Minimizing Nonconvex Quadratic Functions Subject to Bound Constraints

by

Hassan Mohy-ud-Din

A thesis submitted to The Johns Hopkins University in conformity with the requirements for the degree of Master of Arts.

Baltimore, Maryland

December, 2014

Abstract

We present an active-set algorithm for finding a local minimizer to a nonconvex bound-constrained quadratic problem. Our algorithm extends the ideas developed by Dostál and Schöberl[1] that is based on the linear conjugate gradient algorithm for (approximately) solving a linear system with a positive-definite coefficient matrix. This is achieved by making two key changes. First, we perform a line search along negative curvature directions when they are encountered in the linear conjugate gradient iteration. Second, we use Lanczos iterations to compute approximations to leftmost eigen-pairs, which is needed to promote convergence to points satisfying certain second-order optimality conditions. Preliminary numerical results show that our method is efficient and robust on nonconvex bound-constrained quadratic problems.

Advisor: Dr. Daniel P. Robinson

Acknowledgments

I would like to thank Allah (God) for all the blessings, making me worthy to achieve whatever little I could.

I have always been passionate about mathematics. In my undergraduate and graduate studies, I became fascinated by Fourier Analysis and Optimization Theory and their ubiquitous place in theoretical and applied sciences. I have always advised my friends, colleagues, and students to sit in an optimization class as it forms an integral part of every discipline. I met Professor Daniel Robinson in a CIS seminar where he gave a talk on *Two-phase matrix splitting methods for BQP and LCP*. After the talk, I approached and requested him to be a part of my Graduate Board Oral (GBO) exam committee. He advised me that I should first sit in his class on Nonlinear Optimization so that I am familiar with the topics of the course. I followed his advice and took the course in Fall 2012. It was an amazing experience as I got to learn about theoretical and practical aspects of optimization, read one of the best references on the subject; *Numerical Optimization by Nocedal and Wright*, and made new friends. The course was exceptionally taught and, no surprises, Professor Robinson won the

Professor Joel Dean Award for teaching. I did really well in the course (got Grade A) and also managed to pass my GBO exam in Spring 2013. During the course, I met a good friend Yueling Loh who was assisting the course and was a PhD student of Professor Robinson. I expressed my desire to work on a project with Professor Robinson and discussed the prospects of it. She strongly recommended me to discuss with Professor Robinson.

As a PhD student in the Department of Electrical and Computer Engineering (ECE), I was earning a double masters degree in Applied Mathematics and Statistics (AMS). Almost all my colleagues would pass ten courses (non-thesis option) to earn the degree so that it does not take away much of their time of PhD research. I chose the difficult path and went for a masters degree (thesis option). The whole idea was to learn to do research in optimization theory and, hopefully, publish a paper. I discussed with Professor Robinson who, very kindly, agreed to supervise me. He shared seven potential projects and I said 'wow' to each one of them. The project we converged on culminated in this thesis. He also accepted my suggestion to conduct a weekly journal club where we would discuss selected papers. The whole process of research and collaboration was extremely fruitful and engaging as I got to learn the most important component of quality research i.e. asking the right questions. He has always been patient, helpful, and cooperative. His geometric thinking is exceptional. He is well read and never short of ideas. I have thoroughly enjoyed working with him and look forward to extend this collaboration on many more projects in future. I would like to thank him for all the guidance and support he extended to me.

I would also like to thank Dr. Rachael Tappenden for collaborating on this project, Yueling Loh for being a good friend, Kristin Bechtel for patiently listening to my concerns during the administrative process, and the entire faculty, staff, fellow colleagues and students for making my stay worthwhile. I would specially like to thank Professor Arman Rahmim, my PhD supervisor, for allowing me to pursue a research degree in AMS, Professor Laurent Younes for giving me an opportunity to teach a month-long course on *MATLAB with Applications* at AMS department, and Dr. Asadullah who has been a source of constant support during my stay in USA. I have always consulted him for his valuable advice in almost every matter. Finally, I would to like to thank ECE department for bestowing me with a PhD fellowship that allowed me to become a part of Johns Hopkins University and pursue my interests.

Thank You ...

Dedication

I have always believed that the biggest seat of learning is home. I owe almost everything to my family. My parents, Mr. Abdul Hafeez Siddiqui and Ms. Saleema Hafeez, and my sisters, Ms. Aroosa Kiran Fatima, Ms. Zeenat Ayesha Adnan, and Dr. Sabahat Zahra Siddiqui for being a great source of energy and joy. They managed to provide an intellectual environment which was multi-dimensional. I developed a penchant for poetry, philosophy, theology, sports, music and literature. Nature was more stimulating then technology. Family relationships made me realize that whatever I do effects everyone. It infused confidence, self-control and simplicity. I learned the power of words and speech which enticed me to pursue a career in academics and research. I was advised to be steady in tough times and be grounded in prosperity. I was taught to be vulnerable and robust. The constitution was based on ethics, virtue, and morality. Thank You so much for everything. A little token for your life's investment.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Quadratic problems form an important class of nonlinear optimization problems. They involve the minimization of a quadratic objective function subject to linear constraints on optimization variables. Quadratic programs find applications in a wide variety of disciplines, e.g., portfolio optimization,[2] machine learning,[3,4] computer vision,[5] signal and image processing,[6,7] optimal control,[8] optimization with partial differential equation (PDE) constraints,[9] shape theory,[10] sequential quadratic programming,[11,12] linearly constrained Lagrangian methods,[13] and many more.

An important subclass of constrained quadratic problems are those with only simple bounds on the optimization variables. Such bound-constrained quadratic problems (BCQPs), which are the focus of this thesis, are important in their own right with applications to augmented Lagrangian methods,[14,15] PDE-constrained optimization problems,[16] machine learning,[3,4] and beyond.

In this section we lay the foundation for the thesis by first introducing in Section 1.1 the basic optimization problem that we study, i.e., the BCQP. In Section 1.2 we then give the contributions of this thesis and explain why they are important. In Section 1.3 we summarize the notation used throughout, while in Section 1.4 we summarize key definitions. Finally, we finish in Section 1.5 by giving the optimality conditions for the BCQP that we study.

1.1 The problem formulation

In this thesis we focus on a special optimization problem called a *bound constrained quadratic problem* (BCQP). A BCQP is found in many image processing and medical imaging applications with upper and lower bounds on the parameter(s) of interest (e.g., image intensity). Specifically, the BCQP may be written as

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & q(x) := \frac{1}{2}x^T Hx - c^T x \\ \text{subject to} \quad & l \le x \le u, \end{aligned} \tag{1.1}$$

where q is the quadratic objective function, H is a given symmetric $n \times n$ Hessian matrix, $c \in \mathbb{R}^n$ is a given column vector, and $l, u \in \mathbb{R}^n$ are column vectors such that $l < u$ componentwise; components of l/u are allowed to have $-\infty/\infty$ values. We note that there is no loss in generality with assuming that $l < u$ since if $l_i = u_i$ for some i, then any solution x^* to (1.1) must also satisfy $x_i^* = l_i$. Consequently, the ith variable may be considered as fixed and need not be a variable in the optimization problem.

The tractability of solving the BCQP (1.1) depends on the properties of q, in particular those of H. If H is positive semidefinite, (1.1) is a convex optimization problem, which means that any local minimizer is also a global minimizer. In the special case of $H = 0$, the convex BCQP reduces to a linear program. If H is indefinite, problem (1.1) is nonconvex so that finding the global solution is NP-hard.[17,18] In particular, a local minimizer is not necessarily a global minimizer.

1.2 The contributions of this thesis

This thesis presents a new algorithm for solving BCQPs. The core components of the method are similar to those presented by Dostál and Schöberl.[1] In particular, a prominent role is served by the linear conjugate gradient (CG) algorithm for approximately solving a linear system with a positive-definite coefficient matrix. Their method has proved effective on a diverse range of applications that require the solution to *convex* BCQPs. Unlike their method, our proposed algorithm is applicable to nonconvex BCQPs. To tackle the challenges associated with non-convexity, we use a special procedure when the linear CG iteration encounters negative curvature, and include a mechanism (based on the Lanczos iteration for approximating eigenvalues) that incorporates the leftmost eigen-vector when appropriate. Our numerical results indicate that our method is efficient and robust.

1.3 Notation

The notation used is standard in optimization and gathered here for reference.

- The optimization variables are denoted as the vector $x \in \mathbb{R}^n$.
- The quadratic objective function (see (1.1)) is denoted by $q : \mathbb{R}^n \to \mathbb{R}$.
- The gradient of q is denoted by $g := \nabla_x q$, i.e., $g(x) = Hx - c$.
- Given the kth iterate x_k, we use $q_k := q(x_k)$ and $g_k := g(x_k)$ to denote the objective function and its first derivative, respectively, evaluated at x_k.
- The search direction at iterate x_k is represented by s_k.
- For any matrix M, we use $M \succ 0$ to mean that M is positive definite, $M \succeq 0$ to mean that M is positive semidefinite, $M \not\succ 0$ to mean that M is not positive definite, and $M \not\succeq 0$ to mean that M is not positive semidefinite.
- The index set of all variables is $\mathcal{N} := \{1, 2, \ldots n\}$.
- The ith component of a vector v will sometimes be written as $[v]_i$.
- The subvector of v with components given by the index set $\mathcal{S}$ is given by $[v]_{\mathcal{S}}$.
- The vector two-norm or induced matrix norm is denoted by $\|.\|$.
- The minimum eigenvalue of a matrix A is represented by $\lambda_{min}(A)$.
- Given any vector y, we let $[y]^+ := \max(y, 0)$ and $[y]^- := \min(y, 0)$.

1.4 Definitions

In this section, we define various optimization terminology used in the thesis. Our first definition concerns the feasible set for problem (1.1).

Definition 1.4.1 (Feasible set and feasible point) *The feasible set, Ω, for problem* (1.1) *is defined as*

$$\Omega := \{x \in \mathbb{R}^n \ \mid \ l \leq x \leq u\},$$

which is closed and convex. We say that x is a feasible point if and only if $x \in \Omega$.

We may associate with any feasible point the active set.

Definition 1.4.2 (Active set) *The active set at any $x \in \Omega$, denoted as $\mathcal{A}(x)$, is*

$$\mathcal{A}(x) := \{i \in \mathcal{N} \ \mid \ x_i = l_i \ \ or \ \ x_i = u_i\}.$$

The active set satisfies the identity $\mathcal{A}(x) = \underline{\mathcal{A}}(x) \cup \bar{\mathcal{A}}(x)$, where

$$\underline{\mathcal{A}}(x) := \{i \in \mathcal{N} \ \mid \ x_i = l_i\} \quad and \quad \bar{\mathcal{A}}(x) := \{i \in \mathcal{N} \ \mid \ x_i = u_i\},$$

are disjoint (since $l < u$) subsets of $\mathcal{A}(x)$.

The variables that are not in the active set are considered to be in the free set.

Definition 1.4.3 (Free set) *The free set of variables at x, denoted as $\mathcal{F}(x)$, is*

$$\mathcal{F}(x) := \mathcal{N} \setminus \mathcal{A}(x).$$

We use the concept of the free and chopped gradient as first introduced by Dostál and Schöberl[1] and given in the next definition. In the next section, we show how these objects may be used to characterize first-order optimality.

Definition 1.4.4 (Free gradient and chopped gradient) *At a pint x, the free gradient, $\varphi(x)$, and chopped gradient, $\beta(x)$, are defined as*

$$[\varphi(x)]_i := \begin{cases} [g(x)]_i & \textit{if } i \in \mathcal{F}(x); \\ 0 & \textit{if } i \in \mathcal{A}(x); \end{cases} \quad \textit{and} \quad [\beta(x)]_i := \begin{cases} 0 & \textit{if } i \in \mathcal{F}(x); \\ [g(x)]_i^- & \textit{if } i \in \underline{\mathcal{A}}(x); \\ [g(x)]_i^+ & \textit{if } i \in \bar{\mathcal{A}}(x). \end{cases}$$

1.5 Optimality conditions

In this section, we present first- and second-order optimality conditions for the BCQP (1.1). We begin with the first-order necessary conditions.

Result 1.5.1 (first-order necessary optimality conditions) *If x^* is a minimizer for problem* (1.1)*, then with $g(x^*) \equiv [g(x^*)]^+ + [g(x^*)]^-$ it follows that*

$$x^* \in \Omega, \quad [g(x^*)]^+ \cdot (x^* - l) = 0, \quad \textit{and} \quad [g(x^*)]^- \cdot (u - x^*) = 0,$$

which are equivalent to

$$x^* \in \Omega \quad \textit{and} \quad \nu(x^*) := \varphi(x^*) + \beta(x^*) = 0. \tag{1.2}$$

The conditions in (1.2) naturally lead us to the idea of a first-order KKT point.

Definition 1.5.1 (first-order KKT point) *We say that a vector x is a first-order KKT point if and only if $x \in \Omega$ and $\nu(x) = 0$ with ν is defined in* (1.2).

To state second-order optimality conditions, we need to define the sets of strongly active constraints at a first-order KKT point x given by

$$\underline{\mathcal{A}}^+(x) := \{i \in \underline{\mathcal{A}}(x) : [g(x)]_i^+ > 0\} \quad \text{and} \quad \bar{\mathcal{A}}^+(x) := \{i \in \bar{\mathcal{A}}(x) : [g(x)]_i^- < 0\},$$

with their union being defined as

$$\mathcal{A}^+(x) := \underline{\mathcal{A}}^+(x) \cup \bar{\mathcal{A}}^+(x).$$

We also define the sets of weakly active constraints at a first-order KKT point x as

$$\underline{\mathcal{A}}^0(x) := \{i \in \underline{\mathcal{A}}(x) : [g(x)]_i^+ = 0\} \quad \text{and} \quad \bar{\mathcal{A}}^0(x) := \{i \in \bar{\mathcal{A}}(x) : [g(x)]_i^- = 0\},$$

with their union being defined as

$$\mathcal{A}^0(x) := \underline{\mathcal{A}}^0(x) \cup \bar{\mathcal{A}}^0(x).$$

We may now state the second-order necessary and sufficient optimality conditions.

Result 1.5.2 (second-order necessary and sufficient optimality conditions) *The vector x^* is a minimizer for problem* (1.1) *if and only if*

$$x^* \in \Omega, \quad \nu(x^*) = 0, \quad \textit{and} \quad p^T H p \ge 0 \;\; \textit{for all}\; p \in \mathcal{C}(x^*),$$

where the critical cone $\mathcal{C}(x^*)$ is defined using $\mathcal{A}^+ = \mathcal{A}^+(x^*)$ and $\mathcal{A}^0 = \mathcal{A}^0(x^*)$ as

$$\mathcal{C}(x^*) := \{p \in \mathbb{R}^n : [p]_{\mathcal{A}^+} = 0 \;\; \textit{and} \;\; [p]_{\mathcal{A}^0} \ge 0\}.$$

The second-order optimality condition given in Result 1.5.2 are generally not practical since verifying that H is positive semi-definite over the convex cone $\mathcal{C}(x^*)$ is difficult. Instead, we will use a practical condition that verifies that the curvature condition holds over a smaller set, but for which verification is much easier. This leads to the definition of what we will call a second-order point. We comment that the goal of the algorithm we propose in Chapter 4 is to generate a sequence of iterates $\{x_k\}$ that has a second-order point as a limit point.

Definition 1.5.2 (A second-order point) *We say that x is a second-order point if and only if with $\mathcal{F} = \mathcal{F}(x)$ it follows that $x \in \Omega$, $\nu(x) = 0$, and $H_{\mathcal{F}\mathcal{F}} \succeq 0$, where $H_{\mathcal{F}\mathcal{F}}$ is the submatrix of H consisting of the rows and columns indexed by $\mathcal{F}$.*

Note that the condition $H_{\mathcal{F}\mathcal{F}} \succeq 0$ is equivalent to saying that $p^T Hp \geq 0$ for all $p \in \mathcal{C}^S(x) := \{p \in \mathbb{R}^n : [p]_{\mathcal{A}(x)} = 0\}$. Since, in general, we only know that $\mathcal{C}^S(x) \subseteq \mathcal{C}(x)$, the conditions in Definition 1.5.2 are not, in general, sufficient for ensuring that a point is a minimizer. On the upside, however, the condition $H_{\mathcal{F}\mathcal{F}} \succeq 0$ can (in exact arithmetic) be verified computationally. In this thesis, we search for second-order points, and comment that most algorithms in the literature merely ensure convergence to first-order KKT points.

Chapter 2

Minimizing Quadratic Functions Subject to Bound-Constraints

Many methods have been proposed to minimize quadratic objective functions subject to linear constraints. This class of problems is more general than the BCQPs that are the focus of this thesis, and the associated algorithms are necessarily more complex. In addition to being more complex, the tools they use are more limited since the general linear constraints are more complicated than simple bounds on the optimization variables. Since the focus of this thesis is the BCQP, in this chapter we describe the three main classes of algorithms used to solve BCQPs: (i) traditional active-set methods (Section 2.1); (ii) gradient projection methods (Section 2.2); and (iii) interior-point methods (Section 2.3).

2.1 Traditional active-set methods

Traditional *primal* active-set methods[19–21] solve (1.1) by generating *primal* feasible iterates. During each iteration, a set of variables—called the active variables or the active set—are forced to be equal to either their lower or upper bound, while the remaining variables (the free variables) are optimized. In particular, the optimal free variables may be obtained by solving a system of equations with coefficient matrix $H_{\mathcal{F}\mathcal{F}}$. If any of the optimal free variables violates their constraint bounds, then the one that is violated "first" becomes a member of the active-set. This process may be continued until the minimizer over the set of free variables is feasible.[22,23] If the minimizer over this partition of active and free variables is not optimal for problem (1.1), then an active variable is chosen (by inspecting all of their dual/Lagrange multiplier values) to be moved to the set of free variables, and the entire process is then repeated. The key attribute that drives convergence to a solution is that the objective function q is strictly decreased each time the sets of active/free variables change. Since the total number of partitions of the variables is finite, this means that eventually this process must terminate, and it turns out (modulo some details and assumptions) that the final iterate solves (1.1). We also comment that it is only at the solution that the dual variables (i.e., the Lagrange multipliers) are dual feasible.

Let's now consider one major assumption that was being made during the last paragraph. In order to solve the reduced system of equations we would need to know that the matrix $H_{\mathcal{F}\mathcal{F}}$ is indeed nonsingular. This is guaranteed if $H \succ 0$, but not if H

is indefinite and possibly singular. Thus, it is clear that modifications must be made to traditional primal active-set methods to ensure their applicability to nonconvex problems. Here, we do not give any more details in this direction, but simply point the reader to Nocedal and Wright [24, Chapter 16] to learn more.

Traditional primal active-set methods benefit from warm starts (i.e., the ability to efficiently use an accurate estimate of a solution) and have the advantage of computing accurate solutions when $H_{\mathcal{F}\mathcal{F}}$ is ill-conditioned. On the negative side, traditional active-set methods[25–27] are not suited for large-scale problems for two main reasons. First, they add/remove one variable at a time from the active and free sets of variables. Since a linear system of equations must be solved each time these sets change, there may be, in the worse case, a combinatorial number of linear systems solved. Although computational savings may be realized through matrix factorization updating strategies, this fact still limits their use for very large-scale problems. Second, the auxiliary problems need to be solved accurately, which precludes the use of iterative methods for solving linear systems of equations such as CG or MINRES.

Most traditional *dual* active-methods are restricted to strictly convex problems, although, for example, the algorithm introduced by Golfarb and Idnani[28] for strictly convex problems was adapted by Boland[29] to handle merely convex problems. In contrast to their primal counterpart, traditional dual active-set methods first compute a dual feasible point and then maintain dual feasibility throughout; it is only the terminal point that is primal feasible. Since the main contribution of this thesis is that of a new algorithm for solving nonconvex BCQPs, we do not discuss dual active-set methods further since they are restricted to convex problems.

2.2 Gradient projection methods

A feature of traditional active-set methods that limits their application to very large-scale problems is the one-in and one-out nature in which the active set of variables evolves. To overcome this weakness, gradient projection methods[30–35] adjust the set of active variables based on minimizing the objective function along the path defined by projecting the ray defined along the negative gradient direction, i.e., along the projected gradient path. This strategy, therefore, does not place any limit on the number of variables that may leave or enter the active set of variables during each iteration. The Goldstein-Levitin-Polyak[30,31] gradient projection method, though suitable for large-scale problems, has the tendency to "zigzag" based on different variables entering and leaving the active-set (at least on degenerate problems) and exhibits a slow convergence rate even once the optimal active-set has been identified.[32] To overcome this latter weakness, Bertsekas[33] proposed a Newton-like gradient projection method that exhibits a superlinear rate of convergence once the active set associated with an optimal solution has be identified.

Perhaps the most popular approaches for ensuring fast local convergence as well as quick evolution of the active set of variables are two-phase methods. One of the first such methods was proposed by Dembo and Tulowitzki,[32] in which they suggested the use of the CG method to solve a reduced system of equations with matrix $H_{\mathcal{F}\mathcal{F}}$, where the free variables in the set $\mathcal{F}$ are the complement of those variables deemed active by searching along the projected gradient path. The authors are able to utilize the finite termination property of CG to conclude finite termination of the overall method, at least on nondegenerate problems. Yang and Tolle[34] extended these ideas and proved finite termination even for degenerate problems. A perceived downside of this approach was that for starting points far from an optimal solution, it may take many iterations before identifying the active set of variables at the solution. Moré and Toraldo[35] studied a similar approach that used only the gradient projection steps until a "suitable" active set was found or no further "substantial" progress toward a solution was measured. At this point, CG was used to approximately solve the reduced system of linear equations, which was equivalent to approximately minimizing a reduced unconstrained quadratic subproblem. This approach achieved convergence for strictly convex problems and finite termination for nondegenerate problems. Importantly, their numerical experiments also clearly showed that the gradient project steps very quickly identified the variables that were active at the solution *provided the Hessian matrices were well condition.* In fact, there was a rather strong connection between the conditioning of the Hessian matrix and the speed at which the active variables at the solution were discovered by the projected gradient steps.

To overcome the poor performance of the method by Moré and Toraldo[35] on ill-conditioned problems, Robinson et. al[36] proposed a two-phase matrix-splitting method that also utilized CG to optimize over a subspace defined from an active-set of variables. In their method, a gradient projection step was replaced by a more general matrix splitting iteration for which the projected gradient, projected Jacobi, and projected Gauss-Seidel iterations are special cases. By using these more powerful matrix splitting iterations—in place of projected gradient iterations—to perform the active-variable identification, they were able to show better performance in comparison to the method by Moré and Toraldo as the condition number of the Hessian matrix became even moderate in size. On the downside, the method by Robinson et. al is in general slightly more expensive since, even on sparse problems, computing a projected Gauss-Seidel step is modestly more expensive than performing a projected gradient step. It is also important to mention that their method was, in theory, applicable to nonconvex problems, but in that case the only known practical splitting iteration is equivalent to the projected gradient step as was already commonly used.

Robinson et al.[36] also numerically compared their algorithm to a method by Dostál and Schöberl,[1] which was designed to solve convex BCQPs. The Dostál and Schöberl method was modestly inferior on well-condition problems, but became equally efficient on moderately ill-conditioned problems, and eventually became superior on ill-conditioned problems. Their method is not, in spirit, that different from the method by Moré and Toraldo described above. The biggest and most important difference is that they developed conditions that determined how accurately each reduced linear system should be solved using CG. This dynamic nature was, in fact, the reason why the method by Robinson et. al was inferior on ill-conditioned problems: their method wasted great computational effort in approximately solving an ill-conditioned linear system using CG (a task sure to demand an exorbitant cost without a suitable preconditioner), whereas the adaptive conditions used by Dostál and Schöberl more quickly realized when the current active-set was incorrect, and moved on.

In this thesis we take the ideas introduced by Dostál and Schöberl for convex BCQPs, and extend them to the nonconvex setting. In doing so, we obtain an efficient algorithm for solving nonconvex and convex BCQPs on well- and ill-conditioned problems, and in addition promote convergence to second-order optimality points.

2.3 Interior-point methods

Unlike active-set methods, interior-point methods[37–39] generate iterates in the interior of the feasible region. The generated iterates follow a continuous path to the optimal solution that can be parameterized by a positive parameter. The parameter regularizes the linear system of equations that is solved in each iteration. Interior-point methods make rapid progress to the optimal solution since a *single* linear system solve is required per iteration. This feature makes interior-point methods a popular choice for large-scale problems. Probably their biggest disadvantage is that they do not—in contrast to active-set methods—efficiently use a good initial estimate of a solution. This attribute is a consequence of the fact that their efficiency usually requires an initial iterate that is well centered in the interior of the feasible region.

Ye and Tse [40] proposed an interior ellipsoid method for convex problems. During each iteration, the objective function is minimized oven an ellipsoidal region whose size depends on the distance between the current strictly feasible point and the boundary of the feasible region. They proved for convex problems that if the sequence of iterates converges, then the limit point must be optimal, while for strictly convex problems the sequence of iterates converges to the unique optimal solution.[41] Coleman and Liu[42] proposed an interior-point Newton method with strong convergence properties in the sense that the generated sequence of iterates converged to a point that satisfied certain second-order conditions. Moreover, if the limit point satisfied second-order sufficiency conditions, then the local rate of convergence was 2-step quadratic. A closely related class of methods consists of barrier methods, which avoid explicit inequality constraints by penalizing points close to the boundary of the feasible region.[43,44] This modification results in equality-constrained subproblems that must be solved (approximately) using a Newton-based method.[45] Finally, perhaps the most successful methods are primal-dual interior-point methods,[37,46,47] which exhibit superlinear asymptotic convergence and can accommodate infeasible starting points. We emphasize that every method described in this section suffers—in contrast to active-set methods—from the inability to efficiently use a good initial estimate of a solution, which is very important in certain applications such as in optimal control.[48]

Chapter 3

The Dostál and Schöberl Algorithm for Convex Problems

In this chapter, we expound upon the Dostál and Schöberl algorithm[1] (henceforth called the DS algorithm) because it forms the basis of our proposed algorithm for solving *nonconvex* BCQPs. In the process, we generalize the presentation of their original method since we incorporate the lower and upper bound constraints present in the optimization problem (1.1) (their formulation only considered lower bounds).

3.1 An overview of the method

The DS algorithm is built around the use of CG to explore the reduced space defined over the set of free variables, i.e., the set complementary to an active set of variables. Unlike the two-phase methods described in Section 2.2, they developed an adaptive stopping condition to be used for judging when the CG iterations should be terminated. As we will see, the stopping condition is based on quantities related to the KKT conditions (see (1.2)). The *CG step* is critical and is the focus of Section 3.2.

As with all active-set methods, the DS algorithm has a strategy for adding variables to the active-set when necessary. This is done by monitoring the iterates computed by CG and taking action as soon as an iterate becomes infeasible. As soon as a violation is detected, the violated variable is added to the current active set. In addition, a single projected gradient step is performed to ensure that sufficient progress is achieved. These two actions taken in tandem results in the *expansion step*, which is the topic of Section 3.3.

The expansion step gets triggered if any CG step violates the bounds on the variables, i.e., become infeasible. If this situation never happens for the current active set, then the CG iterations will converge to the unique solution of the reduced linear system of equations. (Recall that algorithm DS is only appropriate for strictly convex problems, which means that the reduced linear system of equations will always be defined by a positive-define matrix). This is equivalent to saying that the iterates will converge to a minimizer of q over the space of free variables, i.e., those not in the active set. If the limiting point is not optimal, then eventually a decision has to be made to free-up at least one of the active variables; this is called the *proportioning step* and is considered in Section 3.4.

3.2 The conjugate gradient step

An important aspect of algorithm DS is the ability to use CG on the subspace of free variables. Let $\widetilde{x} \in \mathbb{R}^n$ and consider using linear CG to solve the problem

$$\begin{aligned} \underset{p \in \mathbb{R}^n}{\text{minimize}} \quad & \widetilde{q}(p) = \tfrac{1}{2}(\widetilde{x}+p)^T H(\widetilde{x}+p) - c^T(\widetilde{x}+p) \\ \text{subject to} \quad & I_{\mathcal{A}} p = [p]_{\mathcal{A}} = 0, \end{aligned} \tag{3.1}$$

where $I_{\mathcal{A}} = I_{\mathcal{A}(\widetilde{x})}$ contains the rows of the $n \times n$ identity matrix that correspond to the indices in $\mathcal{A}(\widetilde{x})$. We may then easily see that the columns of the matrix $I_{\mathcal{F}}^T$ with $I_{\mathcal{F}} = I_{\mathcal{F}(\widetilde{x})}$ form a basis for the null space of $I_{\mathcal{A}}$. This leads to the following algorithm, which is based on [49, Algorithm 5.4.2].

Algorithm 1 The CG algorithm for the reduced space problem (3.1).

```
1: Define $\mathcal{A} = \mathcal{A}(\widetilde{x})$ and $\mathcal{F} = \mathcal{F}(\widetilde{x})$.
2: Set $p_0 = 0$, $g_0 \leftarrow \nabla\widetilde{q}(p_0) \equiv H(\widetilde{x}+p_0) - c$, $v_0 \leftarrow ([g_0]_{\mathcal{F}}, 0) \equiv \varphi(\widetilde{x})$, and $s_0 \leftarrow v_0$.
3: for $j = 0, 1, 2, \ldots$ do
4:     Set $\alpha_j \leftarrow (g_j^T v_j)/(s_j^T H s_j)$.
5:     Set $p_{j+1} \leftarrow p_j - \alpha_j s_j$.
6:     Set $g_{j+1} \leftarrow g_j - \alpha_j H s_j \equiv \nabla\widetilde{q}(p_{j+1})$.
7:     Set $v_{j+1} \leftarrow ([g_{j+1}]_{\mathcal{F}}, 0) \equiv \varphi(\widetilde{x}+p_{j+1})$.
8:     Set $\gamma_j \leftarrow (g_{j+1}^T v_{j+1})/(g_j^T v_j)$.
9:     Set $s_{j+1} \leftarrow v_{j+1} - \gamma_j s_j$.
10: end for
```

Algorithm 1 computes a sequence $\{p_j\}$ that converges to the unique solution of problem (3.1). If we now define $x_j := \widetilde{x} + p_j$, we may adjust the linear CG algorithm so that it generates iterates $\{x_j\}$ that converge to the solution of the subproblem in terms of x instead of p; this results in Algorithm 2.

Algorithm 2 The CG algorithm for the reduced space problem (3.1) in terms of x.

```
1: Define $\mathcal{A} = \mathcal{A}(\widetilde{x})$ and $\mathcal{F} = \mathcal{F}(\widetilde{x})$.
2: Set $x_0 \leftarrow \widetilde{x}$, $g_0 \leftarrow Hx_0 - c$, $s_0 \leftarrow \varphi(x_0)$.
3: for $j = 0, 1, 2, \ldots$ do
4:     Set $\alpha_j \leftarrow \big(g_j^T \varphi(x_j)\big)/(s_j^T H s_j)$.
5:     Set $x_{j+1} \leftarrow x_j - \alpha_j s_j$.
6:     Set $g_{j+1} \leftarrow g_j - \alpha_j H s_j$.
7:     Set $\gamma_j \leftarrow \big(g_{j+1}^T \varphi(x_{j+1})\big)/\big(g_j^T \varphi(x_j)\big)$.
8:     Set $s_{j+1} \leftarrow \varphi(x_{j+1}) - \gamma_j s_j$.
9: end for
```

As mentioned in the overview Section 3.1, steps generated from Algorithm 2, i.e., steps that reduce the quadratic objective function over the space of free variables, form the basis of the DS algorithm. Since they ultimately minimize q in the space of free variables, they have the effect, if iterated forever, of driving $\{\varphi(x_j)\}$ to zero (recall Definition (1.4.4)). Thus, we only continue the computation of CG when

$$\|\beta(x_k)\|^2 \leq \Gamma \tilde{\varphi}(x_k)^T \varphi(x_k) \tag{3.2}$$

is satisfied for some $\Gamma > 0$, where the *reduced* free gradient $\tilde{\varphi}$ is defined as

$$\tilde{\varphi}(x) := \min \left\{ \frac{x-l}{\bar{\alpha}}, \frac{u-x}{\bar{\alpha}}, \varphi(x) \right\} \tag{3.3}$$

for some fixed $\bar{\alpha} \in (0, \|H\|^{-1}]$. When (3.2) holds, this indicates that substantial progress from continuing CG is still possible.

3.3 The expansion step

It is possible that the CG Algorithm 2 computes a direction s_j such that the vector $x_j - \alpha_j s_j$ in Step 5 violates the bound constraints in problem (1.1). In this situation, we first take a step along the CG direction s_j to the nearest blocking constraint(s) as described in Step 4 of Algorithm 3; this adds at least one variable to the current active set. This is then immediately followed by a gradient projection step in the space of free variables (see Step 6). The projected gradient step allows for potentially many new variables to be added to the current active-set. Since these steps only expand/increase the size of the active set, the overall resulting step is called an expansion step. In Algorithm 3, the projection operator P_Ω is defined componentwise by

$$[P_\Omega(x)]_i := \max\{l_i, \min\{x_i, u_i\}\}, \tag{3.4}$$

where the definition of the feasible set Ω is stated as Definition 1.4.1.

Algorithm 3 The expansion step.

```
1: available constant: $\bar{\alpha} \in (0, \|H\|^{-1}]$.
2: input: vector $s_j$ from Algorithm 2 that satisfies $(x_j - \alpha_j s_j) \notin \Omega$.
3: Set $\alpha_{feas} \leftarrow \max\{\alpha : x_j - \alpha s_j \in \Omega\}$.
4: Set $x_{j+\frac{1}{2}} \leftarrow x_j - \alpha_{feas} s_j$.
5: Set $g_{j+\frac{1}{2}} \leftarrow g_j - \alpha_{feas} H s_j$.
6: Set $x_{j+1} \leftarrow P_\Omega(x_{j+\frac{1}{2}} - \bar{\alpha}\varphi(x_{j+\frac{1}{2}}))$.
7: Set $g_{j+1} \leftarrow H x_{j+1} - c$.
8: Set $s_{j+1} \leftarrow \varphi(x_{j+1})$.
```

3.4 The proportioning step

If (3.2) does not hold, it indicates that a Lagrange multiplier estimate associated with at least one active variable has a substantially wrong sign in comparison to the size of the free gradient. It is thus prudent to step off of such a variable, i.e., to transfer that variable to the set of free variables. Therefore, it makes sense to minimize the objective function q along the direction $s_j = \beta(x_j)$ (see Step 2 of Algorithm 4) while maintaining feasibility. This computation results in the point given in Step 6. Updates of this type first release variables from their bounds, and then potentially fix some. In general, nothing can be said about the net difference in activities.

Algorithm 4 The proportioning step.

```
1: available constant: $\bar{\alpha} \in (0, \|H\|^{-1}]$.
2: Set $s_j \leftarrow \beta(x_j)$.
3: Set
       $$\alpha_{min} \leftarrow \begin{cases} (s_j^T g_j)/(s_j^T H s_j) & \text{if } s_j^T H s_j > 0; \\ \infty & \text{otherwise.} \end{cases}$$
4: Set $\alpha_{feas} \leftarrow \max\{\alpha : x_j - \alpha s_j \in \Omega\}$.
5: Set $\alpha_j \leftarrow \min\{\alpha_{min}, \alpha_{feas}\}$.
6: Set $x_{j+1} \leftarrow x_j - \alpha_j s_j$.
7: Set $g_{j+1} \leftarrow g_j - \alpha_j H s_j$.
8: Set $s_{j+1} \leftarrow \varphi(x_{j+1})$.
```

3.5 The complete algorithm

The DS algorithm for solving problem (1.1) when $H \succ 0$ is given by Algorithm 5.

Algorithm 5 The DS Algorithm for strictly convex BCQP.

```
1: input: $x_0 \in [l, u]$ with $l < u$ and $\tau_{stop} > 0$.
2: Choose $\Gamma > 0$ and $\bar{\alpha} \in (0, \|H\|^{-1}]$.
3: Set $g_0 \leftarrow Hx_0 - c$ and $s_0 \leftarrow \varphi(x_0)$.
4: for $k = 0, 1, 2, \ldots$ do
5:     if $\|\nu(x_k)\| \leq \tau_{stop}$ then
6:         return an approximate first-order KKT point $x_k$.
7:     end if
8:     if (3.2) is satisfied then
9:         Set $\alpha_{cg} \leftarrow \big(g_k^T \varphi(x_k)\big)/(s_k^T H s_k)$.
10:        Set $\alpha_{feas} \leftarrow \max\{\alpha : x_k - \alpha s_k \in \Omega\}$.
11:        if $\alpha_{cg} \leq \alpha_{feas}$ then                    ▷ conjugate gradient step
12:            Set $x_{k+1} \leftarrow x_k - \alpha_{cg} s_k$.
13:            Set $g_{k+1} \leftarrow g_k - \alpha_k H s_k$.
14:            Set $\beta_k \leftarrow \big(g_{k+1}^T \varphi(x_{k+1})\big)/\big(g_k^T \varphi(x_k)\big)$.
15:            Set $s_{k+1} \leftarrow \varphi(x_{k+1}) - \beta_k s_k$.
16:        else                                                        ▷ expansion step
17:            Set $x_{k+1/2} \leftarrow x_k - \alpha_{feas} s_k$.
18:            Set $g_{k+1/2} \leftarrow g_k - \alpha_{feas} H s_k$.
19:            Set $x_{k+1} \leftarrow P_\Omega(x_{k+1/2} - \bar{\alpha}\varphi(x_{k+1/2}))$.
20:            Set $g_{k+1} \leftarrow Hx_{k+1} - c$.
21:            Set $s_{k+1} \leftarrow \varphi(x_{k+1})$.
22:        end if
23:    else                                                            ▷ proportioning step
24:        Set $s_k \leftarrow \beta(x_k)$.
25:        Set
               $$\alpha_{min} \leftarrow \begin{cases} (s_k^T g_k)/(s_k^T H s_k) & \text{if } s_k^T H s_k > 0; \\ \infty & \text{otherwise.} \end{cases}$$
26:        Set $\alpha_{feas} \leftarrow \max\{\alpha : x_k - \alpha s_k \in \Omega\}$.
27:        Set $\alpha_k \leftarrow \min\{\alpha_{min}, \alpha_{feas}\}$.
28:        Set $x_{k+1} \leftarrow x_k - \alpha_k s_k$.
29:        Set $g_{k+1} \leftarrow g_k - \alpha_k H s_k$.
30:        Set $s_{k+1} \leftarrow \varphi(x_{k+1})$.
31:    end if
32: end for
```

Chapter 4

A New Algorithm for Nonconvex Problems

In the previous section we elaborated on the Dostál and Schöberl algorithm (called the DS algorithm) for solving *strictly convex* BCQPs. In this chapter we describe our new algorithm for solving *nonconvex* BCQPs, which we call NC-DS, since it may solve NonConvex problems. In particular, our new method includes additional machinery to handle the complications that may arise when the problem is nonconvex. First, we use negative curvature directions when they are encountered in the CG iteration, which we call *negative curvature CG steps* (see Section 4.1). Second, to promote convergence to points satisfying second-order conditions, we use Lanczos iterations to compute, when needed, approximations to the left most eigen-pairs, which result in a *negative curvature Lanczos step* (see Section 4.2).

4.1 A negative curvature CG step

If the CG Algorithm 2 finds a direction of nonpositive curvature, i.e. $s_j^T H s_j \leq 0$, then an alternative course of action must be taken. We choose to simply minimize the objective function along either the direction s_j (if s_j is not a descent direction) or along s_j (if s_j is a descent direction) subject to remaining feasible. This is achieved by using Algorithm 6 below.

Algorithm 6 The negative curvature CG step.

```
1: available constants: $\{\eta, \rho\} \subset (0,1)$.
2: input: vector $s_j$ from Algorithm 2 that satisfies $s_j^T H s_j \leq 0$.
3: if $s_j^T g(x_j) < 0$ then
4:     Set $s_j \leftarrow -s_j$.                    ▷ ensure that $s_j$ is not a descent direction.
5: end if
6: Set $\alpha_{feas} \leftarrow \max\{\alpha : x_j - \alpha s_j \in \Omega\}$.
7: Set $x_{j+1} \leftarrow x_j - \alpha_{feas} s_j$.
```

4.2 A negative curvature Lanczos step

One final step must be considered in order to escape first-order KKT points that are not necessarily minimizers. Specifically, if the current iterate, say x_k, is near a first order KKT point in the sense that

$$\nu(x_k) \leq \tau_{stop} \tag{4.1}$$

for some stopping tolerance $\tau_{stop} > 0$, then we must be prepared to compute a sufficiently large negative curvature direction in the space of free variables. In our algorithm this amounts to either verifying that x_k is an approximate second-order point that satisfies

$$\lambda_{min}(H_{\mathcal{F}\mathcal{F}}) \geq -\tau_{stop}, \tag{4.2}$$

or calculating a pair (s_k, λ_k) that approximates the leftmost eigen-pair of the reduced Hessian matrix $H_{\mathcal{F}\mathcal{F}}$ by satisfying

$$[s_k]_{\mathcal{A}} = 0 \quad \text{and} \quad [s_k]_{\mathcal{F}}^T H_{\mathcal{F}\mathcal{F}} [s_k]_{\mathcal{F}} \leq -\tfrac{1}{2}\tau_{stop} \|[s_k]_{\mathcal{F}}\|^2. \tag{4.3}$$

This can be achieved, for example, via the matrix-free Lanczos algorithm,[50] which is practical for very large-scale sparse problems. If both (4.1) and (4.2) are satisfied, we return x_k as the approximate second-order point (see Definition 1.5.2). Otherwise, we take the longest step possible subject to remaining feasible along either the direction s_k if $\nabla q(x_k)^T s_k \geq 0$, or along $-s_k$ if $\nabla q(x_k)^T s_k < 0$. This is equivalent to minimizing the objective function q along the ray defined by the direction s_k.

4.3 The complete algorithm

Our new method for solving nonconvex BCQPs is given by Algorithm 7.

Algorithm 7 Algorithm NC-DS for solving nonconvex BCQPs.

```
1: Input: $x_0 \in [l, u]$, $\{\Gamma, \tau_{stop}\} > 0$, and $\bar{\alpha} \in (0, \|H\|^{-1}]$.
2: Set $g_0 \leftarrow Hx_0 - c$ and $s_0 \leftarrow \varphi(x_0)$.
3: for $k = 0, 1, 2, \ldots$ do
4:     if (4.1) holds then                                  ▷ approximate first-order KKT point
5:         Compute approximate eigen-pair $(s_k, \sigma_k)$ satisfying (4.2) or (4.3).
6:         if (4.2) is satisfied then
7:             return approximate second-order KKT point $x_k$.
8:         else                                                      ▷ negative curvature step
9:             if $s_k^T g(x_k) < 0$ then
10:                Set $s_k \leftarrow -s_k$.
11:            end if
12:            Set $\alpha_{feas} \leftarrow \max\{\alpha : x_k - \alpha s_k \in \Omega\}$ and $x_{k+1} \leftarrow x_k - \alpha_{feas} s_k$.
13:            Set $g_{k+1} \leftarrow g_k - \alpha_{feas} H s_k$ and $s_{k+1} \leftarrow \varphi(x_{k+1})$.
14:        end if
15:    else
16:        if (3.2) holds then
17:            if $s_k^T H s_k \leq 0$ then                           ▷ negative curvature CG step
18:                Compute $\alpha_{feas}$ from Algorithm 6 with input vector $s_j = s_k$.
19:                Set $x_{k+1} \leftarrow x_k - \alpha_{feas} s_k$, $g_{k+1} \leftarrow g_k - \alpha_{feas} H s_k$, and $s_{k+1} \leftarrow \varphi(x_{k+1})$.
20:            else
21:                Set $\alpha_{cg} \leftarrow \big(g_k^T \varphi(x_k)\big)/(s_k^T H s_k)$ and $\alpha_{feas} \leftarrow \max\{\alpha : x_k - \alpha s_k \in \Omega\}$.
22:                if $\alpha_{cg} \leq \alpha_{feas}$ then                     ▷ conjugate gradient step
23:                    Set $x_{k+1} \leftarrow x_k - \alpha_{cg} s_k$ and $g_{k+1} \leftarrow g_k - \alpha_k H s_k$.
24:                    Set $\beta_k \leftarrow \big(g_{k+1}^T \varphi(x_{k+1})\big)/\big(g_k^T \varphi(x_k)\big)$ and $s_{k+1} \leftarrow \varphi(x_{k+1}) - \beta_k s_k$.
25:                else                                                  ▷ expansion step
26:                    Set $x_{k+1/2} \leftarrow x_k - \alpha_{feas} s_k$ and $g_{k+1/2} \leftarrow g_k - \alpha_{feas} H s_k$.
27:                    Set $x_{k+1} \leftarrow P_\Omega\big(x_{k+1/2} - \bar{\alpha}\varphi(x_{k+1/2})\big)$.
28:                    Set $g_{k+1} \leftarrow H x_{k+1} - c$ and $s_{k+1} \leftarrow \varphi(x_{k+1})$.
29:                end if
30:            end if
31:        else                                                       ▷ proportioning step
32:            Set $s_k \leftarrow \beta(x_k)$.
33:            Set
                   $\alpha_{min} \leftarrow \begin{cases} (s_k^T g_k)/(s_k^T H s_k) & \text{if } s_k^T H s_k > 0; \\ \infty & \text{otherwise.} \end{cases}$
34:            Set $\alpha_{feas} \leftarrow \max\{\alpha : x_k - \alpha s_k \in \Omega\}$ and $\alpha_k \leftarrow \min\{\alpha_{min}, \alpha_{feas}\}$.
35:            Set $x_{k+1} \leftarrow x_k - \alpha_k s_k$, $g_{k+1} \leftarrow g_k - \alpha_k H s_k$, and $s_{k+1} \leftarrow \varphi(x_{k+1})$.
36:        end if
37:    end if
38: end for
```

Chapter 5

Numerical Results

We created a Matlab implementation of Algorithm 7 and tested it by solving randomly generated problems of the form (1.1) with varying numbers of variables (n) and condition numbers of H (H_{cond}). Specifically, we tested our implementation on all combinations of number of variables in the set $\{10^2, 10^3\}$ and condition numbers for H in the set $\{10^2, 10^4, 10^6, 10^8\}$. Each instance of H was generated by Matlab's *sprandsym* routine. The problems were created so that there were (roughly) an equal number of lower-active, upper-active, and inactive primal variables for at least one of the optimal solutions (since H is indefinite, there may be multiple local minimizers).

We generated 50 problem instances for each of the 8 possible combinations and report in Tables 5.1 and 5.2 the mean and standard deviation of various quantities of performance. Specifically, we provide measures of the following: the number of iterations ("Total Iterations"); the number of negative curvature CG steps ("Neg. Curv.

Steps"); the number of linear CG steps ("CG Steps"); the number of expansion steps ("Expansion Steps"); the number of proportioning steps ("Proportioning Steps"); and the number of matrix-vector products calculated with H ("Hx Products"). We also note that statistics for the negative curvature Lanczos steps (see Section 4.2) were not included since they rarely occurred in practice.

The results in this section were obtained with the following choice of control parameters: the stopping tolerance was $\tau_{stop} = 10^{-5}$, the scalar in (3.2) was $\Gamma = 10^2$, and the fixed step length was $\bar{\alpha} = \frac{1}{2}\|H\|^{-1}$. Although not stated in Algorithm 7, we also imposed an iteration limit of 20000, although this was never reached in our tests. The values chosen for the parameters $\bar{\alpha}$ and Γ were based on empirical performance.

Table 5.1: Results for Algorithm 7. The mean (mean) and standard deviation (s.d.) were computed over 50 randomly generated nonconvex test problems.

		Total Iterations		Neg. Curv. Steps		CG Steps	
n	H_{cond}	mean	s.d.	mean	s.d.	mean	s.d.
1e+02	1e+02	1.1e+02	0.2e+02	4.9e+01	0.9e+01	3.6e+01	1.1e+01
1e+02	1e+04	1.8e+02	0.3e+02	5.3e+01	1.2e+01	1.0e+02	0.3e+02
1e+02	1e+06	2.5e+02	0.9e+02	5.7e+01	1.2e+01	1.5e+02	0.8e+02
1e+02	1e+08	2.3e+02	0.8e+02	5.1e+01	1.3e+01	1.4e+02	0.7e+02
1e+03	1e+02	8.7e+02	0.9e+02	7.1e+02	0.9e+02	1.2e+02	0.2e+02
1e+03	1e+04	1.5e+03	0.1e+03	7.1e+02	0.7e+02	6.5e+02	1.0e+02
1e+03	1e+06	3.9e+03	0.6e+03	6.8e+02	0.6e+02	3.1e+03	0.6e+03
1e+03	1e+08	8.6e+03	3.4e+03	6.5e+02	0.6e+02	7.7e+03	3.4e+03

We now make some observations about the performance of Algorithm 7. Tables 5.1 and 5.2 show that Algorithm 7 is efficient, at least on average. However, since every one of the $400 = 8 \cdot 50$ test problems were solved, we can also conclude that our

Table 5.2: Results for Algorithm 7. The mean (mean) and standard deviation (s.d.) are computed over 50 randomly generated nonconvex test problems.

		Expansion Steps		Proportioning Steps		Hx Products	
n	H_{cond}	mean	s.d.	mean	s.d.	mean	s.d.
1e+02	1e+02	1.6e+01	0.6e+01	4.5e+00	1.2e+0	1.2e+02	0.2e+02
1e+02	1e+04	2.4e+01	0.8e+01	6.1e+00	1.6e+00	2.1e+02	0.4e+02
1e+02	1e+06	3.2e+01	1.3e+01	7.3e+00	1.9e+00	2.8e+02	1.0e+02
1e+02	1e+08	3.0e+01	1.2e+01	7.0e+00	2.0e+00	2.6e+02	0.9e+02
1e+03	1e+02	4.6e+01	1.1e+01	6.4e+00	1.3e+00	9.2e+02	0.9e+02
1e+03	1e+04	1.0e+02	0.1e+02	8.0e+00	1.1e+00	1.6e+03	0.1e+03
1e+03	1e+06	1.7e+02	0.2e+02	9.9e+00	1.6e+00	4.1e+03	0.6e+03
1e+03	1e+08	2.3e+02	0.4e+02	1.1e+01	0.3e+01	8.8e+03	3.5e+03

method is robust, i.e., it solved all of the problems. We do see, however, that the total number of iterations increases with the problem size and the condition number of H, which is typical for active set methods. The number of matrix-vector products increases roughly proportionally to the number of iterations (see Figure 5.1).

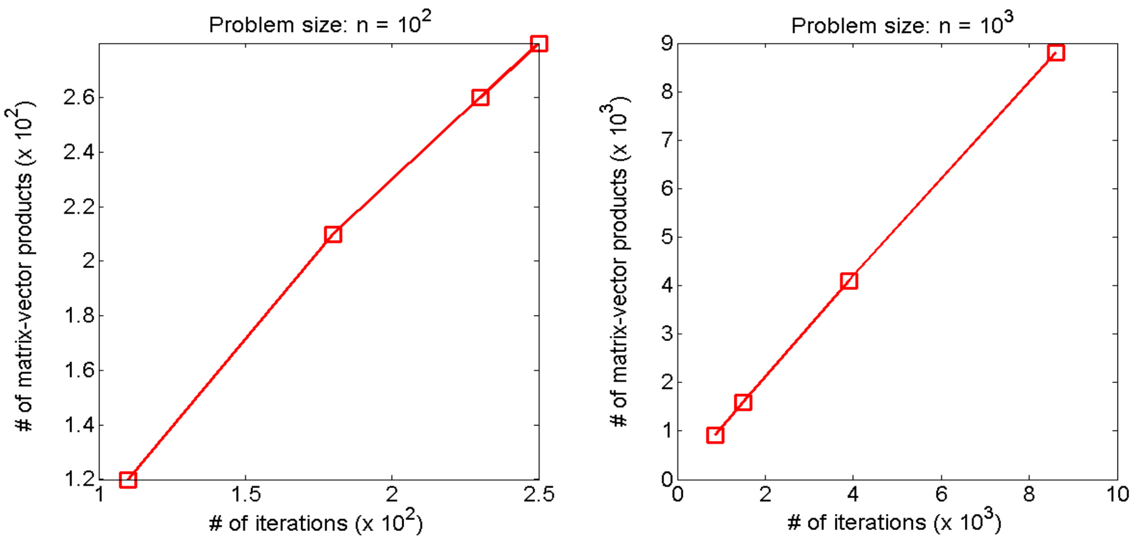

Figure 5.1: Plots of the number of iterations versus the number of matrix-vector products for problem sizes $n = 10^2$ and $n = 10^3$. The points on the curves represent (scaled) values from Tables 5.1 and 5.2 that correspond to condition numbers for H from the set $\{10^2, 10^4, 10^6, 10^8\}$.

For problems of a fixed size, varying the condition number of H had little effect on the number of negative curvature CG steps. In contrast, the number of CG steps increased along with the condition number of H. This behavior was expected since, in general, the number of CG iterations needed to solve a linear system of equations defined with a positive-definite coefficient matrix increases with the condition number.

It is important to discuss the effect that the choice of Γ had on the numerical results. It is clear that larger values of Γ makes it easier to satisfy (3.2). Thus, an extremely large value for Γ would make it "harder" to compute proportioning steps. In fact, they would not be computed until a highly accurate minimizer, over the current set of inactive variables, was obtained. This computationally inefficient choice for Γ is not necessarily better than the other extreme. If Γ was chosen to be extremely small, condition (3.2) would be "easy" to satisfy so that many proportioning steps would be computed. In contrast to the previous case, this would mean that Algorithm 7 would keep stepping off of bounds (proportioning step) far away from any minimizer in the space defined by the current set of free variables. In our results, the value of $\Gamma = 10^2$ represented the best trade-off between these two extremes, and was the value used in the numerical results that we presented.

Chapter 6

Conclusions

We presented a new algorithm in Chapter 4 for solving *nonconvex* bound-constrained quadratic problems (BCQPs). Our method was motivated by ideas that were first introduced by Dostál and Schöberl[1] for *strictly convex* BCQPs. In order to handle negative curvature, we incorporated directions of negative curvature in a fairly natural manner. Although our algorithm was slightly more complicated than that presented by Dostál and Schöberl, it had the added capability of solving both convex and nonconvex BCQPs. In fact, if the BCQP happened to be strictly convex, then our method reduced to the method by Dostál and Schöberl.

The numerical experiments that we presented in Chapter 5 indicated that our method was both efficient and robust on well-conditioned and ill-conditioned problems. That level of performance on problems that spanned a large range of condition numbers is a testament to the dynamic nature of the termination conditions that we used. In particular, these conditions were used to guide the choice of step types that were computed and used during each iteration.

It is important to develop algorithms that are robust in terms of solving problems with diverse condition numbers. Not only are they important in their own right, but they are great choices as the subproblem solver in, for example, augmented Lagrangian methods.[15,51] For this reason, our first direction for future research will be to incorporate our new solver into an augmented Lagrangian framework.

As a second direction of research, we will explore whether ideas similar to those used within our algorithm can be used to develop algorithms for solving other classes of problems. A natural starting point would be the class of linear complementarity problems. Such problems are important in computational mechanics, financial engineering, transportation, and beyond. This line of research would be particular important for *asymmetric* linear complementarity problems for which essentially no practical algorithms for the large-scale case exist.

Bibliography

[1] Z. Dostál and J. Schöberl, “Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination,” *Comput. Optim. Appl.*, vol. 30, no. 1, pp. 23–43, 2005. [Online]. Available: http://dx.doi.org/10.1007/s10589-005-4557-7

[2] A. F. Perold, “Large-scale portfolio optimization,” *Management science*, vol. 30, no. 10, pp. 1143–1160, 1984.

[3] S. R. Gunn *et al.*, “Support vector machines for classification and regression,” *ISIS technical report*, vol. 14, 1998.

[4] E. Osuna, R. Freund, and F. Girosi, “Training support vector machines: an application to face detection,” in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on.* IEEE, 1997, pp. 130–136.

[5] M. A. Figueiredo, R. D. Nowak, and S. J. Wright, “Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems,”

Selected Topics in Signal Processing, IEEE Journal of, vol. 1, no. 4, pp. 586–597, 2007.

[6] Y. Vardi, L. Shepp, and L. Kaufman, "A statistical model for positron emission tomography," *Journal of the American Statistical Association*, vol. 80, no. 389, pp. 8–20, 1985.

[7] G. T. Herman, *Fundamentals of computerized tomography: image reconstruction from projections.* Springer, 2009.

[8] E. Polak, *Computational methods in optimization: a unified approach.* Academic press, 1971, vol. 77.

[9] P. G. Ciarlet, *The finite element method for elliptic problems.* Elsevier, 1978.

[10] J. Sokolowski and J.-P. Zolésio, *Introduction to shape optimization.* Springer, 1992.

[11] E. Philip, W. MURRAY, M. A. SAUNDERS, and M. H. Wright, "User's guide for npsol 5.0: A fortran package for nonlinear programming," *Technical Report SOL 86-6*, 2001.

[12] U. Tulowitzki, "Successive inexact quadratic programming for nonlinear optimization problems." *PhD Dissertation, Yale University, New Haven, Connecticut, USA*, no. 5, 1984.

[13] I. Bongartz, A. Conn, N. Gould, M. Saunders, and P. L. Toint, “A numerical comparison between the lancelot and minos packages for large-scale nonlinear optimization: the complete results,” Tech. Rep. 97/14, Department of Mathematics, FUNDP, Namur, Belgium, Tech. Rep., 1997.

[14] E. G. Birgin and J. M. Martínez, *Practical Augmented Lagrangian Methods for Constrained Optimization.* SIAM, 2014, vol. 10.

[15] A. R. Conn, N. I. M. Gould, and P. L. Toint, “A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds,” *SIAM J. Numer. Anal.*, vol. 28, pp. 545–572, 1991.

[16] M. Hintermüller, K. Ito, and K. Kunisch, “The primal-dual active set strategy as a semismooth Newton method,” *SIAM J. Optim.*, vol. 13, no. 3, pp. 865–888 (electronic) (2003), 2002. [Online]. Available: http://dx.doi.org/10.1137/S1052623401383558

[17] B. Contesse, “Une caractérisation complète des minima locaux en programmation quadratique.” *Numerische Mathematik*, vol. 34, pp. 315–332, 1980.

[18] A. Forsgren, P. Gill, and W. Murray, “On the identification of local minimizers in inertia-controlling methods for quadratic programming,” *SIAM journal on matrix analysis and applications*, vol. 12, no. 4, pp. 730–746, 1991.

[19] P. E. Gill, W. Murray, and M. A. Saunders, “User’s guide for QPOPT 1.0: a For-

tran package for quadratic programming," Department of Operations Research, Stanford University, Stanford, CA, Report SOL 95-4, 1995.

[20] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "A schur-complement method for sparse quadratic programming." DTIC Document, Tech. Rep., 1987.

[21] J. T. Betts and P. D. Frank, "A sparse nonlinear optimization algorithm," *Journal of Optimization Theory and Applications*, vol. 82, no. 3, pp. 519–541, 1994.

[22] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization.* London: Academic Press Inc. [Harcourt Brace Jovanovich Publishers], 1981.

[23] R. Fletcher, *Practical methods of optimization.* John Wiley & Sons, 2013.

[24] J. Nocedal and S. J. Wright, *Numerical Optimization.* New York: Springer-Verlag, 1999.

[25] L. Chen, Y. Wang, and G. He, "A feasible active set qp-free method for nonlinear programming," *SIAM Journal on Optimization*, vol. 17, no. 2, pp. 401–429, 2006.

[26] P. E. Gill, W. Murray, and M. A. Saunders, "Users guide for sqopt version 7: Software for large-scale linear and quadratic programming," 2008.

[27] N. I. Gould and P. L. Toint, "An iterative working-set method for large-scale nonconvex quadratic programming," *Applied Numerical Mathematics*, vol. 43, no. 1, pp. 109–128, 2002.

[28] D. Goldfarb and A. Idnani, “A numerically stable dual method for solving strictly convex quadratic programs,” *Mathematical programming*, vol. 27, no. 1, pp. 1–33, 1983.

[29] N. L. Boland, “A dual-active-set algorithm for positive semi-definite quadratic programming,” *Mathematical Programming*, vol. 78, no. 1, pp. 1–27, 1996.

[30] A. A. Goldstein, “Convex programming in hilbert space,” *Bulletin of the American Mathematical Society*, vol. 70, no. 5, pp. 709–710, 1964.

[31] E. S. Levitin and B. T. Polyak, “Constrained minimization methods,” *USSR Computational mathematics and mathematical physics*, vol. 6, no. 5, pp. 1–50, 1966.

[32] R. S. Dembo and U. Tulowitzki, *On the minimization of quadratic functions subject to box constraints.* Yale University, Department of Computer Science, 1984.

[33] D. P. Bertsekas, “Projected newton methods for optimization problems with simple constraints,” *SIAM Journal on control and Optimization*, vol. 20, no. 2, pp. 221–246, 1982.

[34] E. K. Yang and J. W. Tolle, “A class of methods for solving large, convex quadratic programs subject to box constraints,” *Mathematical Programming*, vol. 51, no. 1-3, pp. 223–228, 1991.

[35] J. J. Moré and G. Toraldo, “On the solution of large quadratic programming problems with bound constraints,” *SIAM Journal on Optimization*, vol. 1, no. 1, pp. 93–113, 1991.

[36] D. P. Robinson, L. Feng, J. M. Nocedal, and J.-S. Pang, “Subspace accelerated matrix splitting algorithms for asymmetric and symmetric linear complementarity problems,” *SIAM Journal on Optimization*, vol. 23, no. 3, pp. 1371–1397, 2013.

[37] S. J. Wright, *Primal-dual interior-point methods.* Siam, 1997, vol. 54.

[38] R. J. Vanderbei, “Loqo: An interior point code for quadratic programming,” *Optimization methods and software*, vol. 11, no. 1-4, pp. 451–484, 1999.

[39] C. Mészáros, “The bpmpd interior point solver for convex quadratic problems,” *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 431–449, 1999.

[40] Y. Ye and E. Tse, “An extension of karmarkar’s projective algorithm for convex quadratic programming,” *Mathematical Programming*, vol. 44, no. 1-3, pp. 157–179, 1989.

[41] Y. Ye, “An extension of karmarkars algorithm and the trust region method for quadratic programming,” in *Progress in Mathematical Programming.* Springer, 1989, pp. 49–63.

[42] T. F. Coleman and J. Liu, “An interior newton method for quadratic programming,” Cornell University, Tech. Rep., 1993.

[43] A. V. Fiacco and G. P. McCormick, *Nonlinear programming: sequential unconstrained minimization techniques.* Siam, 1990, vol. 4.

[44] P. Carbonetto, “Intuition behind primal-dual interior-point methods for linear and quadratic programming.”

[45] D. B. Ponceleon, “Barrier methods for large-scale quadratic programming,” DTIC Document, Tech. Rep., 1991.

[46] S. Mehrotra, “On the implementation of a primal-dual interior point method,” *SIAM Journal on optimization*, vol. 2, no. 4, pp. 575–601, 1992.

[47] J. Dominguez and M. D. González-Lima, “A primal-dual interior-point algorithm for quadratic programming,” *Numerical Algorithms*, vol. 42, no. 1, pp. 1–30, 2006.

[48] M. B. Milam, K. Mushambi, and R. M. Murray, “A new computational approach to real-time trajectory generation for constrained mechanical systems,” in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 1. IEEE, 2000, pp. 845–851.

[49] A. R. Conn, N. I. Gould, and P. L. Toint, *Trust region methods.* Siam, 2000, vol. 1.

[50] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators.* United States Governm. Press Office, 1950.

[51] F. E. Curtis, H. Jiang, and D. P. Robinson, "An adaptive augmented lagrangian method for large-scale constrained optimization," *Mathematical Programming*, pp. 1–45, 2013.

Biographical Sketch

Hassan Mohy-ud-Din was born and raised in Lahore, Pakistan. He earned a BS in Electronics Engineering from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology in June 2006. He pursued a career in teaching, serving as Lecturer at the University of Engineering and Technology Lahore ($2006-08$) and Development Engineer at the School of Science and Engineering, LUMS ($2008-09$).

Hassan joined the Department of Electrical and Computer Engineering (ECE), Johns Hopkins University in 2009 on an ECE fellowship to read for a PhD degree. His expected graduation is in January 2015. He is also pursuing an MA (thesis) in Applied Mathematics and Statistics. His research lies at the intersection of applied mathematics and medical imaging. His research work has won the 2014 Bradley-Alavi fellowship from SNMMI and 2014 SIAM Student Travel Award. He has presented his work at various conferences and universities and carries a teaching experience of over 9 years. He has also made academic trips to Canada, France, and Hong Kong.

Post PhD, Hassan is expected to join Yale University to pursue this Postdoctoral studies under Professor Chi Liu and Professor Richard E. Carson.