 6.A.2. *Every (local) minimizer $\bar{w}$ of* (6.29) *has $\bar{w}_1^T \bar{w}_2 = 0$.*

Proof. Assume the contrary, i.e. that $\bar{w}_1^T \bar{w}_2 \neq 0$. Due to feasibility of $\bar{w} = [\bar{w}_1;\ \bar{w}_2]$ we have $\bar{w} \geq 0$. Thus, there exists an $i \in \{1, \ldots, n\}$ such that $\bar{w}_{(i)}, \bar{w}_{(i+n)} > 0$. Define $\tilde{w}$ element-wise as

$$\tilde{w}_{(k)} = \begin{cases} \bar{w}_{(k)} - \epsilon, & k = i, i+n \\ \bar{w}_{(k)}, & \text{otherwise} \end{cases} \tag{6.34}$$

for $\epsilon > 0$ sufficiently small so that $\tilde{w} > 0$. We will show that for a sufficiently small, positive α the point $(1+\alpha)\tilde{w}$ is feasible and of smaller objective value than the local

minimizer $\bar{w}$ while $\|\bar{w} - (1+\alpha)\tilde{w}\|$ can become arbitrarily small for an appropriate choice of α and ϵ. This will be a contradiction of (local) optimality of $\bar{w}$.

Indeed, note that $\mathbf{1}^T\tilde{w} < \mathbf{1}^T\bar{w} \leq \gamma$ and $\|\tilde{w}\|_2 < \|\bar{w}\|_2 \leq 1$, thus $(1+\alpha)\tilde{w}$ is feasible for any $\alpha > 0$ sufficiently small. Moreover, $g(\tilde{w}) = g(\bar{w})$ giving $g((1+\alpha)\tilde{w}) = (1+\alpha)^2 g(\bar{w}) < g(\bar{w})$, since $g(\bar{w}) < 0$ as otherwise $\bar{w}$ would be a global maximizer of (6.29). □

Finally, we present detailed comparison results of our algorithm against `Ipopt` on the problems described in Section 6.4.2.

Table 6.4: Comparison of the suggested active-set algorithm against `Ipopt` in calculating SQP search directions on 63 nonconvex problems of the `CUTEst` library. These include all the problems with linear inequality constraints, at most $n = 2000$ variables and an indefinite Hessian at the initial point. The initial point is computed with `GUROBI` as the nearest feasible point (in the two norm) to the initial point suggested by `CUTEst`. The column m denotes the number of inequality constraints, while columns $t, \delta f^*,$ error denote the solution time (in seconds), the difference in objective value $f(\delta x^*) - f(0)$ achieved and the residual error defined by (6.27) of the suggested active-set solver. Similarly, $\delta f^*_{\text{ipopt}}, \text{error}_{\text{ipopt}}$ denote respectively identical metrics for `Ipopt`. $t^{\text{sparse}}_{\text{ipopt}}$ ($t^{\text{dense}}_{\text{ipopt}}$) denotes `Ipopt`'s timings when the matrices are passed in sparse (dense) format. Hardware used: Intel E5-2640v3, 64GB memory.

Problem	Parameter	n	m	t	$t^{\text{dense}}_{\text{ipopt}}$	$t^{\text{sparse}}_{\text{ipopt}}$	δf^*	$\delta f^*_{\text{ipopt}}$	error	$\text{error}_{\text{ipopt}}$
AVION2		49	113	4.44×10^{-2}	4.67×10^{-2}	1.72×10^{-2}	-3.65×10^{2}	-3.65×10^{2}	1.48×10^{-10}	5.43×10^{-4}
BLOCKQP1	N=100	210	521	7.18×10^{-3}	1.31	5.07×10^{-2}	-6.75	-6.75	9.99×10^{-16}	9.47×10^{-9}
BLOCKQP2	N=100	210	521	1.33×10^{-2}	1.23	5.29×10^{-2}	-6.75	-6.75	6.66×10^{-16}	9.37×10^{-9}
BLOCKQP3	N=100	210	521	1.55×10^{-2}	7.11×10^{-1}	3.53×10^{-2}	-4.02	-4.02	2.16×10^{-15}	9.22×10^{-9}
BLOCKQP4	N=100	210	521	9.19×10^{-3}	7.81×10^{-1}	4.00×10^{-2}	-4.02	-4.02	1.02×10^{-14}	9.00×10^{-9}
BLOCKQP5	N=100	210	521	9.13×10^{-3}	7.25×10^{-1}	3.33×10^{-2}	-4.02	-4.02	1.16×10^{-14}	9.33×10^{-9}
BLOWEYA	N=100	202	304	2.06×10^{-1}	6.15×10^{-1}	4.80×10^{-2}	-2.90×10^{-3}	-2.90×10^{-3}	1.10×10^{-12}	4.46×10^{-5}
BLOWEYB	N=100	202	304	3.52×10^{-1}	5.78×10^{-1}	4.47×10^{-2}	-2.57×10^{-3}	-2.58×10^{-3}	1.45×10^{-12}	4.83×10^{-5}
BLOWEYC	N=100	202	304	3.70×10^{-2}	5.04×10^{-1}	4.55×10^{-2}	-2.71×10^{-3}	-2.72×10^{-3}	3.04×10^{-15}	3.16×10^{-5}
EQC		9	21	2.37×10^{-4}	3.36×10^{-2}	3.61×10^{-2}	-1.65	-4.90×10^{3}	0.00	5.73
EXPFITA		5	22	9.99×10^{-4}	5.86×10^{-3}	7.31×10^{-3}	-1.53×10^{1}	-1.53×10^{1}	7.81×10^{-12}	1.16×10^{-7}
EXPFITB		5	102	8.98×10^{-4}	1.02×10^{-2}	1.35×10^{-2}	-7.62×10^{1}	-7.62×10^{1}	1.55×10^{-11}	2.94×10^{-6}
EXPFITC		5	502	2.64×10^{-3}	4.95×10^{-2}	6.19×10^{-2}	-3.81×10^{2}	-3.81×10^{2}	8.58×10^{-13}	1.28×10^{-4}
FERRISDC	n=100	1100	2110	1.63×10^{1}	4.87×10^{2}	2.11	-2.31×10^{-1}	-2.31×10^{-1}	9.98×10^{-9}	4.94×10^{-5}
GOULDQP1		32	81	4.17×10^{-3}	2.65×10^{-2}	1.42×10^{-2}	-8.79×10^{2}	-8.79×10^{2}	9.78×10^{-13}	7.85×10^{-5}
HIMMELBJ		45	61	4.42	3.33×10^{-2}	2.12×10^{-2}	-7.39×10^{1}	-7.39×10^{1}	3.59	1.14×10^{-5}
HS105		8	17	1.35×10^{-3}	4.02×10^{-3}	4.48×10^{-3}	-8.08×10^{1}	-8.08×10^{1}	1.45×10^{-12}	2.67×10^{-7}
HS24		2	5	3.55×10^{-4}	6.28×10^{-3}	6.52×10^{-3}	-1.31×10^{-1}	-1.31×10^{-1}	2.22×10^{-16}	1.62×10^{-8}
HS36		3	7	2.95×10^{-4}	5.02×10^{-3}	5.45×10^{-3}	-1.83×10^{2}	-1.83×10^{2}	7.11×10^{-14}	9.97×10^{-9}
HS37		3	8	2.95×10^{-4}	5.69×10^{-3}	4.99×10^{-3}	-1.83×10^{2}	-1.83×10^{2}	9.95×10^{-14}	9.97×10^{-9}
HS41		4	9	1.63×10^{-4}	4.82×10^{-3}	5.28×10^{-3}	-1.56×10^{-2}	-1.56×10^{-2}	5.55×10^{-17}	1.33×10^{-8}
HS44		4	10	4.90×10^{-4}	8.34×10^{-3}	6.47×10^{-3}	-1.30	-1.30	2.44×10^{-15}	3.58×10^{-8}
HS44NEW		4	10	6.56×10^{-4}	4.32×10^{-3}	4.42×10^{-3}	-1.85	-1.85	2.22×10^{-15}	8.04×10^{-9}
HS55		6	14	2.52×10^{-4}	5.74×10^{-3}	5.44×10^{-3}	-1.00	-2.24	4.00	5.62×10^{-5}

Table 6.5: Continuation of Table 6.4

Problem	Parameter	n	m	t	$t^{\text{dense}}_{\text{ipopt}}$	$t^{\text{sparse}}_{\text{ipopt}}$	δf^*	$\delta f^*_{\text{ipopt}}$	error	$\text{error}_{\text{ipopt}}$
LEUVEN7		360	1605	5.52×10^{-1}	6.02	3.37×10^{-1}	-4.86×10^{2}	-4.86×10^{2}	2.16×10^{-12}	5.71×10^{-5}
LINCONT		1257	2435	6.70×10^{1}	4.71×10^{2}	6.44×10^{-1}	-1.11×10^{-18}	-1.56×10^{-5}	2.90×10^{-10}	3.04×10^{-7}
MPC10		1530	3571	2.98×10^{1}	5.95×10^{2}	9.30×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.64×10^{-11}	9.90×10^{-5}
MPC11		1530	3571	3.14×10^{1}	6.48×10^{2}	9.61×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	2.26×10^{-11}	1.35×10^{-4}
MPC12		1530	3571	3.02×10^{1}	6.63×10^{2}	9.94×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.45×10^{-11}	9.20×10^{-5}
MPC13		1530	3571	3.14×10^{1}	6.05×10^{2}	8.91×10^{-1}	-3.15×10^{3}	-3.15×10^{3}	1.52×10^{-11}	1.89×10^{-4}
MPC14		1530	3571	3.04×10^{1}	6.18×10^{2}	9.32×10^{-1}	-3.15×10^{3}	-3.15×10^{3}	2.79×10^{-11}	1.05×10^{-4}
MPC15		1530	3571	3.23×10^{1}	5.99×10^{2}	9.14×10^{-1}	-3.15×10^{3}	-3.15×10^{3}	2.21×10^{-11}	1.07×10^{-4}
MPC16		1530	3571	3.24×10^{1}	6.27×10^{2}	9.19×10^{-1}	-3.15×10^{3}	-3.15×10^{3}	1.62×10^{-11}	1.07×10^{-4}
MPC2		1530	3571	2.82×10^{1}	5.60×10^{2}	8.77×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.16×10^{-11}	1.06×10^{-4}
MPC3		1530	3571	2.78×10^{1}	6.10×10^{2}	9.16×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.22×10^{-11}	1.05×10^{-4}
MPC4		1530	3571	3.02×10^{1}	6.35×10^{2}	9.39×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.55×10^{-11}	1.03×10^{-4}
MPC5		1530	3571	2.94×10^{1}	6.34×10^{2}	9.33×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.57×10^{-11}	1.04×10^{-4}
MPC6		1530	3571	3.04×10^{1}	5.72×10^{2}	9.08×10^{-1}	-3.15×10^{3}	-3.15×10^{3}	1.56×10^{-11}	1.91×10^{-4}
MPC7		1530	3571	2.90×10^{1}	6.46×10^{2}	9.59×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.86×10^{-11}	1.77×10^{-4}
MPC8		1530	3571	2.76×10^{1}	5.97×10^{2}	9.02×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.36×10^{-11}	1.50×10^{-4}
MPC9		1530	3571	2.95×10^{1}	6.46×10^{2}	9.47×10^{-1}	-3.16×10^{3}	-3.16×10^{3}	1.52×10^{-11}	1.04×10^{-4}
NASH		72	138	2.02×10^{-2}	9.08×10^{-2}	2.35×10^{-2}	-1.34×10^{-16}	-2.56×10^{-5}	1.10×10^{-13}	2.65×10^{-7}
NCVXQP1	N=1000	1000	2500	3.47×10^{-1}	5.04×10^{1}	2.53×10^{-1}	-7.82×10^{4}	-7.82×10^{4}	9.09×10^{-12}	3.34×10^{-7}
NCVXQP2	N=1000	1000	2500	3.62×10^{-1}	5.02×10^{1}	2.54×10^{-1}	-5.74×10^{4}	-5.74×10^{4}	1.27×10^{-11}	2.51×10^{-7}
NCVXQP3	N=1000	1000	2500	3.68×10^{-1}	4.79×10^{1}	2.40×10^{-1}	-5.00×10^{4}	-5.00×10^{4}	1.27×10^{-11}	2.73×10^{-7}
NCVXQP4	N=1000	1000	2250	3.50×10^{-1}	3.45×10^{1}	2.04×10^{-1}	-7.82×10^{4}	-7.82×10^{4}	5.18×10^{-11}	3.34×10^{-7}
NCVXQP5	N=1000	1000	2250	3.55×10^{-1}	3.45×10^{1}	2.06×10^{-1}	-5.74×10^{4}	-5.74×10^{4}	1.71×10^{-9}	2.51×10^{-7}

Table 6.6: Continuation of Table 6.4

Problem	Parameter	n	m	t	$t^{\text{dense}}_{\text{ipopt}}$	$t^{\text{sparse}}_{\text{ipopt}}$	δf^*	$\delta f^*_{\text{ipopt}}$	error	$\text{error}_{\text{ipopt}}$
NCVXQP6	N=1000	1000	2250	3.64×10^{-1}	3.30×10^{1}	1.93×10^{-1}	-5.00×10^{4}	-5.00×10^{4}	5.82×10^{-11}	2.73×10^{-7}
NCVXQP7	N=1000	1000	2750	3.58×10^{-1}	7.04×10^{1}	2.94×10^{-1}	-7.82×10^{4}	-7.82×10^{4}	5.13×10^{-10}	3.34×10^{-7}
NCVXQP8	N=1000	1000	2750	3.67×10^{-1}	7.37×10^{1}	2.93×10^{-1}	-5.74×10^{4}	-5.74×10^{4}	8.31×10^{-9}	2.51×10^{-7}
NCVXQP9	N=1000	1000	2750	3.84×10^{-1}	6.71×10^{1}	2.79×10^{-1}	-5.00×10^{4}	-5.00×10^{4}	2.12×10^{-8}	2.73×10^{-7}
PENTAGON		6	15	1.09×10^{-3}	9.63×10^{-3}	9.26×10^{-3}	-2.70×10^{-2}	-2.27×10^{-2}	8.27×10^{-15}	2.09×10^{-6}
QC		9	22	2.12×10^{-4}	1.35×10^{-2}	1.22×10^{-2}	-1.11×10^{2}	-1.11×10^{2}	0.00	1.88×10^{-5}
QCNEW		9	21	2.28×10^{-4}	1.66×10^{-2}	1.51×10^{-2}	-1.65	-4.11×10^{1}	0.00	1.99×10^{-4}
QPNBLEND		83	157	1.50×10^{-2}	6.94×10^{-2}	2.06×10^{-2}	-9.28×10^{-3}	-9.28×10^{-3}	2.50×10^{-13}	4.68×10^{-5}
QPNBOEI1		384	891	6.12×10^{-1}	1.12×10^{1}	1.65×10^{-1}	-4.37×10^{1}	-4.37×10^{1}	7.79×10^{-12}	8.29×10^{-5}
QPNBOEI2		143	363	4.07×10^{-2}	1.01	6.23×10^{-2}	-1.00×10^{3}	-1.00×10^{3}	1.22×10^{-9}	8.42×10^{-5}
QPNSTAIR		467	905	7.41×10^{-1}	1.50×10^{1}	2.32×10^{-1}	-1.15×10^{2}	-1.15×10^{2}	4.35×10^{-11}	2.31×10^{-4}
SOSQP1	N=1000	2000	5001	1.03	3.39×10^{2}	2.19×10^{-1}	-5.00×10^{-1}	-5.00×10^{-1}	3.11×10^{-14}	4.57×10^{-9}
SOSQP2	N=1000	2000	5001	1.19	2.17×10^{3}	1.06	-5.00×10^{-1}	-5.00×10^{-1}	4.94×10^{-15}	4.99×10^{-9}
STATIC3		434	240	9.99×10^{-1}	7.63×10^{-1}	5.71×10^{-2}	-1.47×10^{4}	-1.47×10^{4}	4.60×10^{-6}	1.46×10^{-4}
STNQP1	P=10	1025	2560	3.91×10^{-1}	5.02×10^{1}	1.63×10^{-1}	-1.63×10^{3}	-1.63×10^{3}	2.56×10^{-13}	1.05×10^{-8}
STNQP2	P=10	1025	2560	3.92×10^{-1}	4.36×10^{1}	1.60×10^{-1}	-1.63×10^{3}	-1.63×10^{3}	4.55×10^{-13}	1.05×10^{-8}

7
Conclusions

In this section we conclude the thesis by summarizing the main contributions and suggesting some directions for future research.

Batch Bayesian Optimization: Bayesian Optimization is a powerful Machine Learning algorithm used for the hyperparameter tuning of very expensive black-box processes. In applications where there exists the capacity to perform multiple evaluations of the black-box process in parallel, existing Bayesian Optimization algorithms are either reliant on expensive multidimensional integrations or on heuristics. In chapter 2 we have shown that by adopting a distributionally ambiguous framework, we can avoid integrations entirely, and rely instead on Semidefinite Programming. Crucially, we proved that first and second order information of the resulting acquisition criterion are available as easy-to-compute by-products of the Semidefinite solution. Our approach been shown to perform favorably in a number of synthetic functions and a reinforcement learning application.

The relevance of this result is that Batch Bayesian Optimization can be performed, without heuristics, even in situations where the black-box function can be evaluated at dozens of different locations, in parallel.

Directions of Future Research: In the Distributional Ambiguous framework that we used, we considered *all* distributions that shared the same mean and covariance

as the Gaussian posterior distribution predicted by the Gaussian Process surrogate model. Results from the Distributional Robust Community can be used to tighten the set of considered distributions to more closely match the normal distribution. For example, [44] shows how to incorporate constraints like unimodality, symmetry or bounded support in the distribution set.

Approximate ADMM for Semidefinite Programming: The Bayesian Optimization algorithm presented in chapter 2 requires an efficient Semidefinite Optimization Solver. Even though state-of-the-art SDP solvers allowed the Algorithm of chapter 2 to scale BO beyond what is possible with existing integration-based approaches, there might be the need to apply BO in even more parallelizable black-box models. This requires solving large Semidefinite problems faster.

Chapter 3 addresses this issue by suggesting an approximate ADMM method that can be significantly faster than existing approaches. We showed that by projecting into the Semidefinite Cone approximately, using state-of-the-art eigensolvers, a significant speedup can be brought to the well known ADMM algorithm when solving Semidefinite Problems. For example, the Bayesian Optimization application of Chapter 2 can get a tenfold speedup with our approach, while examples from the SDPLIB collection get accelerated by up to 20x. Moreover, we prove novel theorems that ensure that the asymptotic convergence of our algorithm is the same as in exact ADMM.

The importance of this contribution is twofold: First it shows that the class of Krylov-Subspace eigensolvers are very useful to ADMM, in that it can provide significant speedups, and, secondly, that this can be done while preserving ADMM's asymptotic behavior.

Directions of Future Research: Explore other state-of-the art eigensolvers such as [58, Algorithm II] and [59].

Accuracy of Approximate Projection to the Semidefinite Cone: In order to prove that the approximate ADMM of chapter 3 converges (either to a solution or to a certificate of infeasibility), we had to show that the projection errors could be controlled appropriately. Standard eigenvalue perturbation theory suggests that the projection error would be inversely proportional to the spectral gap, yielding poor bounds in the presence of small eigenvalues. This would be both practically and theoretically problematic since the spectral gap is, in general, unknown. In Chapter 4 we showed that a small spectral gap is of no concern, by deriving bounds that are gap independent. Our bounds are easy to compute and were essential in proving convergence for the Algorithm of chapter 3.

Directions for future research: Provide a bound of the projection accuracy for other norms, beyond the Frobenius norm, or, ideally, provide a bound for any unitarily invariant norm.

Matrix Scaling: Scaling of symmetric matrices to become doubly stochastic, is a standard operation arising in many scientific fields. Usually, this is achieved by a very efficient algorithm, called Matrix Balancing, that finds the closest doubly stochastic matrix using the mutual entropy as a distance metric. In Chapter 5 we show that the use of the Frobenius norm as a distance metric can also lead to a very efficient algorithm that can exploit the sparsity and structure of the original matrices. We demonstrated the efficiency of our approach in matrices arising from the human genome, clustering applications, and the SuiteSparse collection.

Nonconvex Optimization: Finally, we considered the solution of nonconvex quadratic problems that include a norm constraint. We devised an active set algorithm that operates by solving a series of Trust Region Subproblems. To accommodate the needs of our algorithm, we proved novel results for TRS regarding the existence and the computation of its local-nonglobal minimizers. We demonstrated the advantages of our algorithm in problems arising in document classification and Sequential Quadratic Programming.

References

[1] G. H. Golub and C. F. Van Loan. *Matrix Computations*. 4th. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[2] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd. New York: Springer, 2006.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] K. P. Murphy. *Machine Learning A Probabilistic Perspective)*. Cambridge, Massachusetts, USA: MIT Press, 2012.

[5] E. Brochu, V. M. Cora, and N. de Freitas. "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning". In: *ArXiv e-prints* (2010). arXiv: `1012.2599 [cs.LG]`.

[6] S. Boyd et al. "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Found. Trends Mach. Learn.* 3.1 (2011), pp. 1–122.

[7] J. Eckstein and D. P. Bertsekas. "On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators". In: *Mathematical Programming* 55.1 (1992), pp. 293–318.

[8] G. Banjac et al. "Infeasibility Detection in the Alternating Direction Method of Multipliers for Convex Optimization". In: *Optimization-Online e-prints* (2017).

[9] J. Snoek, H. Larochelle, and R. P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems*. 2012, pp. 2951–2959.

[10] D. Lizotte et al. "Automatic Gait Optimization with Gaussian Process Regression". In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI'07. Hyderabad, India: Morgan Kaufmann Publishers Inc., 2007, pp. 944–949.

[11] B. Shahriari et al. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1 (2016), pp. 148–175.

[12] D. R. Jones. "A Taxonomy of Global Optimization Methods Based on Response Surfaces". In: *Journal of Global Optimization* 21.4 (2001), pp. 345–383.

[13] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[14] J. Gonzalez, M. A. Osborne, and N. D. Lawrence. "GLASSES : Relieving The Myopia Of Bayesian Optimisation". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2016. URL: http://arxiv.org/pdf/1510.06299v1.pdf (visited on 10/27/2015).

[15] T. Desautels, Krause A., and J. W. Burdick. "Parallelizing Exploration-Exploitation Tradeoffs in Gaussian Process Bandit Optimization". In: *Journal of Machine Learning Research* 15 (2014), pp. 4053–4103. URL: http://jmlr.org/papers/v15/desautels14a.html.

[16] A. Shah and Z. Ghahramani. "Parallel Predictive Entropy Search for Batch Global Optimization of Expensive Objective Functions". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes et al. Curran Associates, Inc., 2015, pp. 3330–3338.

[17] V. Picheny, T. Wagner, and D. Ginsbourger. "A benchmark of kriging-based infill criteria for noisy optimization". In: *Structural and Multidisciplinary Optimization* 48.3 (2013), pp. 607–626.

[18] C. Chevalier and D. Ginsbourger. "Fast Computation of the Multi-Points Expected Improvement with Applications in Batch Selection". In: *Learning and Intelligent Optimization: 7th International Conference, LION 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 59–69.

[19] S. Marmin, C. Chevalier, and D. Ginsbourger. "Differentiating the Multipoint Expected Improvement for Optimal Batch Design". In: *Machine Learning, Optimization, and Big Data: First International Workshop, MOD 2015, Taormina, Sicily, Italy, July 21-23, 2015, Revised Selected Papers*. Ed. by Panos Pardalos et al. Cham: Springer International Publishing, 2015, pp. 37–48.

[20] A. Genz and F. Bretz. *Computation of Multivariate Normal and T Probabilities*. 1st. Springer Publishing Company, Incorporated, 2009.

[21] J. Gonzalez et al. "Batch Bayesian Optimization via Local Penalization". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Ed. by Arthur Gretton and Christian C. Robert. Vol. 51. Proceedings of Machine Learning Research. Cadiz, Spain: PMLR, 2016, pp. 648–657. URL: http://proceedings.mlr.press/v51/gonzalez16a.html.

[22] D. Ginsbourger et al. "A multi-points criterion for deterministic parallel global optimization based on gaussian processes". In: *Journal of Global Optimization, in revision* (2009).

[23] S. Zymler, D. Kuhn, and B Rustem. "Distributionally robust joint chance constraints with second-order moment information". In: *Mathematical Programming* 137.1 (2013), pp. 167–198.

[24] B. O'Donoghue et al. *SCS: Splitting Conic Solver, version 1.2.6*. https://github.com/cvxgrp/scs. 2016.

[25] ApS MOSEK. *The MOSEK optimization toolbox for Python manual*. 2015. URL: http://www.mosek.com/.

[26] M. Garstka, M. Cannon, and P. J. Goulart. *COSMO: A conic operator splitting method for convex conic problems*. arXiv 1901.10887. 2019.

[27] L. Vandenberghe and S. Boyd. "Semidefinite Programming". In: *SIAM Review* 38.1 (1996), pp. 49–95. URL: http://www.jstor.org/stable/2132974.

[28] B. P.G. Van Parys, P. J. Goulart, and M. Morari. "Distributionally robust expectation inequalities for structured distributions". In: *Optimization-Online e-prints* (2015). URL: http://www.optimization-online.org/DB_HTML/2015/05/4896.html.

[29] D. Ginsbourger, R. Le Riche, and L. Carraro. "Kriging Is Well-Suited to Parallelize Optimization". In: *Computational Intelligence in Expensive Optimization Problems*. Ed. by Y. Tenne and C. K. Goh. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 131–162.

[30] B O'Donoghue et al. "Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding". In: *Journal of Optimization Theory and Applications* 169.3 (2016), pp. 1042–1068.

[31] Y. Zheng et al. "Chordal decomposition in operator-splitting methods for sparse semidefinite programs". In: (2017). URL: https://arxiv.org/abs/1707.05058.

[32] R. H. Byrd, J. Nocedal, and R. A. Waltz. "Knitro: An Integrated Package for Nonlinear Optimization". In: *Large-Scale Nonlinear Optimization*. Ed. by G. Di Pillo and M. Roma. Boston, MA: Springer US, 2006, pp. 35–59.

[33] R. Fletcher. *Practical Methods of Optimization; (Second Edition)*. New York, NY, USA: Wiley-Interscience, 1987.

[34] J. Schulman et al. "Proximal Policy Optimization Algorithms". In: *ArXiv e-prints* (2017). arXiv: 1707.06347 [cs.LG].

[35] P. Dhariwal et al. *OpenAI Baselines*. https://github.com/openai/baselines. 2017.

[36] A. Shah, A. G. Wilson, and Z. Ghahramani. "Student-t Processes as Alternatives to Gaussian Processes". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*. 2014, pp. 877–885. URL: http://jmlr.org/proceedings/papers/v33/shah14.html.

[37] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. "Gaussian Processes for Data-Efficient Learning in Robotics and Control". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.2 (2015), pp. 408–423.

[38] J.B. Lasserre. *Moments, Positive Polynomials and Their Applications*. Imperial College Press optimization series. Imperial College Press, 2010.

[39] D. Goldfarb and K. Scheinberg. "On parametric semidefinite programming". In: *Applied Numerical Mathematics* 29.3 (1999), pp. 361–377.

[40] M. V. Ramana, L. Tunçel, and H. Wolkowicz. "Strong duality for semidefinite programming". In: *SIAM Journal on Optimization* 7.3 (1997), pp. 641–662.

[41] R. T. Rockafellar and R. J.B. Wets. *Variational Analysis*. Vol. 317. Springer Science & Business Media, 2009.

[42] R. W. Freund and F. Jarre. "A Sensitivity Result for Semidefinite Programs". In: *Oper. Res. Lett.* 32.2 (2004), pp. 126–132.

[43] T. Kato. *Perturbation theory for linear operators; 2nd ed.* Grundlehren Math. Wiss. Berlin: Springer, 1976. URL: https://cds.cern.ch/record/101545.

[44] Van Parys, B. P.G. "Distributionally robust control and optimization". PhD thesis. ETH Zurich, 2015.

[45] G. R. G. Lanckriet et al. "Learning the Kernel Matrix with Semidefinite Programming". In: *J. Mach. Learn. Res.* 5 (Dec. 2004), pp. 27–72.

[46] A. d'Aspremont et al. "A Direct Formulation for Sparse PCA Using Semidefinite Programming". In: *Advances in Neural Information Processing Systems 17*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. MIT Press, 2005, pp. 41–48.

[47] S. Boyd et al. *Linear Matrix Inequalities in System and Control Theory*. Vol. 15. Studies in Applied Mathematics. Philadelphia, PA: SIAM, June 1994.

[48] S. Prajna, A. Papachristodoulou, and P. A. Parrilo. "Introducing SOSTOOLS: a general purpose sum of squares programming solver". In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 1. Dec. 2002, 741–746 vol.1.

[49] P. J. Goulart and S. Chernyshenko. "Global stability analysis of fluid flows using sum-of-squares". In: *Physica D: Nonlinear Phenomena* 241.6 (2012), pp. 692–704.

[50] J. Lavaei and S. H. Low. "Zero Duality Gap in Optimal Power Flow Problem". In: *IEEE Transactions on Power Systems* 27.1 (Feb. 2012), pp. 92–107.

[51] K. C. Toh and M.J. Todd and R.H. Tütüncü and R. H. Tutuncu. "SDPT3 - a MATLAB software package for semidefinite programming". In: *Optimization Methods and Software* 11 (1998), pp. 545–581.

[52] M. Yamashita, K. Fujisawa, and M. Kojima. "Implementation and evaluation of SDPA 6.0 (Semidefinite Programming Algorithm 6.0)". In: *Optimization Methods and Software* 18.4 (2003), pp. 491–505.

[53] B. Stellato et al. "OSQP: An Operator Splitting Solver for Quadratic Programs". In: *ArXiv e-prints* (2017). arXiv: 1711.08013 [math.OC].

[54] Y. Saad. *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics, 2011.

[55] J. Demmel et al. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Ed. by Zhaojun Bai. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.

[56] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998.

[57] R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK Users' Guide*. Society for Industrial and Applied Mathematics, 1998.

[58] A. 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.

[59] A Stathopoulos and J. R. McCombs. "PRIMME: Preconditioned Iterative Multimethod Eigensolver: Methods and Software Description". In: *ACM Trans. Math. Softw.* 37.2 (Apr. 2010), 21:1–21:30.

[60] V. Hernandez et al. *A survey of software for sparse eigenvalue problems*. Tech. rep. STR-6. Available at http://slepc.upv.es. Universitat Politècnica de València, 2009.

[61] Z. Wen, D. Goldfarb, and W Yin. "Alternating direction augmented Lagrangian methods for semidefinite programming". In: *Mathematical Programming Computation* 2.3 (Dec. 2010), pp. 203–230.

[62] Mario Souto, Joaquim D. Garcia, and Alvaro Veiga. *Exploiting Low-Rank Structure in Semidefinite Programming by Approximate Operator Splitting*. arXiv 1810.05231. 2018.

[63] H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. 2nd. CMS books in mathematics. 2017.

[64] M. Benzi, G.H. Golub, and J. Liesen. "Numerical solution of saddle point problems". In: *Acta Numerica* 14 (2005), pp. 1–137.

[65] D. Orban and M. Arioli. *Iterative Solution of Symmetric Quasi-Definite Linear Systems (SIAM Spotlights)*. 2017.

[66] K. Aishima. "Global Convergence of the Restarted Lanczos and Jacobi—Davidson Methods for Symmetric Eigenvalue Problems". In: *Numer. Math.* 131.3 (2015), pp. 405–423.

[67] C.C. Paige. "Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem". In: *Linear Algebra and its Applications* 34 (1980), pp. 235–258.

[68] G. W. Stewart. *Matrix Algorithms: Volume 1, Basic Decompositions*. Society for Industrial Mathematics, 1998.

[69] A. Lemon, A. M. C. So, and Y. Ye. "Low-Rank Semidefinite Programming: Theory and Applications". In: *Foundations and Trends in Optimization* 2.1-2 (2016), pp. 1–156.

[70] Y. Nakatsukasa. *Sharp error bounds for Ritz vectors and approximate singular vectors*. arXiv 1810.02532. 2018.

[71] P. Giselsson, M. Fält, and S. Boyd. "Line Search for Averaged Operator Iteration". In: *ArXiv e-prints* (2016). arXiv: `1603.06772 [math.OC]`.

[72] Pazy, A. "Asymptotic behavior of contractions in Hilbert space". In: *Israel Journal of Mathematics* 9.2 (1971), pp. 235–240.

[73] J. B. Bailion, R. E. Bruck, and S. Reich. "On the asymptotic behavior of nonexpansive mappings and semigroups in Banach spaces". In: *Houston Journal of Mathematics* 4.1 (1978), pp. 1–9.

[74] H. Qi and D. Sun. "A Quadratically Convergent Newton Method for Computing the Nearest Correlation Matrix". In: *SIAM J. Matrix Anal. Appl.* 28.2 (June 2006), pp. 360–385.

[75] N. J. Higham. "Computing the nearest correlation matrix-a problem from finance". In: *IMA Journal of Numerical Analysis* 22.3 (July 2002), pp. 329–343.

[76] J. Xiong, V. Zolotov, and L. He. "Robust Extraction of Spatial Correlation". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26.4 (Apr. 2007), pp. 619–631.

[77] N. Higham, N. Strabic, and V. Sego. "Restoring Definiteness via Shrinking, with an Application to Correlation Matrices with a Fixed Block". In: *SIAM Review* 58.2 (2016), pp. 245–263.

[78] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, 2017.

[79] N. J. Higham. "Matrix Nearness Problems and Applications". In: *Applications of Matrix Theory* (1989). Ed. by M. Gover and S. Barnett.

[80] C. Davis and W. Kahan. "The Rotation of Eigenvectors by a Perturbation. III". In: *SIAM J. Numer. Anal.* 7.1 (1970), pp. 1–46.

[81] H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. 1st. Springer Publishing Company, Incorporated, 2011.

[82] R. R. Phelps. "Convex Sets and Nearest Points". In: *Proceedings of the American Mathematical Society* 8.4 (1957), pp. 790–797.

[83] R. A. Horn and C. R. Johnson. *Matrix Analysis*. 2nd. New York, NY, USA: Cambridge University Press, 2012.

[84] C. Davis, W. Kahan, and H. F. Weinberger. "Norm-Preserving Dilations and their Applications to Optimal Error Bounds". In: *SIAM J. Numer. Anal.* 19.3 (1982), pp. 445–469.

[85] I. Pak. "Four Questions on Birkhoff Polytope". In: *Annals of Combinatorics* 4.1 (Mar. 2000), pp. 83–90.

[86] P. Knight. "The Sinkhorn-Knopp Algorithm: Convergence and Applications". In: *SIAM Journal on Matrix Analysis and Applications* 30.1 (2008), pp. 261–275.

[87] R. Zass and A. Shashua. "Doubly Stochastic Normalization for Spectral Clustering". In: *Advances in Neural Information Processing Systems 19*. Ed. by B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, 2007, pp. 1569–1576.

[88] P. Milanfar. "A Tour of Modern Image Filtering: New Insights and Methods, Both Practical and Theoretical". In: *IEEE Signal Processing Magazine* 30.1 (Jan. 2013), pp. 106–128.

[89] P. Diggle et al. *Analysis of longitudinal data.* 2nd. Oxford statistical science series; 25. 2002.

[90] S. Rao et al. "A 3D Map of the Human Genome at Kilobase Resolution Reveals Principles of Chromatin Looping". In: *Cell* 159.7 (2014), pp. 1665–1680.

[91] M. Idel. *A review of matrix scaling and Sinkhorn's normal form for matrices and positive maps*. arXiv 1609.06349. 2016. eprint: `1609.06349` (math.RA).

[92] P. A. Knight and D. Ruiz. "A fast algorithm for Matrix Balancing". In: *IMA Journal of Numerical Analysis* 33.3 (2013), pp. 1029–1047.

[93] H. Wang and A. Banerjee. "Bregman Alternating Direction Method of Multipliers". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2816–2824.

[94] W. E. Deming and F. F. Stephan. "On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals are Known". In: *The Annals of Mathematical Statistics* 11.4 (1940), pp. 427–444.

[95] Gurobi Optimization LLC. *Gurobi Optimizer Reference Manual*. 2018. URL: `http://www.gurobi.com`.

[96] H. Perfect and L. Mirsky. "The Distribution of Positive Elements in Doubly-Stochastic Matrices". In: *Journal of the London Mathematical Society* s1-40.1 (1965), pp. 689–698.

[97] E. Yaffe and A. Tanay. "Probabilistic modeling of Hi-C contact maps eliminates systematic biases to characterize global chromosomal architecture". In: *Nature Genetics* 43.11 (2011), p. 1059.

[98] I. Jolliffe. "Principal Component Analysis". In: *International Encyclopedia of Statistical Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1094–1096.

[99] P. T. Boufounos and R. G. Baraniuk. "1-Bit compressive sensing". In: *2008 42nd Annual Conference on Information Sciences and Systems*. 2008, pp. 16–21.

[100] A. R. Conn, M. Gould N., and P. Toint. *Trust-Region Methods*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000.

[101] V. Jeyakumar and G. Y. Li. "Trust-Region problems with linear inequality constraints: exact SDP relaxation, global optimality and robust optimization". In: *Mathematical Programming* 147.1 (2014), pp. 171–206.

[102] N. Gould, D. Orban, and P. Toint. "Numerical methods for large-scale nonlinear optimization". In: *Acta Numerica* 14 (2005), pp. 299–361.

[103] N. Gould, M. Hribar, and J. Nocedal. "On the Solution of Equality Constrained Quadratic Programming Problems Arising in Optimization". In: *SIAM Journal on Scientific Computing* 23.4 (2001), pp. 1376–1395.

[104] J. Martinez. "Local Minimizers of Quadratic Functions on Euclidean Balls and Spheres". In: *SIAM Journal on Optimization* 4.1 (1994), pp. 159–176.

[105] S. Adachi et al. "Solving the Trust-Region Subproblem By a Generalized Eigenvalue Problem". In: *SIAM Journal on Optimization* 27.1 (2017), pp. 269–291.

[106] M. Salahi, A. Taati, and H. Wolkowicz. "Local nonglobal minima for solving large-scale extended Trust-Region Subproblems". In: *Computational Optimization and Applications* 66.2 (2017), pp. 223–244.

[107] N. Gould et al. "Solving the Trust-Region Subproblem using the Lanczos Method". In: *SIAM Journal on Optimization* 9.2 (1999), pp. 504–525.

[108] G. Forsythe and G. H. Golub. "On the Stationary Values of a Second-Degree Polynomial on the Unit Sphere". In: *Journal of the Society for Industrial and Applied Mathematics* 13.4 (1965), pp. 1050–1068.

[109] W. Gander, G. H. Golub, and U. von Matt. "A constrained eigenvalue problem". In: *Linear Algebra and its Applications* 114-115 (1989). Special Issue Dedicated to Alan J. Hoffman, pp. 815–839.

[110] S. Choi, C. Paige, and M. Saunders. "MINRES-QLP: A Krylov Subspace Method for Indefinite or Singular Symmetric Systems". In: *SIAM Journal on Scientific Computing* 33.4 (2011), pp. 1810–1836.

[111] J. R. Silvester. "Determinants of block matrices". In: *The Mathematical Gazette* 84.501 (2000), pp. 460–467.

[112] A. Beck and Y. Vaisbourd. “Globally Solving the Trust Region Subproblem Using Simple First-Order Methods”. In: *SIAM Journal on Optimization* 28.3 (2018), pp. 1951–1967.

[113] K. G. Murty and S. N. Kabadi. “Some NP-complete problems in quadratic and nonlinear programming”. In: *Mathematical Programming* 39.2 (1987), pp. 117–129.

[114] P. E. Gill and W. Murray. “Numerically stable methods for quadratic programming”. In: *Mathematical Programming* 14.1 (1978), pp. 349–372.

[115] A. Wächter and L. T. Biegler. “On the Implementation of an Interior-point Filter Line-search Algorithm for Large-scale Nonlinear Programming”. In: *Mathematical Programming* 106.1 (2006), pp. 25–57.

[116] N. Gould, D. Orban, and P. Toint. “CUTEst: A Constrained and Unconstrained Testing Environment with Safe Threads for Mathematical Optimization”. In: *Comput. Optim. Appl.* 60.3 (2015), pp. 545–557.

[117] H. J. Ferreau et al. “qpOASES: a parametric active-set algorithm for quadratic programming”. In: *Mathematical Programming Computation* 6.4 (2014), pp. 327–363.

[118] M. Journée et al. “Generalized Power Method for Sparse Principal Component Analysis”. In: *J. Mach. Learn. Res.* 11 (2010), pp. 517–553.

[119] XT Yuan and T. Zhang. “Truncated Power Method for Sparse Eigenvalue Problems”. In: *J. Mach. Learn. Res.* 14.1 (2013), pp. 899–925.

[120] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin. “A Modified Principal Component Technique Based on the LASSO”. In: *Journal of Computational and Graphical Statistics* 12.3 (2003), pp. 531–547. URL: `http://www.jstor.org/stable/1391037`.

[121] F. Bach et al. “Optimization with Sparsity-Inducing Penalties”. In: *Foundations and Trends® in Machine Learning* 4.1 (2012), pp. 1–106.

[122] P. Richtárik, M. Takáč, and S. Damla Ahipaşaoğlu. “Alternating Maximization: Unifying Framework for 8 Sparse PCA Formulations and Efficient Parallel Codes”. In: *arXiv e-prints*, arXiv:1212.4137 (2012), arXiv:1212.4137.

[123] Y. Zhang and L. El Ghaoui. “Large-Scale Sparse Principal Component Analysis with Application to Text Data”. In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 532–539.